Intelligent Automation And Soft Computing, 2018

Vol.24,no.4,883–899

Hybrid Architecture for Autonomous Load Balancing in Distributed Systems based on Smooth Fuzzy Function

Moazam Ali, Susmit Bagchi*

Department of Aerospace and Software Engineering (Informatics),
Gyeongsang National University, Jinju, South Korea

ABSTRACT

Due to the rapid advancements and developments in wide area networks and powerful computational resources, the load balancing mechanisms in distributed systems have gained pervasive applications covering wired as well as mobile distributed systems. In large-scale distributed systems, sharing of distributed resources is required for enhancing overall resource utilization. This paper presents a comprehensive study and detailed comparative analysis of different load balancing algorithms employing fuzzy logic and mobile agents. We have proposed a hybrid architecture for integrated load balancing and monitoring in distributed computing systems employing fuzzy logic and autonomous mobile agents. Furthermore, we have proposed a smooth and composite fuzzy membership function in order to model fine grained load information in a system. The simulation study and a detailed qualitative as well as quantitative analysis of algorithmic performances are presented. Lastly, a deployment environment is described.

KEYWORDS: Distributed computing, load balancing, fuzzy logic, smooth function, load monitoring, software agents.

1 INTRODUCTION

IN distributed systems, the load balancing mechanism is the key factor to achieve improved processing speed and optimum utilization of distributed resources. The application of fuzzy logic in designing load balancing algorithms is an attractive research approach in the presence of uncertainties. In distributed systems, fuzzy logic based load balancing approaches are used to balance overall workload in order to avoid generation of overloaded and under-loaded nodes (Rajani et al., 2015; John et al., 2003). In general, fuzzy logic is composed of fuzzy sets and fuzzy rule-base to model and make decisions under uncertainties (Sabahi et al., 2016; Fu et al., 2009). The load balancing algorithmic outputs have a high level of uncertainties which can be benefitted from applications of fuzzy logical reasoning (Emami et al., 1998). Fuzzy inference is a decision making process, which formulates mapping from a given input set to an output set by using fuzzy membership function (Seth et al., 2012). Membership functions are used in the fuzzification and defuzzification steps of a fuzzy logic based system to map crisp input values to fuzzy linguistic terms and vice versa. In fuzzy logic different types of membership functions are proposed by the researchers such as, triangular, trapezoidal, Gaussian, bell sigmoidal and polynomial functions (Garibaldi et al., 2003). However, most of the functions specified are non-smooth in nature and, in general are not C^{∞} class. The non-smooth functions have sharp boundaries which restrict the function from collecting fine grained information on continuous basis, because such functions are not differentiable everywhere. Therefore, loss of information affects the overall performance of a system.

In order to overcome this problem, we have proposed a smooth and composite membership function in C^{∞} class, which is able to extract fine-grained information providing flexible and robust analysis. A hybrid architecture for distributed load monitoring and balancing is proposed employing fuzzy logic and autonomous mobile agents. Furthermore, we have presented a comparative analysis of different load balancing algorithms to enable users to gain insight into applications of

CONTACT Susmit Bagchi profsbagchi@gmail.com

different algorithms under different computing environments.

1.1 Motivation

In large scale distributed systems, load balancing is the key factor for efficient utilization of resources and enhancing overall system performance. In the literature, a number of different load balancing approaches are employed, however, fuzzy logic based load balancing approaches have outnumbered the other approaches. The attractions of fuzzy logic based load balancing approaches are their inherent abilities to formulate nonlinear functions of arbitrary complexities (Seth et al., 2012). The researchers have proposed a two level fuzzy based approach for dynamic load balancing to characterize uncertainty in distributed systems (Ivanisenko et al., 2015). In this approach processor speed and the queue length of nodes are used to balance the load of nodes in cluster level. The computing capacity and average queue length of clusters are used to balance load of a cluster in cloud level by using fuzzy logical approach (Ali et al., 2016). This approach have achieved improved average response time and throughput. However, this approach causes high intercommunication cost between cloud load manager and cluster load manager in a system. In another approach, the researchers have proposed a fuzzy grouping technique which utilizes a membership graph representing the current status of CPU time and memory space for inferring service priority and load distribution (Ahn et al., 2007). This approach has achieved improved overall response time and throughput. However, this approach causes high overhead intercommunication, which is created by transmitting load information to load-collector frequently.

In this paper, we have proposed a smooth and composite fuzzy membership function to collect fine grained information from the available nodes in a distributed system. We have also presented a hybrid architecture for load monitoring and balancing employing fuzzy logic as well as mobile agents. The main features of our proposed hybrid architecture are intelligent monitoring and decision making in a hybrid platform autonomously aiming high throughput and low response time in a system.

The main contributions of this paper are as follows.

- A detailed comparative study is presented covering different fuzzy logic based load balancing algorithms as well as applications of mobile agents in load monitoring in distributed systems.
- We have proposed a novel fuzzy membership function for collecting fine-grained information of nodes for decision making and load monitoring in our hybridized model.
- A hybrid architecture is designed to monitor, estimate and distribute load in a large-scale distributed system to achieve load-balancing with autonomy.
- A detailed comparative analysis of various algorithmic performances is presented by considering fuzzy logic based as well as traditional (non-fuzzy) algorithms.

Rest of the paper is organized as follows. Section 2 illustrates load balancing algorithms and software agents. Section 3 presents load monitoring using agents. Section 4 represents fuzzy logic based load balancing architectures. Section 5 presents designing of smooth functions and hybrid architecture. A detailed comparative analysis of different load balancing algorithms is presented in section 6. Section 7 presents implementation directions. Lastly, section 8 concludes the paper.

2 TAXONOMY OF LOAD BALANCING ALGORITHMS

IN general, load balancing approaches are divided into two main categories such as, static load balancing and dynamic load balancing (Rajani er al., 2015; Ivanisenko et al., 2015). The taxonomy of load balancing approaches is illustrated in Figure 1.

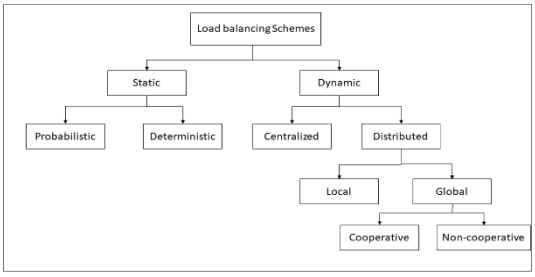

Figure 1. Taxonomy of load balancing approaches in distributed systems

Static load balancing approaches use statistical information for decision making purposes. This approach works with fixed number of processes known at compile time. Furthermore, static load balancing algorithms are divided into two types such as, probabilistic and deterministic algorithms. In probabilistic approaches, attributes of the systems are used while in deterministic approaches the characteristics of processes and properties of nodes are used. In deterministic approaches optimization is difficult and costs are comparatively higher (Ivanisenko et al., 2015).

On the other hand, in dynamic load balancing (DLB), decisions are based on the current state of the systems (i.e. system load) and the number of processes are not fixed (El-Abd, 2002; Khan et al., 2015). As illustrated in Figure 1, DLB is further divided into centralized and distributed load balancing algorithms. In centralized load balancing, all nodes share their

load information with one single node (master or coordinator) for making the load balancing decisions. In distributed load balancing, the load balancing algorithm is implemented by replicating it on all the nodes in a system (Naaz et al., 2010). From the functional point of view, the distributed load balancing is divided into local and global load balancing algorithms. In local load balancing algorithm, the decisions of load balancing are limited to a single group. However, in global load balancing algorithm, the scope of decision making is not limited to a certain group of network nodes. Furthermore, in terms of decision-making process, the global load balancing algorithm can be cooperative (each node in distributed systems cooperates with each other to make load balancing decisions) and non-cooperative (each node in distributed systems is autonomous and load balancing decisions are made independently).

3 LOAD MONITORING USING AGENTS

SOFTWARE agents are emerging technological entities capable of autonomously managing and maintaining distributed systems (Nwana, 1996). Software agents are goal oriented and act accordingly for achieving user's goals (Long et al., 2011; Horvat et al., 2000). Functional classification of software agents is illustrated in Figure 2. It is shown that different agents have different inherent behavioral properties such as autonomy, learning and, cooperation as explained by (Franklin et al., 1996; Wooldridge et al., 1995).

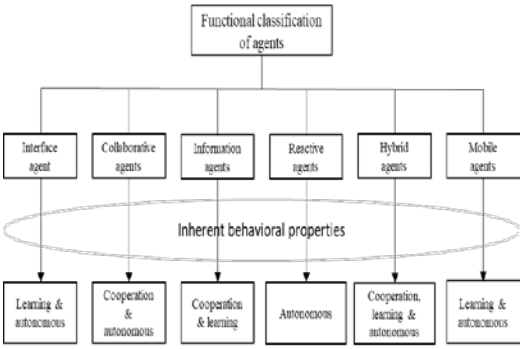

Figure 2. **Functional classification and properties of agents**

Agent-based load monitoring is employed in distributed systems for enhancing system efficiency and for better resources utilization. Researchers have proposed different approaches for load monitoring in distributed systems (Brandt et al., 2009; Jiang et al., 2015). However, in this paper, we will be focusing only on agent-based monitoring systems. Following are the most relevant approaches we have considered specifically for this paper.

3.1 Agent-based monitoring using fuzzy logic

Funika W. et al. have proposed agent-based monitoring using fuzzy logic and rule base. In this approach, monitoring is done by Semantic-based Autonomous Monitoring and Management system (SAMM) (Funika et al., 2011). The administrators are provided a GUI interface for controlling and monitoring agents. In SAMM system, agents can exchange information with each other by intercommunication. In SAMM-CA system, there are five modes of operations with slightly different functional capabilities. Schematic diagram of SAMM-CA design is illustrated in Figure 3 as proposed in (Funika et al., 2011).

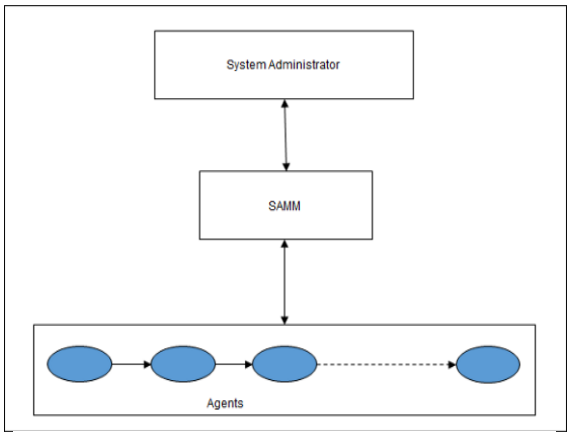

Figure 3. **Schematic diagram of SAMM-CA**

3.2 Agent-based adaptive monitoring

The agent based adaptive monitoring system is based on agents and multi-layered framework consisting of communication layer, interpreter layer and, management layer (Kwon et al., 2006). The function of communication layer is to manage protocol components (HTTP, Custom and Multicast); exchange of messages is performed by interpreter layer by using appropriate query components. The agent management layer is responsible for controlling the lifecycle of agents and the actual execution of tasks according to user's requests. The schematic diagram of the multi-layered framework is illustrated in Figure 4 as proposed in (Kwon et al., 2006).

4 FUZZY LOGIC BASED LOAD BALANCING ARCHITECTURES

IN this section, we have studied different fuzzy logic based load balancing approaches by using distributed and centralized methods.

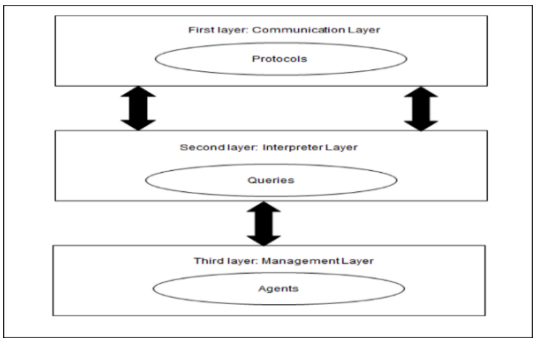

Figure 4. Schematic diagram of Multi-layered framework

4.1 Fuzzy load balancing for homogeneous distributed systems

Alakeel M. A. et al. have proposed a fuzzy dynamic load balancing approach for homogenous distributed systems. The usage of fuzzy logic is to deal with inaccurate load information while making the load distribution decisions and maintaining overall system stability (Alakeel, 2012). This approach specifies how, when and by whom (which node) load balancing mechanism is implemented. For obtaining the current load of the system, load balancer nodes broadcast request message to all the connected nodes for determining their current status. Load balancer assigns a fuzzy membership value to the load of individual node including itself in the interval of [0, 1]. The fuzzy membership values are assigned according to the relative load of the node to the overall system load. The assignment of the fuzzy membership value is achieved by forming a fuzzy set defined as, LOADED = {light, normal, heavy} representing the respective load of a system. This mapping is performed by considering the probability of migrating a task from an overloaded node to an under-loaded node (Giarratano, 1989).

4.2 Fuzzy load balancing for heterogeneous distributed systems

Karimi A. et al. have proposed a dynamic load balancing algorithm based on fuzzy logic (Karimi et al., 2009). The algorithm considers distributed network consisting of n nodes where each node is a complex combination of multiple types of resources (memory, CPUs, disks and switches etc.). All the nodes in the distributed network are heterogeneous in configuration settings such that, configurations of the resources at one node are different from the configurations of the physical resources of the other nodes. The proposed system model is consisting of routing table, load index, cost table and a fuzzy controller as illustrated in Figure 5 (Karimi et al., 2009).

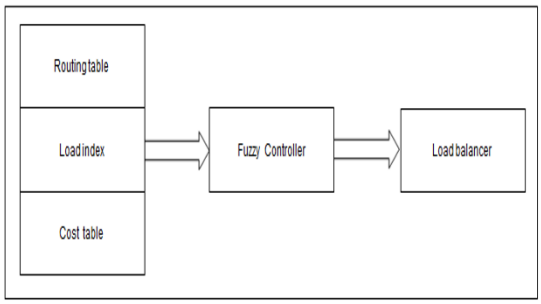

Figure 5. Schematic diagram of dynamic system model

The routing table is used to store the information about communication links between the nodes in the system. The load values of the nodes are indicated by the load index. In order to deal with load imbalances, the researchers have proposed four policies governing the actions of load balancing which are, information policy, transfer policy, location policy and selection policy (Karimi et al., 2009). In order to determine the node status such as, receiver, sender or neutral, a fuzzy rule-based controller is proposed. The cost table is used to provide information regarding communication costs and the number of heavily loaded nodes. Load index and routing table are used to compute the cost table and, the heavily loaded nodes are extracted from the obtained cost table. The threshold value of load index is classified into five categories, which are mapped between 0 to w and threshold is fixed at s (where w is the maximum value of load index and s is the threshold value) (Karimi et al., 2009). Authors have considered five different fuzzy levels to describe the fuzzy load index values represented as, very lightly loaded, lightly loaded, moderately loaded, heavily loaded and very heavy loaded. The proposed dynamic load balancing algorithm employing fuzzy logic shows significantly better response time than the round robin and randomized algorithms (Karimi et al., 2009).

4.3 Fuzzy load balancing for centralized system

Researchers have proposed hierarchical architecture and centralized method for load balancing using fuzzy logic (Moosavi et al., 2011). In centralized load balancing method, communication overload from data collection is considerably decreased. In the proposed architecture, load balancing task is divided into two levels: (1) group level global manager and, (2) local manager at the node level, as illustrated in Figure 6 (Moosavi et al., 2011).

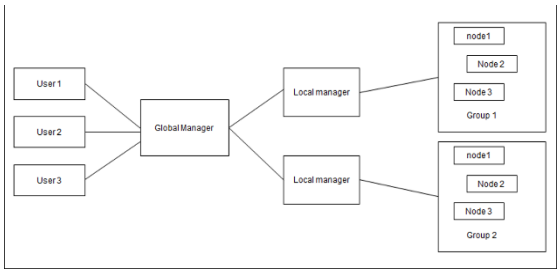

Figure 6. **Schematic diagram of hierarchical architecture**

The node, acting as local manager functions as a designated representative of its own group, performs load balancing tasks and communicates with other local managers. Nodes in each group only interact with its own group managers. Global managers will communicate with designated representatives (managers) of each group. In the proposed algorithm, weight assignment is done on two levels such as, global level and local manager levels. First, the global manager will send tasks to local managers and then local managers will send tasks to nodes in the respective groups based on weight assignments. The fuzzy controller is responsible for weight assignment. The criteria for assigning the weights are based on two variables i.e. current load and waiting time of the last processed task. The load balancing process is handled in three stages such as, (1) Data Collection Stage, (2) Weight Assignment Stage and, (3) Distribution Stage (Moosavi et al., 2011). In the proposed model, current load and the waiting time information about last processed task in each node are considered as input to fuzzy controller. The result of the proposed algorithm shows that it has improved performance as compared to static and dynamic algorithms in terms of Drop Rate, Throughput and Response Time (RT).

4.4 Fuzzy load balancing for multi-level centralized architecture

Barazandeh I. et al. have proposed a fuzzy logic based control approach to reflect global state of uncertainty in load distribution decisions (Barazandeh et al., 2009). The proposed model is hierarchical in structure and centralized in type. The input variables to fuzzy controller are current load and waiting time of last completed task. The fuzzy controller infers specific weights called bias as output (Barazandeh et al., 2009). The migration of loads is based on the computed biases. In their design structure, load balancing is done into two levels such as, (1) Group level by Global Load Balancer (GLB), and, (2) Node level by Local Load Balancer (LLB), as illustrated in Figure 7 (Barazandeh et al., 2009). The multi-level model is significantly better in load management and it reduces the communication overhead. In the proposed model, every group contains one node which is called Designated Representative (DR). This DR node is responsible for local load balancing and it communicates with Global Load Balancer (GLB). In each group, nodes will only connect and communicate to their own group's DR. GLB is connected to the designated representative of each group. Mechanisms of load balancing are centralized at both levels. In the beginning of biasing, a set of new fuzzy rule-based biases are produced based on the collected information from both groups and nodes in the system. GLB distributes the arrived tasks in groups based on group biases. Once the tasks arrived at each DR of the group, they are distributed among all the nodes in that particular group according to biases of nodes. In the proposed model each DR maintains a control box which is used for fuzzy based biasing. It means that only DR nodes implement biasing in this model (Barazandeh et al., 2009).

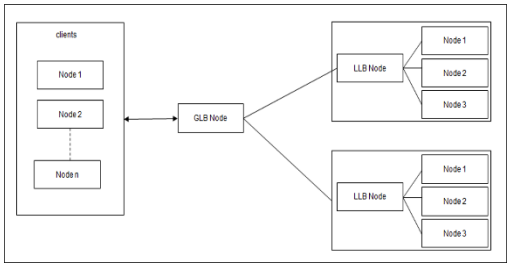

Figure 7. **Schematic diagram of multi-level architecture**

5 DESIGNING SMOOTH FUNCTIONS AND HYBRID ARCHITECTURE

5.1 Designing smooth composite function

A fuzzy membership function maps a set of input values into an interval set, where the sets are crisp. The degree of membership values varies in between [0, 1], which is known as membership grade (Barazandeh et al., 2009). In the proposed model we have formulated a smooth composite fuzzy membership function in C^{∞} class. A smooth function $f(x)$ is continuous at point a if $\lim_{x \to a} f(x) = f(a)$ and $f(x)$ is continuously differentiable on a set X representing a C^{∞} class. It means that, $\forall a \in X$ the Taylor expansion of $f(x = a)$ exists and, $D^n f(x)|_{x=a} \in (-\infty, +\infty)$, where $n \in Z^+ \cup \{0\}$. The proposed membership function is designed to solve the problem of load balancing in distributed systems by equalizing load among all the nodes. In order to distribute load, our composite membership function collects fine grained

information from all the nodes and, maps into unit intervals following smooth composite function. The proposed composite membership function is given below.

Let a real-valued function be represented as, $f:(A\subset R)\to R^{+}\cup\{0\}$ where,

$$f(x\in A)=|\sin x|,\quad A=\{x:0\le x\le 3\pi\} \tag{1}$$

Let a second real-valued function be given by,

$$g:(B\subset A)\to R^{+}\cup\{0\},\quad g(x\in B)=|\cos x|,\quad B=\{x:\pi/2\le x\le 5\pi/2\} \tag{2}$$

The smooth and composite fuzzy membership function is generated as,

$$\mu:A\to[0,1],\quad \mu(A)=f(A)\cup g(B) \tag{3}$$

The composite profile of fuzzy membership function is given in Figure 8. In the proposed membership function there are overlapping regions. The importance of these overlapping regions is to give robustness to the controller to fire at least one rule for each possible input. In addition, overlapping regions are important for completeness. The completeness of a fuzzy set is represented by total overlap $\omega>0$ describing the characteristics of a reference set. When the ω value is decreased, the partition of the universe of discourse of fuzziness is also decreased. The decrease in completeness level causes inefficient control because greater region of universe of discourse will be prone to larger set of activated rules. The cross-over points between two overlapping smooth functions are computed as given below,

$$E\subset\mu(A),\quad E=(f(A)\cap g(B))\setminus\{0\} \tag{4}$$

The entire overlapping area can be computed as,

$$\omega=\sum_{k=1}^{4}\left(\int_{I_k\subset B}f(x)dx+\int_{J_k\subset B}g(x)dx\right),\quad f(\bigcup_{k=1}^{4}I_k\cap\bigcup_{k=1}^{4}J_k)=g(\bigcup_{k=1}^{4}I_k\cap\bigcup_{k=1}^{4}J_k)=E \tag{5}$$

5.2 Creating rule-base

The fuzzy rule-base (FRB) system creates a set of rules in the form of (if-then) statements for generating control outputs. In fuzzy rule-base, *if* part is known as antecedent, and *then* part is called consequent (Jeong et al., 2004). In our rule-base system we have two antecedent covering resources such as, CPU and RAM. However, the linguistic terms in our rule base system are: *very_lightly_loaded*, *lightly_loaded*, *moderately_loaded*, *highly_loaded* and *very_highly_loaded*. The corresponding consequent linguistic terms are: *no_migrate*, *calculate_again* and, *migrate_load*. Fuzzy reasoning (if-then) rules have two basic features. In our proposed model the first rule will partially match input data to make an inference. In the second step, fuzzy inference system will combine all the conclusions of the employed rules to form a final outcome. A set of inference rules of our proposed model is defined in Table 1.

The fuzzy inference rules and membership functions are the processes to carry out the fuzzification and defuzzification. The inference rules illustrated in Table 1 show a decision process that is generated based on both antecedents.

5.3 Hybrid architecture

The proposed system architecture following hybridization of mobile agents and fuzzy logic based load balancing framework is illustrated in Figure 9. The proposed model is distributed in nature, such that it employs load distribution in a distributed manner by replicating the control algorithm on each and every node. The replication will remove the bottleneck and a single point of failure. All nodes in the distributed systems will monitor and balance their local workload accordingly. Moreover, the combination of fuzzy logic

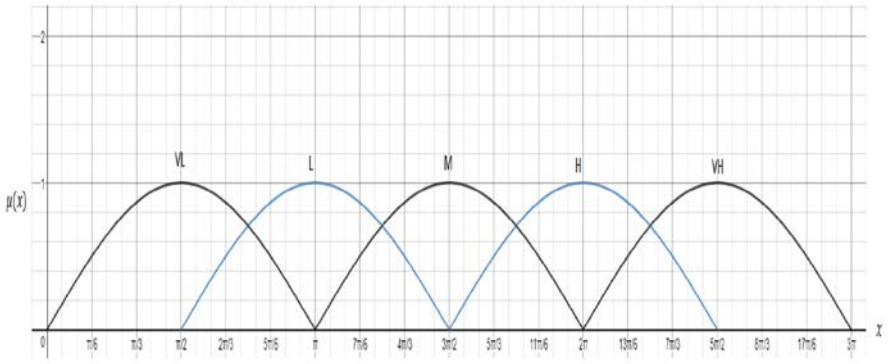

Figure 8. Composite fuzzy membership function

Table 1. Fuzzy inference rules

			CPU		
RAM	VH	H	M	L	VL
VH	NM	NM	NM	NM	CA
H	NM	NM	NM	CA	ML
M	NM	NM	CA	ML	ML
L	NM	CA	ML	ML	ML
VL	CA	ML	ML	ML	ML

Legends: VH: very_highly_loaded, H: highly_loaded, M: moderately_loaded, L: lightly_loaded, VL: very_lightly_loaded, ML: migrate_load, CA: calculate_again, NM: no_migrate.

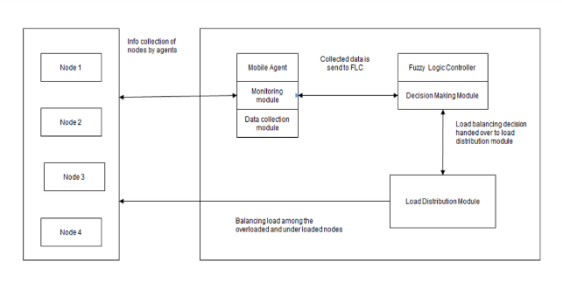

Figure 9. Fuzzy logic and mobile agent based hybrid architecture

and mobile agents forms a hybrid model. In the hybrid model, the fuzzy logic will make decisions and mobile agents will collect systems load information in distributed environment. The key components of our proposed model are: Mobile agents, Fuzzy logic controller (FLC) and Load distribution module. The mobile agent is used to collect the current load, searching and discovering of resources, to update status information about connecting network links and, to perform monitoring for potential nodes entering or leaving the system autonomously.

On the basis of status information collected from mobile agents, the fuzzy controller module will take intelligent decisions for load balancing. The load distribution module is used to migrate the load to under-loaded nodes. Load distribution module is acting accordingly on the decision made by the fuzzy controller for transferring the load. In our proposed model, if at any instant a node gets overloaded, then the mobile agent will autonomously execute to collect the current status of all the nodes for making load distribution decisions. The key characteristics of our proposed architecture are intelligent monitoring and decision making in a hybrid platform autonomously aiming high throughput and low response time in a system. The proposed architecture aims to yield high throughput and low response time based on the following design parameters: (a) mobile agents are used to determining the current status (load monitoring) of the available nodes, (b) load balancing decisions are made by a fuzzy logic controller which gives the precise result according to the status of the nodes and, (c) the proposed model is based on distributed approach i.e. each node will make their own decision regarding load balancing reducing response time and increasing throughput.

5.4 Deployment model of mobile agents

The proposed deployment model for mobile agent based monitoring system is illustrated in Figure 10. In our deployment model, we have one monitoring node and four client nodes. The monitoring node communicates with client nodes to monitor their current load status. Mobile agents collect the current status information and send a response message to monitoring node. After sending the status information the mobile agent will migrate to the next node autonomously.

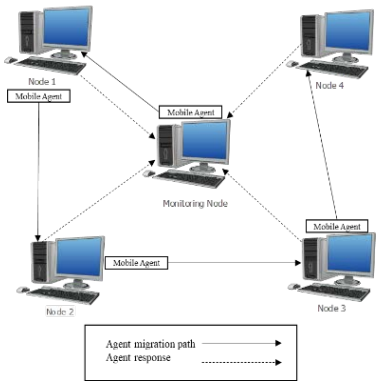

Figure 10. Deployment model of mobile agent monitoring system

The mobile agents migrate to the next node depending on node availability. If a node is added or removed their updated status information are maintained in a database. Unlike traditional load monitoring approaches, our proposed agent based monitoring system enable mobile agents to efficiently collect status information of a node and send this information immediately to monitoring node. The goal of our proposed approach is to reduce the computational time as well as waiting time to improve the overall system performance.

5.5 Simulation study

A simulation study of the designed fuzzy system is conducted to evaluate the response profiles. In this section, the experimental setup is described generating random load by two synthetic benchmarks (Heavy Load and CPU stress). The instantaneously available CPU and RAM resources are fed into fuzzyfication module to generate inference rule sequences according to rule-base. The simulation is evaluated for five different levels of quantization such that, VL: from 0 up to 0.20, L: from 0.21 up to 0.40, M: from 0.41 up to 0.60, H: from 0.61 up to 0.80 and, VH: from 0.81

up to 1. The snapshot of rule firing sequences under random load generation events are illustrated in Figures 11, 12, 13, 14 and, 15, respectively. In the figures, Y-axis represents decision logic from rule-base, where 1: ML, 2: CA and, 3: NM. The simulation is conducted in five sets. Each simulation experiment is conducted for 30 minutes to measure the rule firing sequences under synthetic benchmarks.

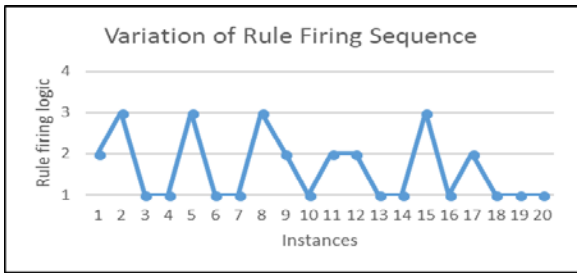

Figure 11. Variation of rule firing sequence (set 1)

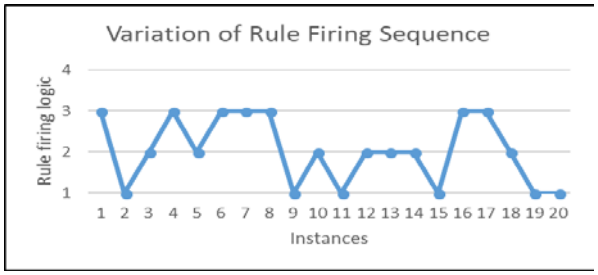

Figure 12. Variation of rule firing sequence (set 2)

Figure 13. Variation of rule firing sequence (set 3)

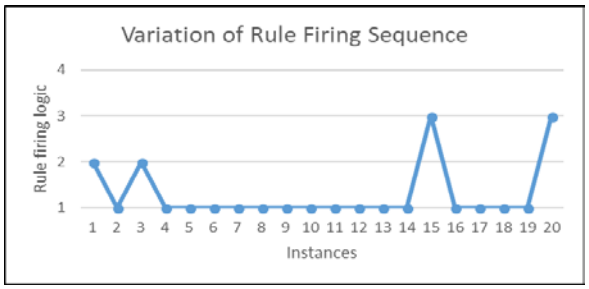

Figure 14. Variation of rule firing sequence (set 4)

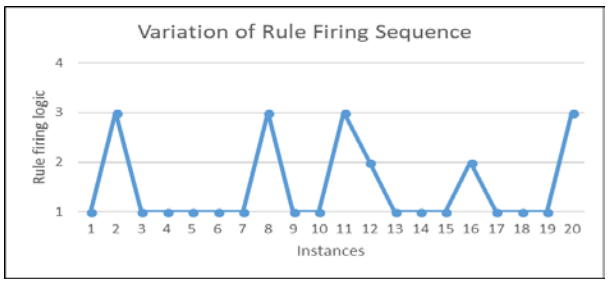

Figure 15. Variation of rule firing sequence (set 5)

5.6 Comparative analysis

The performance of different algorithmic approaches such as, centralized, distributed and hybrid vary considerably. Therefore, it is important to analyze the factors which are affecting the efficiency and performance of the distributed load balancing algorithms (Cheung et al., 2001). Table 2 illustrates the comparative analysis of different fuzzy based load balancing approaches by considering different parameters, where the respective data are gathered, filtered and compared from (Alakeel, 2016; Cheung et al., 2001; Karimi et al., 2009; Ivanisenko et al., 2015; Ahn et al., 2007). In order to measure the efficiency of the fuzzy load balancing algorithms several different indicators/parameters are determined, which are explained below.

5.7 Parametric discussions

5.7.1 Overhead

The overhead is used as parameter to measure the amount of computation done by any load balancing algorithm involving inter-process communication and task migration. In centralized load balancing approach, majority of algorithmic overhead is L (Low). This is because, the centralized approach uses single node or master node for load balancing, which reduces the communication overhead between nodes. In distributed load balancing approach, the ratio of overhead is 50% L (Low) and 50% H (High). The high overhead is due to inter-nodal communication to select appropriate node for load migration. In hybrid load balancing approach, majority of algorithmic overhead is M (Medium). This is because, in a hybrid approach, a distributed system is segmented into groups. The control within each group of nodes is centralized and, the load balancing is distributed globally in a system among the groups. The use of group-wise central node is the main reason for introducing overhead in hybrid load balancing approach.

Table 2. Comparative analysis of different fuzzy load balancing algorithms

Comparative Analysis of Load Balancing Algorithms														
	Centralized				Distributed					Hybrid				
Parameters	RRA-IFC	RA-IFC	Com-IFGS	Sim-IFGS	FA-TLFA	RRA-DOC	RA-DOC	FA-DOC	FESA-DLBA	FA-HDS	SA	TA	RA	
Overhead	H	H	L	L	L	H	H	L	L	M	M	L	H	
Throughput	L	L	H	H	H	L	L	H	H	H	M	M	L	
Process Migration	H	M	M	M	M	L	H	H	H	H	M	M	H	
Response Time	M	L	H	M	L	L	L	H	H	L	H	H	H	
Resource Utilization	L	L	M	M	H	L	L	H	H	H	M	L	L	
Fault Tolerance	L	L	L	L	L	M	L	H	H	M	M	M	L	
Waiting Time	M	H	L	L	M	L	H	L	L	L	×	L	H	
Scalability	M	L	L	M	L	H	×	H	H	L	×	×	×	
Performance	M	L	H	M	H	M	L	H	H	H	M	H	L	
Reliability	L	L	L	L	M	M	L	H	H	H	×	M	×	

Legends: H: High, L: Low, M: Medium, ×: not determined, RRA-DOC: Round Robin Algorithm in Distribute Object Computing, RA-DOC: Random Algorithm in Distribute Object Computing, RRA-IFC: Round Robin Algorithm Intelligent Fuzzy Controller, Com-IFGS: Complex Intelligent Fuzzy Grouping system, Sim-IFGS: Simple Intelligent Fuzzy Grouping System, FA-TLFA: Fuzzy Algorithm in Two Level Fuzzy Approach, RA-IFC: Random Algorithm in Intelligent Fuzzy Controller, FA-DOC: Fuzzy Algorithm in Distributed Object Computing, FESA-DLBA: Fuzzy Enhance Symmetric Algorithm in Dynamic Load Balancing Algorithm, FA: Fuzzy Algorithm in Hybrid Dynamic System, SA: Shortest Algorithm, TA: Threshold Algorithm, RA: Random Algorithm.

5.7.2 Throughput

Throughput parameter is used to compute the total number of tasks whose executions have been completed successfully. Therefore, higher throughput of an algorithm indicates higher performance of the overall system. In centralized load balancing approach, majority of algorithmic throughput is H (High). Because the master node contains prior information of current load status of all the nodes and their rapid distribution results in a high throughput of a system. In distributed load balancing approach, the ratio of throughput is 50% L (Low) and 50% H (High). In hybrid load balancing approach, the majority of algorithmic throughputs are M (Medium). This is because the reselection of the centralized node for load balancing affects the throughput of the system.

5.7.3 Process Migration

Process migration indicates transfer of a task from one node to another node on demand. Reduction in migration time enhances the system performance. In centralized load balancing approach, the majority of load balancing algorithms employ process migration in category M (Medium). This is because only the master node is responsible for load migration however, efficient load migration depends on the number of nodes in a distributed system. In distributed load balancing approach, majority of algorithms perform process migration in category H (High). This is because each node handles load balancing locally, which increases process migration frequency. In hybrid load balancing approach, the ratio of process migration is 50% H (High) and 50% M (Medium). Hence, it is observed that lesser the number of nodes in a group, higher the efficiency of process migration.

5.7.4 Response Time

Response time determines the time taken to complete a task by the load balancer. For better system performance, response time must be kept low. In centralized load balancing approach, the ratio of response time is 50% L (Low) and 50% M (Medium). The response time depends on scale of the overall system; if the number of nodes is high then the response time will be low. Thus, the overall response time will degrade, when the master node processes more requests in case of high number of nodes. In distributed load balancing approach, the ratio of response time is 50% H (High) and 50% L (Low). Selecting appropriate node for load transfer would consume computational time. In the hybrid approach, the majority of algorithmic response time is in category H (High).

5.7.5 Resource Utilization

In order to get the better performance from the overall system, the algorithm must ensure the appropriate use of all the system resources. In centralized load balancing approach, the ratio of resource utilization is 50% L (Low) and 50% M (Medium). The resource utilization efficiency depends on the number of nodes in a system and the corresponding network saturation. In distributed load balancing approach, the ratio of resource utilization is 50% L (Low) and 50% H (High). The 50% L (Low) category indicates that some of the resources are wasted for searching receiver node. In hybrid load balancing approach, majority of algorithmic resource utilization is in category L (Low). Once again, the resources are wasted on reselection of master node, which reduces the efficiency of resource utilization.

5.7.6 Fault Tolerance

Fault tolerance metric measures the ability of an algorithm to work continuously and to perform load balancing uniformly in the event of some failures. An algorithm is said to be efficient if a minor fault cannot degrade the performance of an algorithm. In centralized load balancing approach, all algorithmic fault tolerance is in category L (Low) because of the presence of a single point of failure. Hence, effectively there is no fault tolerance mechanism in centralized load balancing approach. In distributed load balancing approach, the majority of algorithmic fault tolerance is in category H (High). There is no single point of failure in the system and, the load balancing is done by every node. In hybrid load balancing approach, majority algorithmic fault tolerance is in category M (Medium). As each group nominates master node for load balancing, thus if a node crashes then the new master node will be nominated. The crashed master node would result in data loss however, the whole system would not crash.

5.7.7 Waiting Time

Waiting time determines the time period spent by a task in ready queue. Minimum the waiting time better the system performance. In centralized load balancing approach, the ratio of waiting time is in 50% L (Low) and 50% M (Medium) categories, because a large number of tasks would wait in the ready queue for some time. In distributed load balancing, majority of algorithmic waiting time is in L (Low) category. When a node becomes overloaded, it would search for another suitable node to transfer the load. Hence, there is no waiting time for tasks in distributed load balancing approach. In hybrid approach, majority of algorithmic waiting time is in L (Low) category. If the size of a group is kept small and the master node has prior information about nodes, then efficient task migration occurs to realize load balancing within a group.

5.7.8 Scalability

Scalability is the ability of an algorithm to provide optimized results in a system comprised of a finite number of nodes. A system is scalable if despite of gradual increase in the number of nodes, the algorithm continues to perform uniform load balancing. Highly scalable algorithm will result in reliable and stable system. In centralized load balancing approach, majority of algorithmic scalability is in L (Low) category. There is no restriction on adding or removing of nodes in a system. In distributed load balancing approach, majority of algorithmic scalability is in H (High) category. If the number of nodes is increased, then only the inter-nodal communications would increase in distributed load balancing systems. If a node is removed from the system then it will not affect the overall system. In hybrid load balancing approach, scalability is in L (Low) category, because the network is segmented into groups and the master node is assigned to each group. If the number of nodes in a system increases, then network is re-segmented into groups so that each group contains nearly equal number of nodes.

5.7.9 Performance

Performance parameter is used to check the overall efficiency of a system. In centralized load balancing approach, the ratio of performance is 50% H (High) and 50% M (Medium) categories. Performance metric is dependent on various parameters such as number of nodes, average response time and waiting time etc. In distributed load balancing approach, majority of algorithmic performances are in H (High) category, whereas in hybrid load balancing approach, the majority of algorithmic performances are in H (High) category.

5.7.10 Reliability

Reliability is used to measure the system stability. In centralized load balancing approach, majority of algorithmic reliability are in L (Low) category due to the existence of single point of failure in the system. In distributed load balancing approach, majority of algorithmic reliability are in H (High) category, because the failure of one or more nodes will not cause failure of the entire system. In hybrid load balancing approach, the ratio of reliability is in 50% H (high) and 50% M (Medium) categories, because if the selected master node fails then another master node would be re-elected.

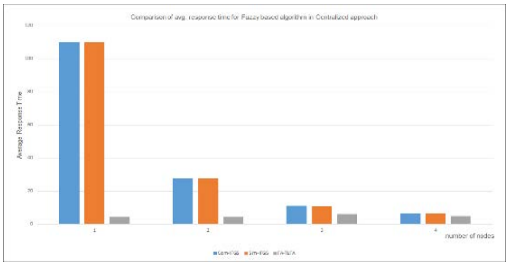

Figure 17. Avg. response time of Complex-IFGS, Simple-IFGS and FA-TLFA in centralized approach

5.8 Analysis of algorithmic performances

In this section we will quantitatively analyze different algorithmic performance in Centralized, Distributed and Hybrid load balancing approaches presented in (Kwork et al., 2004; Ahn et al., 2007; Ali et al., 2016). The comparative studies are performed based on normalized response time and throughput. Comparison of average response time for round robin and random algorithms in centralized approach is illustrated in Figure 16. In order to distribute load in round robin algorithm, the master or server node

distributes load in round robin cycles. In this approach using the distributed system, the average response time is dependent on the number nodes such that increasing the number of nodes will decrease the average response and vice versa. In randomized algorithm, the server node distributes load randomly without any prior information about the status of nodes in a distributed system. However, there is a probability of already overloaded nodes and sending extra load to those nodes will worsen the load balance. For this reason the average response time of the randomized algorithm is higher than the round robin algorithm. In Figure 17, we have compared the performance of complex-IFGS (com-IFGS), simple-IFGS (sim-IFGS) and Fuzzy logic based algorithm FA (TLFA), where the respective data are gathered, filtered and compared from (Ahn et al., 2007; Ali et al., 2016). In case of complex-IFGS and simple-IFGS, the average response time is nearly equal. However, average response time of complex-IFGS is slightly better than simple-IFGS. In case of FA (TLFA) the average response time is improved, because in this approach two levels of load balancing is used which reduces the average response time. Due to two levels load balancing, FA (TLFA) outperforms complex-IFGS and simple-IFGS algorithms as shown in Figure 17.

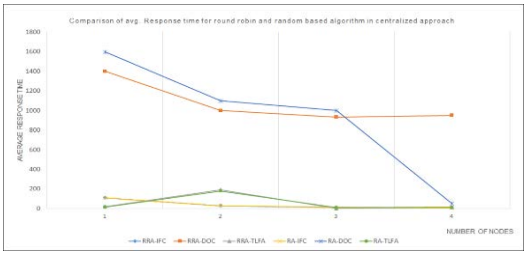

Figure 16. Variations of response time of round robin and random algorithm in centralized approach

In Figure 18, the comparison of throughput is illustrated for round robin algorithm (RRA) and randomized algorithm (RA), where the respective data are gathered, filtered and compared from (Saxena et al., 2012; Kwork et al., 2004; Ahn et al., 2007; Ali et al., 2016). As illustrated in Figure 18, the performance of randomized algorithm is aperiodic and tends to overshoot or undershoot with respect to the number of nodes. In comparison, the throughput of round robin algorithm is better than the randomized algorithm. This is because, in randomized algorithmic approach the load is distributed randomly and if a node is overloaded then it will degrade the performance.

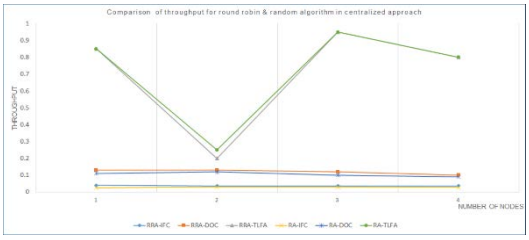

Figure 18. Throughput comparison of Round Robin and Randomized Algorithm in centralized approach

In Figure 19, the comparison of complex-IFGS, simple-IFGS and, Fuzzy algorithmic throughput is illustrated, where the respective data are gathered, filtered and compared from (Ahn et al., 2007; Ali et al., 2016). The graph shows that, in complex-IFGS and simple-IFGS the throughput values are almost equal. In comparison, the throughput of fuzzy algorithm (FA-TLFA) is high, because the fuzzy algorithm is using two levels of fuzzy load balancing approach. In level 1, the system is segmented into groups and every group has its own local load balancer. In level 2, only the local load balancers will communicate with the global load balancer. Thus the communication overhead is reduced and it results in an increase in the overall performance as well as efficiency of the system.

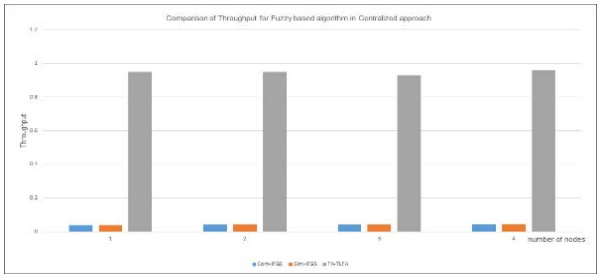

Figure 19. Throughput comparison of Complex-IFGS, Simple-IFGS and FA-TLFA in centralized approach

Figure 20 illustrates the comparison of average response time for round robin and fuzzy logic based algorithms for distributed load balancing, where the respective data are gathered, filtered and compared from (Kun-Ming et al., 2004; Karimi et al., 2009). In distributed load balancing approach, load balancing algorithm is replicated on all the nodes. By comparing fuzzy based algorithm with round robin algorithm, the graph indicates that fuzzy based algorithmic response time is better than the round robin algorithm, because in fuzzy algorithm load distribution takes place in an intelligent manner. It means that based on prior status information, the underloaded nodes accept additional load and vice versa. In contrast, in round robin algorithm the load is distributed without knowing the current status of nodes. Thus, the performance is dependent on the current load status of a node such

that transferring load to an underloaded node will be optimal.

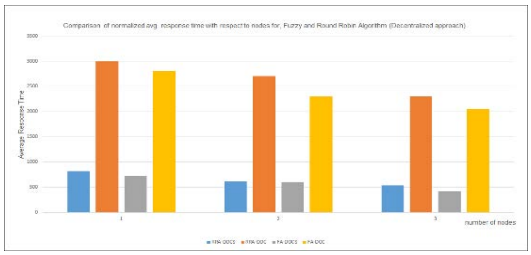

Figure 20. Comparison of avg. response time with respect to number of nodes in distributed approach

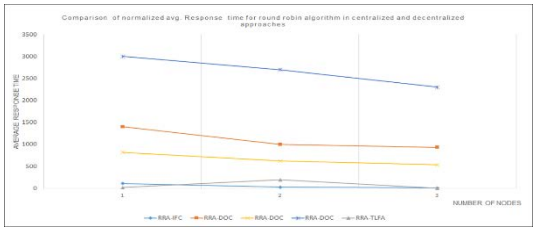

Figure 22. Comparison of avg. response time with respect to number of nodes in centralized and distributed approach

Figure 21 illustrates the comparison of average response time with respect to number of tasks for fuzzy (FA-DLB), randomized (RA-DLB), fuzzy enhanced symmetric algorithm (FESA-DLBA) and randomized algorithms (RA-DLBA), where the respective data are gathered, filtered and compared from (Cheung & Kwok, 2001; Cheung, 2001). In comparison between fuzzy (FA-DLBA) and randomized (RA-DLB) algorithms, the fuzzy-based algorithm performs better, because fuzzy logic based system computes the number of heavily loaded nodes in order to determine the status of each node in a system. Randomized algorithm randomly assigns tasks to nodes irrespective of the status of nodes and causes performance degradation. In conclusion, the fuzzy logic-based algorithm shows better performance than the randomized algorithm.

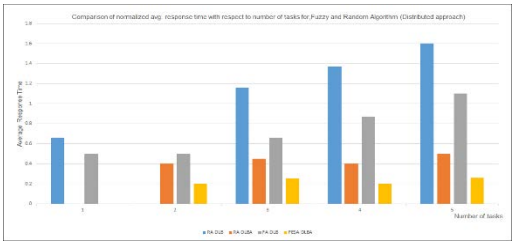

Figure 21. Comparison of avg. response time with respect to number of tasks in distributed approach

The comparative study of variations of normalized response time of round robin algorithm in centralized and distributed approach is illustrated in Figure 22, where the respective data is obtained from (Ahn et al., 2007; Ali et al., 2016; Cheung, 2001; Kun-Ming et al., 2004). In case of round robin algorithm in centralized approach, normalized average response time sharply decreases when the number of nodes increases. In centralized approach, master node is responsible for load distribution by maintaining a cost table containing the current load information of all the nodes. In distributed approach the average response time is dependent not only to increasing the number of nodes but also on the current load status. There is a probability that if there is an increase in the number of nodes but if they are not overloaded, then the average response time will not increase. Conversely, if there is increase in the number of nodes as well as in the load, then the average response time will increase.

The comparative study of variations of normalized response time of fuzzy logic based algorithms in centralized and distributed approach is illustrated in Figure 23, where the respective data are gathered, filtered and compared from (Ahn et al., 2007; Ali et al., 2016; Cheung, 2001; Kun-Ming et al., 2004). In the centralized approach there is an exponential decay in the average response time with respect to increase in the number of nodes, because in the centralized approach only the master node is responsible for load balancing. If a node is overloaded it will only send a request to the master node. The master node will search for an appropriate node for load balancing. Thus, the centralized load balancing reduces communication overhead as well as average response time. In distributed approach load balancing is performed on every available node. If a node is overloaded then it will contact all the nodes for migrating the extra load. The optimal migration depends upon the number of nodes in the system. If a node takes less time to search an appropriate node then the average response time is less because of less communication overhead and vice versa.

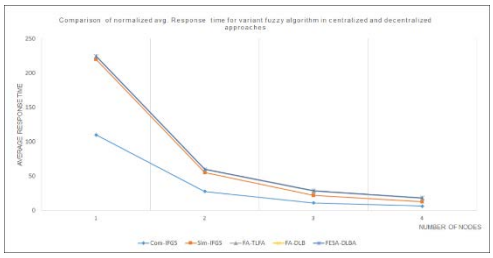

Figure 23. Comparison of avg. response time with respect to number of nodes in centralized and distributed approaches

In Figure 24, the comparative study of variations of average response time of fuzzy logic based algorithms in centralized, distributed and hybrid approaches are illustrated, where the respective data are gathered, filtered and compared from (Alakeel, 2016; Ali et al.,

2016; Cheung, 2001). In comparison to centralized approach, the average response time increases gradually in distributed and hybrid approaches. In the hybrid approach, load balancing is performed in centralized manner but the master node is selected for a fixed period of time. In order to perform further load balancing, another node is selected as master for some time period and so on. However, selecting and reselecting a master node results in increasing average response time. The bottleneck of single point of failure in centralized approach results in higher average response time.

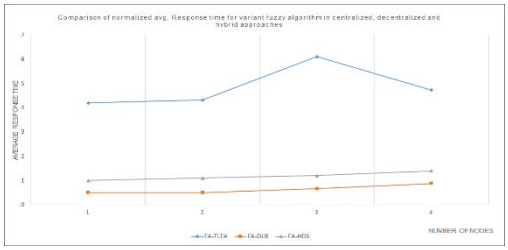

Figure 24. Comparison of avg. response time with respect to number of nodes in centralized, distributed and hybrid approaches

The comparative study of variations of normalized response time (ΔTna) with respect to Distributed Round Robin (DRR) and Centralized Dynamic Algorithm (CDA) is illustrated in Figure 25, where the respective data are gathered, filtered and compared from (Karimi et al., 2009; Barazandeh et al., 2009). In centralized approach employing dynamic biasing, the normalized average response time per node decreases rapidly with the increase in number of nodes. When the average response time per node is 51.93ms, the normalized average response time is on peak. With the increase in the number of nodes in CDA the average response time per node is 19.01ms, resulting in decrease in normalized average response time to approximately 35ms. Due to further increase in the number of nodes in CDA, the average response time per node becomes 1.96ms and, the normalized average response time is decreased to approximately 10ms. Thus, the decrease in the average response time per node results in steep decrease in the normalized average response time. In distributed approach employing DRR, the variation of average response time per node (normalized) is aperiodic in nature with undershoot as well as overshoot with the increase in number of nodes. When the average response time per node is 1.5ms, the normalized average response time is on the peak. When the number of nodes is increased, the average response time per node is decreased to 1.25ms and, the normalized cumulative average response time is decreased to 36.41ms. A further increase in the number of nodes results in the increase in average response time per node to approximately 1.66ms and the aggregated normalized response time is decreased to 17.38ms. The overshoot and undershoot in the average response time per node is due to the distributed approach. When the average response time per node is decreased from 1.5ms to 1.25ms, at that time the nodes are not overloaded and the requests are processed swiftly, which results in decrease in average response time. When the number of nodes is further increased the average response time per node is also increased to 1.66ms. It is because at that time the nodes are overloaded and the processing of requests are delayed. The variation of normalized average response time is highly influenced by centralized approach (CDA), because in centralized approach only one node is responsible for managing load of the nodes in a system.

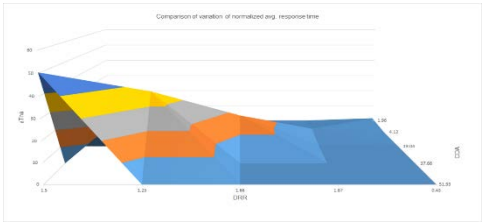

Figure 25. Surface map of normalized ΔTna with respect to DRR and CDA

Figure 26 illustrates a comparative analysis of average response time per node in Distributed Fuzzy Algorithm (DFA) and Centralized Fuzzy Algorithm (CFA) with respect to normalized average response time (ΔTna), where the respective data are gathered, filtered and compared from (Cheung, 2001; Barazandeh et al., 2009). In centralized approach employing CFA, the average response time per node (normalized) decreases rapidly with the increase in number of nodes. When the average response time per node is 52ms, the normalized average response time is high. With the increase in the number of nodes in CFA, the average response time per node becomes 37ms resulting in a decrease in normalized average response time to approximately 36ms. Further increase in the number of nodes in CFA results in the average response time per node reaching 18.57ms and, the normalized average response time is decreased to 17ms. Thus, the decrease in the average response time per node results in steep decrease in the normalized average response time, because centralized load balancing approach employs fuzzy algorithm, which realizes intelligent load balancing mechanism. In centralized approach, the master node has status information of all the nodes and using fuzzy algorithm it intelligently selects the appropriate nodes for transferring the load. In distributed approach employing DFA, the average response time per node (normalized) is appearing to be aperiodic in nature with increase in number of nodes. When the average

response time per node is approximately 0.1ms, the normalized average response time is at maximum. Due to further increase in the number of nodes, the average response time per node is also increased to 0.75ms monotonically. Continuing increase in the number of nodes means continuous decrease in average response time per node to 0.66ms and the aggregated (cumulative) normalized response time is decreased to 17.91ms. Further increase in the number of nodes results in increase of the average response time per node up to 1.1ms. Following the above statistics it can be stated that the average response time per node is not directly dependent on the number of nodes, instead it is highly dependent on the increasing communication overhead due to increasing node count in the system.

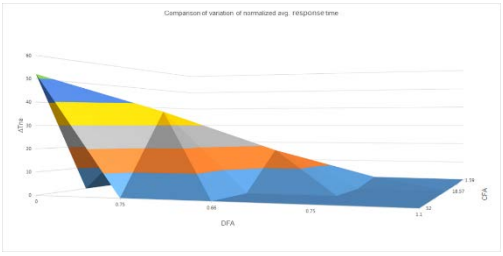

Figure 26. Surface map of normalized ΔTna with respect to DFA and CFA

In Figure 27, the comparative study of variations of normalized average response time (ΔTna) with respect to Centralized Random Algorithm (CRA) and Distributed Random Algorithm (DRA) is illustrated, where the respective data are gathered, filtered and compared from (Kwork et al., 2004; Saxena et al., 2012; Cheung, 2001). In centralized approach which employs Random algorithm (CRA), the average response time per node decreases rapidly when the number of nodes is increased. When the average response time per node is 46ms, the normalized average response time is at maximum. With the increase in the number of nodes in CRA, the average response time per node is 26.66ms resulting in a steep decrease in normalized aggregated average response time to approximately 24.66ms. When the number of nodes is increased, the average response time per node is decreased to 15.23ms resulting in an exponential decrease in normalized cumulative response time to 12.4ms. Due to further increase in the number of nodes in CRA results in exponential decrease in the average response time per node up to 4.07ms. The normalized average response time is decreased to 0.07ms, approximately. Thus, decrease in the average response time per node results in steep decrease in the normalized average response time in centralized approach. In distributed approach employing DRA, the average response time per node gradually increases with the increase in number of nodes. When the average response time per node is 2ms, the normalized average response time is high. When the number of nodes is increased the average response time also increased to 2.83ms, and the normalized aggregated average response time steeply decreases to 12.4ms. A further increase in number of nodes results in average response per node to become 4ms and, the normalized average response time decreases to 0.07ms, approximately. From the above analysis it is clear that even if the algorithm is same in centralized and distributed approaches, the average response time will have continuous variations. It means that when the randomized algorithm is analyzed in centralized approach the average response time decreases with the increase in number of nodes. However, when the randomized algorithm is analyzed in the distributed approach the average response time increases with the increase in number of nodes. The variation of normalized average response time is highly influenced by CRA, because in CRA approach centralized load balancing is employed.

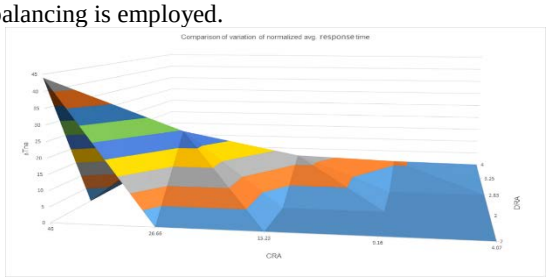

Figure 27. Surface map of normalized ΔTna with respect to CRA and DRA

In Figure 28, the comparative study of variations of normalized average response time with respect to Distributed Random Algorithm (DRA), Distributed Round Robin Algorithm (DRR) and Distributed Fuzzy Biasing Algorithm (DFA) is illustrated, where the respective data are gathered, filtered and compared from (Cheung, 2001; Kun-Ming et all., 2004). The figure illustrates that, average response time of round robin algorithm is gradually increasing and the surface is not smooth. In DRR initially the average response time per node is 1.5ms, however due to further increase in the number of nodes, the average response time per node is decreased to 1.25ms. In the DRR, the average response time per node increases to 1.66ms and 1.87ms with respect to increase in the number of nodes. A further increase in the number of nodes results in decrease in the average response time per node to 0.43ms. The performance of DFA algorithm in distributed approach illustrates that, the average response time per node follows a gradual and smooth increase with respect to number of nodes. Initially the average response time per node is 2ms however, when the number of nodes is increased the average response time per node is also increased to 2.83ms. Continuation of the increase in the number of nodes

results in increase in the average response time per node up to 3.25ms and 4.00ms, respectively. In case of DRA, initially the average response time is 0.1ms, approximately. The average response time per node is increased to 0.75ms with the increase in number of nodes. Further increase in the number of nodes results in decrease in average response time to 0.66ms. From the above analysis it can be concluded that the average response time per node is indirectly proportional to increasing the number of nodes in distributed approach. The average response time per node can be increased or decreased due to communication overhead, network latency and heterogeneity of the system. The average response time is sensitive to joining and leaving of nodes and, unreliable network links.

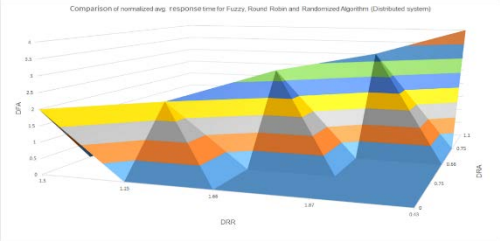

Figure 28. Surface map of Normalized DRA, DRR and DFA in Decentralized System

The comparative study of variations of average response time with respect to fuzzy algorithm (HFA) in hybrid approach, fuzzy algorithm (CFA) in centralized approach and fuzzy algorithm (DFA) in distributed approach is illustrated in Figure 29, where the respective data are gathered, filtered and compared from (Alakeel, 2016; Ahn et al., 2007; Cheung, 2001). Analysis of fuzzy-based algorithm in centralized approach illustrates that, the average response time is decreasing very rapidly with the increase in a number of nodes. Figure 29 illustrates that, the average response time per node is 52ms, but with the increase in number of nodes there is a steep decrease in the average response time per node from 52ms to 1.59ms. In distribute approach employing the fuzzy-based algorithm the average response time is aperiodic with respect to increase in the number of nodes. Initially the average response time per node is 0.1ms approximately however, it is increased to 0.75ms with respect to increase in the number of nodes. Continuing with an increase in the number of nodes, the average response time per node is decreased to 0.66ms. However, due to further increase in the number of nodes the average response time also increases. Aperiodic nature of increase and decrease in the average response time per node shows that in distributed approach performance of fuzzy is not directly dependent on the number of nodes. In the hybrid approach, the average response time per node is gradually increased with respect to the number of nodes. In Figure 29 it can be observed that, initially the average response time per node is 1ms, however it is increased to 1.1ms and 1.3ms with the increase in number of nodes, respectively. This gradual increase in average response time per node continues with the increase in number of nodes. The gradual increase in the average response time per node in hybrid approach is due to the selection and reselection of master node, disconnection of master node and randomized network latency.

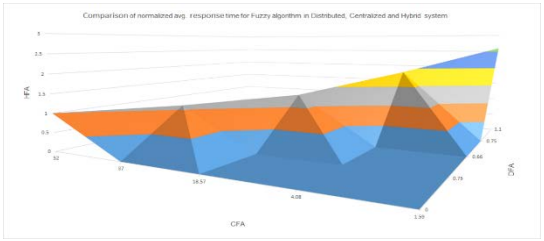

Figure 29. Surface map of normalized CFA, DFA and HFA in Centralized, Distributed and Hybrid systems

6 IMPLEMENTATION DIRECTION

THIS section illustrates implementation and deployment environment for further validation of proposed architecture. The proposed architecture is intended to be implemented in heterogeneous operating systems environment and, Java programming language is used to deploy mobile agent framework i.e. Java Development Framework (JADE). JADE is selected because it is Java-based platform providing a simple, portable and efficient Java API. Agent containers in JADE are distributed among all the nodes in the network. The configurations of our mobile agent framework development architecture and runtime environment specification are illustrated in Table 3. The proposed mobile agent monitoring algorithm is developed using Java eclipse IDE agent platform for monitoring the distributed system. In addition, to test the proposed monitoring algorithm, we have used two additional load generators software which are, (a) CPU Stress and, (b) Heavy Load. The purpose of using load generator software is to generate resource load (CPU and RAM) on our target nodes while monitoring the variations by mobile agents under different load scenarios. In terms of connectivity, one of the nodes is wirelessly connected and remaining nodes are wired connected with monitoring node. The wired connections operate at 100Mbps (maximum) and wireless network bandwidth is 100Mbps on average.

Table 3. Platform specifications of runtime environment and system configuration

Nodes	Specification	Runtime Environment	
		Operating System	Software
Node 1	Intel Celeron G1840 CPU 2.80 GHz RAM: 4GB, HDD: 128 GB NIC: Wireless Adaptor	Windows 10	
Node 2	Intel Core i7-6700 CPU 3.40 GHz, RAM: 8 GB, HDD: 2 TB, NIC: Realtek PCIe GBE Family Controller	Linux darnel 2.6 Fedora	
Node 3	Inter Core i5 3.1 GHz, RAM: 3 GB, HDD: 500 GB, NIC: Realtek PCIe GBE Family Controller	Windows 8	Eclipse 4.6, JADE 4.5.0, JDK 1.8, CPU Stress and Heavy Load.
Node 4	Intel Core 2 Duo E8400 CPU 3.00 GHz, RAM: 3 GB, HDD: 320 GB, NIC: Realtek PCIe GBE Family Controller	Windows 7	
Monitoring Node	Intel Core i7-6700 CPU 3.40 GHz, RAM: 8 GB, HDD: 2 TB, NIC: Realtek PCIe GBE Family Controller	Windows 10	
Network	Ethernet: 100Mbps LAN, Wireless: 100Mbps WAP, Signal strength: 45% (average)	0.83	

7 CONCLUSION

THE purpose of optimal utilization of resources in distributed systems is to minimize computing time and maximize the overall performance. The most frequently used load balancing algorithms of distributed systems are classified according to different system architectures and designing of respective algorithms. In this paper, we have focused on the fuzzy logic based load balancing approaches for load distribution as well as agent-based load monitoring mechanisms. We have put forward a study of current fuzzy based load balancing approaches. In order to highlight and motivate for fuzzy load balancing, a detailed comparative analysis of different load balancing algorithms based on various parameters is carried out. In this paper, we have presented the formulation of smooth composite fuzzy membership function aiming to realize fine grained load estimation. The related fuzzy rule-base is designed. The corresponding system architecture is presented, which is based on hybrid approach for load balancing and load monitoring combining fuzzy logic controller and mobile agents. The proposed architecture hybridizes autonomous mobile agents and fuzzy controller into a single platform, which is distributed over a set of nodes. The evaluations through simulation and implementation directions are presented.

8 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

9 REFERENCES

M. A. Alakeel. (2012). A fuzzy dynamic load balancing algorithm for homogenous distributed systems, World Academy of Science, Engineering and Technology, WASET, 6, 7-10.

M. A. Alakeel. (2016). Application of fuzzy logic in load balancing of homogenous distributed systems, International Journal of Computer Science and Security, volume, IJCSS, 10, 95-106.

H. C. Ahn, Youn. H. Y., Jeon. K. Y., and Lee K. S. (2007). Dynamic load balancing for large distributed system with intelligent fuzzy controller, International Conference on Information Reuse and Integration, IEEE, 576-581.

M. M. Ali and Derakhshi M. R. F. (2016). Two level fuzzy approach for dynamic load balancing in the cloud computing, Journal of Electronic System, DLINE, 6, 17-31.

I. Barazandeh and Mortazavi. S. S. (2009). Two Hierarchical Dynamic Load Balancing Algorithms in Distributed Systems, International Conference on Computer and Electrical Engineering, IEEE, 516-521.

I. Barazandeh, Mortazavi S.S., and Rahmani M.A. (2009). Intelligent fuzzy based biasing load balancing algorithm in distributed systems, 9th IEEE Malaysia International Conference on Communications, Malaysia, IEEE, 713-718.

J. Brandt, Gentile A., Mayo J., and Pebay P. (2009). Resource monitoring and management with OVIS to enable HPC in cloud computing environments, International Symposium on Parallel & Distributed Processing, IEEE, 1-8.

L. S. Cheung and Kwok. Y. K. (2001). A Quantitative comparison of load balancing approaches in distributed object computing systems, Computer Software and Applications Conference, COMPSAC 25th Annual International, IEEE, 257-262.

R. M. Emami, Turksen B. I., and Goldenberg A. A. (1998). Development of a systematic methodology of fuzzy logic modeling, IEEE Transactions on Fuzzy System, IEEE, 6, 346-360.

S. Franklin and Graesser. A. (1996). Is it an agent, or just a Program? A taxonomy for autonomous agents, Third International Workshop on Agents Theories, Architecture, and Languages, Springer Berlin Heidelberg, 21-35.

Y. Fu, Li, H., Jiang, Z. and Wang, S., (2009). Double layers fuzzy logic based mobile robot path planning in unknown environment. *Intelligent Automation & Soft Computing*, *15*(2), 275-288.

W. Funika, Szura F., and Kitowski. (2011). Agent-Based monitoring using fuzzy logic and rules, Computer Science Annual of AGH-UST, 12, 103-113.

J. M. Garibaldi and John, R.I. (2003). May. Choosing membership functions of linguistic terms. In *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, 1, 578-583.

J. Giarratano. (1989). Expert Systems: Principles and Programming, PWSKENT Publishing Company, ISBN-10: 0534384471.

D. Horvat, Cvetkovic. D., Milutinovic.V., Kocovic. P., and Kovacevic. V. (2000). Mobile Agents and Java Mobile Agents Toolkits, International Conference on System Sciences, IEEE, 1-10.

N. I. Ivanisenko and Radivilova .A.T. (2015). Survey of major load balancing algorithms in distributed system, Information Technologies in Innovation Business (ITIB), IEEE, 89 - 92.

J. Jeong and Oh, S.Y, (2004). Automatic Rule Generation of Fuzzy Logic Controllers based on Asynchronous Coevolution of Rule-Level Subpopulations. *Intelligent Automation & Soft Computing*, *10*(3), 195-207.

Y. Jiang, Sun H., Ding J., and Liu Y. (2015). A data transmission method for resource monitoring under cloud computing environment, International Journal of Grid Distribution Computing, IJGDC, 8, 15-24.

C. M. John and V. Teagu. (2003). Autonomous nodes and distribute mechanisms, In Software Security-Theories and Sstems, Springer-Verlag, 58-83.

A. Karimi Zrafshan F., Jantan B.A., Ramli A.R., and Saripan I. (2009). A New fuzzy approach for dynamic load balancing algorithm, International Journal of Computer Science and Information Security, IJCSIS, 6, 1-5.

R. Khan, Haroon. M., and Husain. M. S. (2015). Different technique of load balancing in distributed system: A Review Paper, Conference on Communication Technologies, IEEE, 371-375.

V. Kun-Ming, Chou. Y., and Wang. Y. (2004). A fuzzy-based dynamic load-balancing algorithm, Journal of Information, Technology and Society, Springer, 4, 55-63.

S. Kwon and Choi J. (2006). An agent based adaptive monitoring system, Pacific Rim International Workshop on Multi-Agents, Springer-Verlag, 672-677.

Y. K. Kwork and Cheung L. S. (2004). A new fuzzy-decision based load balancing system for distributed object computing, Journal of Parallel and Distributed Computing, ELSEVIER, 64, 238-253.

Q. Long, Lin. J., and Sun. Z. (2011). Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations, In Simulation Modelling Practice and Theory, ELSEVIER, 19, 1021- 1034.

S. A. Moosavi Nejad, Mortazavi. S.S., Vahdat.B.V. (2011). Fuzzy based design and tuning of distributed systems load balancing controller, 5th Symposium on Advances in Science And Technology, Iran, Mashhad, SASTech.

S. Naaz, Alam. A and Biswas. R. (2010). Implementation of a new Fuzzy Based Load Balancing Algorithm for Hypercubes, International Journal of Computer Science and Information Security, IJCSIS, 8, 270-274.

H. S. Nwana. (1996). Software Agents: An Overview, The knowledge engineering review, 11, 205-244.

S. Rajani and Garg N. (2015). A clustered approach for load balancing in distributed systems, International Journal of Mobile Computing & Application, SSRG-IJMCA, 2, 2393-9141.

F. Sabahi and Akbarzadeh M., (2016). Extended fuzzy logic: Sets and systems, IEEE Transactions on Fuzzy Systems, IEEE, 24, 530 - 543.

S. Saxena, Khan. M. Z., and Sing. R. (2012). Performance analysis in distributed system of dynamic load balancing using fuzzy logic, Engineering and Technology (S-CET), IEEE, 1-5.

S. Seth, Sahu A., Jena K. S. , (2012). Efficient load balancing in cloud computing using fuzzy logic, IOSR Journal of Engineering, IOSRJEN, 2, 65-71, ISSN: 2250-3021.

M. Wooldridge and Jennings. N. R. (1995). Intelligent Agents: Theory and Practice, The Knowledge Engineering Review, 10, 115-152.

10 NOTES ON CONTRIBUTORS

Moazam Ali obtained his BCS in Computer Science in year 2007 from the Islamia College, University of Peshawar and, MS-IT in Computer Networks in year 2012 from the Institute of Management Sciences, Peshawar. Currently, he is pursuing his PhD in Distributed Systems in the Department of Aerospace and Software Engineering (Informatics), Gyeongsang National University, Jinju, South Korea.

Susmit Bagchi has received B.Sc. (Honours) in 1993 from Calcutta University, B.E. (Electronics Engineering) from Nagpur University in 1997, M.E. (Electronics and Telecom Engineering) from Bengal Engineering and Science University (presently IIEST)

in 1999. He obtained Ph.D. (Engineering) from IIEST in 2008 in Information Technology. Currently, he is Associate Professor in Department of Aerospace and Software Engineering, Gyeongsang National University. His research interests are in Distributed Computing and Systems.