# Performance improvement of distributed system through load balancing and task scheduling using fuzzy logic

Abhijit A. Rajguru* and Sulabha S. Apte

WIT,
Solapur, Maharashtra, India
Email: abhijit25012@gmail.com
Email: aptesulabha@gmail.com
*Corresponding author

**Abstract:** Task scheduling and load balancing in distributed network are the most challenging research area in computer science. In distributed systems, scheduling mechanism has more issues as there is no centralised authority to allocate the workload among multiple processors. Further, handling both load balancing and scheduling at hand is a daunting task. In this paper, we propose a fuzzy-based load balancing and task scheduling technique to optimise the performance of distributed system. Initially, clusters are formed and node with larger buffer availability and high CPU speed is elected as cluster head. Tasks are prioritised into flexible and non-flexible using task prioritising strategy. Non-flexible tasks are prioritised over flexible tasks. For non-flexible tasks, essential information of nodes such as CPU speed, work load and distance from cluster head are made pass through the fuzzy system. Node state is obtained as output. Based on states of node, non-flexible tasks are allocated. Our technique dynamically handles scheduling and load balancing at hand. We use simulation results to prove efficiency of our technique.

**Keywords:** distributed computing; fuzzy logic; load balancing; task scheduling.

**Reference** to this paper should be made as follows: Rajguru, A.A. and Apte, S.S. (2019) 'Performance improvement of distributed system through load balancing and task scheduling using fuzzy logic', *Int. J. Engineering Systems Modelling and Simulation*, Vol. 11, No. 1, pp.35–42.

**Biographical notes:** Abhijit A. Rajguru received his BE and MTech in Computer Science and Engineering from the Shivaji University, Kolhapur, Maharashtra, India in 2007 and 2009, respectively. He is a Research Scholar at the Walchand Institute of Technology, Solapur (Solapur University, Maharashtra, India). His research interest includes distributed system, cloud computing, grid computing, and load balancing.

Sulabha S. Apte received her PhD in Computer Engineering. She is having 33 years teaching experience and two years industry experience. She is working as a Professor in the CSE Department in the WIT, Solapur. Her research interest includes computer architecture, distributed system, and image processing.

## 1 Introduction

### *1.1 Distributed systems*

A distributed system consists of set of communication and computing resources, which are shared by active users (Grosu et al., 2001). In other words, a large loosely coupled distributed system is formed by connecting a group of work stations through the communication channel (Bharadwaj et al., 2003). Resource sharing is the foremost advantage of distributed system. Reliability, scalability, economy, inherent distribution and functional separation are the features of distributed system (Nadiminti et al., 2006).

On account of time accuracy and other features of distributed system, it is mainly useful in telecommunication networks such as telephone networks and computer networks (internet). Distributed system has many application like aircraft control system, industrial control systems, multiplayer online games and virtual reality, etc. (http://www.wikipedia.org).

### *1.2 Load balancing*

In parallel and distributed systems, multiple programs are processed parallelly by multiple processors. The quantity of time assigned to a processor to execute task is termed as work load (WL) of a processor (Sharma et al., 2008). The process of equalising loads among the processors is known as load balancing. This technique attains good throughput without the requirements of additional hardware (Kameda et al., 2000).

Load balancing can be achieved statically (at the beginning) or dynamically (run time). Based on this, it is

classified into two types as static load balancing and dynamic load balancing (Grosu and Chronopoulos, 2005).

### 1.2.1 Static load-balancing

Static load balancing balances the load of system using priori knowledge of applications and statistical knowledge about the system. The entire process is performed by considering average behaviour of system.

### 1.2.2 Dynamic load-balancing

In dynamic load balancing technique, loads are allocated to work stations dynamically to balance the WL. This process is continued until processes are completed or terminated. New jobs may be added to the cluster of machines at any time by the user and are scheduled by load balancing system (Chow and Kwok, 2002).

The problem of load balancing becomes significant as demand for computing power increases. The main objective of load balancing is to improve the distributed system performance by means of allocating loads appropriately (Grosu and Chronopoulos, 2005). In parallel and distributed systems, load balancing is considered to be a critical issue for it has to assure fast processing and good utilisation (Guo and Bhuyan, 2006).

## 1.3 Scheduling in distributed systems

Apart from load balancing, task scheduling is also an important technique to improve the performance and throughput of distributed network (Nikravan and Kashani, 2007). The process of distributing tasks among processors is defined as scheduling. Task scheduling is differentiated into local scheduling and global scheduling (Sharma et al., 2008).

### 1.3.1 Local scheduling

This technique allocates the processes to the time slices of the processor.

### 1.3.2 Global scheduling

Global scheduling makes a decision of place to execute a process in the multiprocessor system. Here, the scheduling is accomplished either by a central/master device or may be distributed among the processing elements. Global scheduling is further divided into two types as static scheduling and dynamic scheduling (Sharma et al., 2008).

In static scheduling processes are assigned to processors before execution starts. On other hand dynamic scheduling can reassign the processes to the processor during execution (Jing et al., 2008).

## 1.4 Issues of load balancing and scheduling

The load balancing and scheduling mechanism in distributed systems has various issues as there is no centralised authority to allocate the workload among multiple processors. Some of the issues are described below:

1 Achieving load balancing in the network is difficult, as processes move from a node to another amidst execution of WL (Sharma et al., 2008).

2 The diverse nature of both huge users and computing resources present more complicated challenges in scheduling and processing resources (Viswanathan et al., 2007).

3 An efficient load balancing scheme should support characteristics such as stable, scalable and low overhead. However, these characteristics are interdependent in nature (Grosu and Chronopoulos, 2005).

4 Handling both scheduling and load balancing at hand is a daunting task (Nikravan and Kashani, 2007). Further, the main attributes of scheduling algorithm namely fairness and locality often conflict each other (Isard et al., 2009).

## 1.5 Problem identification

Till date, the literature has enormous work for load balancing and task scheduling in distributed system. But they did not concentrate on the task requests which may be flexible during task allocation.

Resource aware distributed scheduling strategies for distributed systems are described in Viswanathan et al. (2007). In this work, the major consideration is provided to sink nodes and not to source nodes. Also, the sink nodes are selected based on buffer information alone. But there are other factors which influence the task scheduling like distance between each cluster members, workload and deadline of each task, etc. They do not use any specific mechanism to elect the coordinator node [cluster head (CH)].

To overcome the above described problems, we propose to develop a fuzzy-based load balancing and task scheduling technique (FLBTS) for distributed system which provides more accuracy and efficiency.

# 2 Related work

Grosu et al. (2001) have presented a game theoretic framework for obtaining a fair load balancing scheme. Their main goal was to derive a fair and optimal allocation scheme. They have formulated the load balancing problem in single class job distributed systems as a cooperative game among computers and it is also a fair solution. In Grosu and Chronopoulos (2005) the same authors have used a game theoretic frame work for obtaining a fair balancing scheme. In this case they have formulated the load balancing problem in heterogeneous distributed systems as a non-cooperative game among users. In both cases they have showed that the Nash bargaining solution (NBS) of this

game provides a Pareto optimal operation point for the distributed system.

Nikravan and Kashani (2007) have presented a new method for process scheduling in distributed systems. The scheduling in distributed systems is known as an NP-complete problem even in the best conditions, and methods based on heuristic search have been proposed to obtain optimal and suboptimal solutions. In this paper, they have used the power of genetic algorithms for balancing the load efficiently.

Viswanathan et al. (2007) have proposed distributed algorithms referred to as resource-aware dynamic incremental scheduling (RADIS) strategies. These strategies are specifically designed to handle large volumes of computationally intensive arbitrarily divisible loads submitted for processing at cluster/grid systems involving multiple sources and sinks (processing nodes). The design of these strategies adopts the divisible load paradigm, referred to as the divisible load theory (DLT).

Isard et al. (2009) have introduced a powerful and flexible new framework for scheduling concurrent distributed jobs with fine-grain resource sharing. In that, the scheduling problem is mapped to a graph data structure, where edge weights and capacities encode the competing demands of data locality, fairness, and starvation-freedom, and a standard solver computes the optimal online schedule according to a global cost model.

Lin et al. (2007) have addressed the problem of providing deterministic QoS to arbitrarily divisible applications executing in a cluster. Four contributions are made. First, they have extent DLT to compute the minimum number of processors required to meet an application deadline. Second, based on this, a novel algorithmic approach integrating DLT and EDF scheduling has been proposed. Third, important design parameters are identified that affect the performance of real-time divisible-load scheduling algorithms. Finally, they systematically investigated the effects of these design parameters on a set of real-time scheduling algorithms.

Aida and Casanova (2009) have investigated the scheduling of mixed-parallel applications, which exhibit task and data parallelism, in advance reservations settings. They have defined two scheduling problems, RESSCHED and RESSCHEDDL, depending on whether the goal is to minimise application turn around- time or to meet a deadline, respectively. For each problem, they have proposed and evaluated a set of scheduling algorithms. These algorithms were compared for both their performance (i.e., turn-around-time and tightest deadline met) and their resource consumption.

Basu et al. (2011) have proposed a scheduling mechanism for distributed system. Their proposed method is based on performing model checking for knowledge properties. It allows identifying where the local information of a process is sufficient to schedule the execution of a high priority transition. As a result of the model checking, the program is transformed to react upon the knowledge it has at each point. The transformed version has no priorities, and uses the gathered information and its knowledge to limit the enableness of transitions so that it matches or approximates the original specification of priorities.

## 3 Fuzzy-based load balancing and task scheduling technique

### *3.1 Overview*

In this paper, we propose a FLBTS in distributed systems. Our proposed network contains a set of client (C) and server (S) nodes. These nodes form clusters and node with larger buffer and high CPU speed is elected as CH. The elected CH is responsible for coordinating and scheduling tasks. Tasks are prioritised using a Boolean value. Flexible tasks are represented by Boolean value 1 and value 0 denotes non-flexible tasks. When C nodes receive a new task, it forwards the task details to CH. The CH looks for flexible field, for non-flexible tasks, it collects information such as WL, CPU speed and distance from CH. These values are made pass through the fuzzy system. Fuzzification is performed by representing input and output variables in membership functions. Fuzzy rules are evaluated and as an outcome of fuzzy system, S' nodes are classified into three states as, schedulable (SC), likely to be schedulable (LS) and not schedulable (NS). Non-flexible tasks are allocated to S' nodes in SC state and when buffer requirement exceeds the buffer availability of SC state nodes, then they are allocated to S' nodes in LS state. The flexible tasks are allocated to S' nodes in likely to be scheduled (LS) state.

### *3.2 Computation of metrics*

#### *3.2.1 Available buffer estimation*

The buffer availability of node '$n$' at time '$t + 1$' can be calculated as follows,

$$AvaB_{ni}^{t+1} = \left( \frac{\sum_{n=0}^{I-1} \left( (I-n) \cdot AvaB_{ni}^{t-n} \right)}{\sum_{n=0}^{I-1} n} \right) \bullet p \tag{1}$$

where $I$ denote iteration number, $n$ stands for node in the network, $AvaB_{ni}^{t-n}$ is the available buffer computed previously at $t - x$ and $p$ is buffer estimating probability, usually set to 0.95.

#### *3.2.2 WL calculation*

WL at a node is calculated as follows,

$$W_{Li} = N_T \times S_T \tag{2}$$

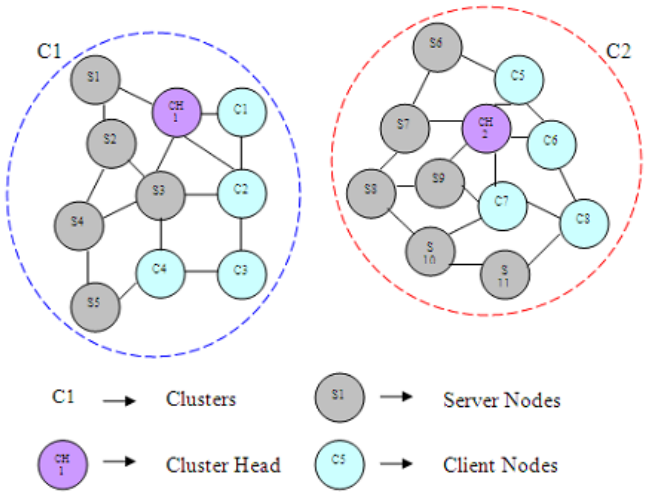
Here, $N_T$ is the number of tasks to be computed and $S_T$ size of tasks.

### *3.3 Network architecture*

Consider the distributed system comprises of set of client nodes (C's) and server nodes (S's). Client nodes (C's) are

nodes that have tasks to be processed by server nodes (S's). Both client and server nodes are formed clusters. Each cluster has a CH, it is responsible for coordinating and scheduling tasks. The network architecture is given below in Figure 1.

**Figure 1** Network architecture (see online version for colours)




### 3.3.1 CH selection

In each cluster, a CH is elected considering larger buffer size and high CPU speed. The step by step process of CH selection is given below:

1 After the deployment of nodes in the network, both C's and S's form clusters.

2 Within the cluster, each node estimated available buffer value ( $AvaB_{ni}^{t-n}$ ) using equation (1).

3 Each node broadcasts network information (Nw info) message in the cluster. Nw info message includes $AvaB_{ni}^{t-n}$ value and CPU speed.

$$Node \xrightarrow{NwInfo} Cluster$$

4 Each node receives the broadcasted value and finally a node with larger buffer size ( $AvaB_{ni}^{t-n}$ ) and high CPU speed is selected as CH.

5 The selected CH broadcasts this selection information to all nodes in the cluster.

$$CH \xrightarrow{CHselecion} Nodes\ in\ the\ CH$$

The selected CH is responsible for controlling, coordinating and scheduling the tasks among C's and S's nodes.

### 3.4 Task prioritisation strategy

The C's nodes contain tasks to be computed and these tasks are distributed among S's nodes by the CH. In real time situations, the task contains stringent deadline requirements, which should be satisfied to avoid unsolicited packet losses. To bring this consideration in our technique we use task prioritisation strategy. By this, each task is marked and prioritised using Boolean value.

In general, each task consists of unique task id, task size and deadline (execution time) fields. In addition to these fields, an extra Boolean field is added to specify the priority of tasks. The Boolean value 1 represents flexible and 0 represents non-flexible. We set priority to non-flexible task over flexible. For flexible task, we can extend its execution time (dead line) when its deadline is missed. But, in the case of non- flexible tasks, deadline should not be missed and the tasks have to be processed immediately. The header of task is shown in Table 1.

**Table 1** Fields in task header

| *Task id* | *Task size* | *Deadline* | *Flexible/non-flexible* |
|---|---|---|---|

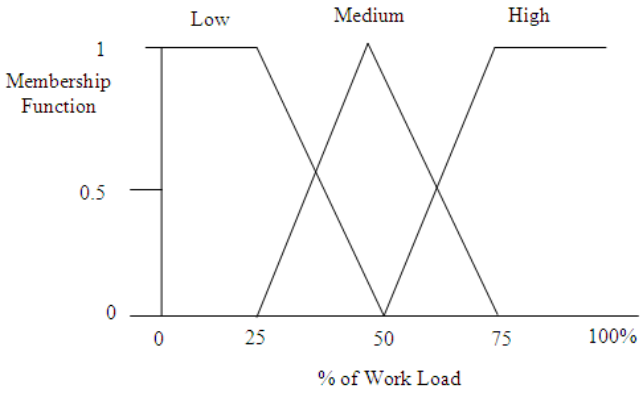
### 3.5 Fuzzy-based task scheduling technique

When C's have new task to be processed, it sends task details to the CH. While receiving task details, the CH looks for flexible field of task. If the value is zero then CH gathers WL, CPU speed and distance from the CH information from S's nodes of its cluster. WL of node is estimated as per equation (2). This information's of each server node (S) are made pass through the fuzzy system to predict the states of S nodes. As an outcome of fuzzy system, a server node (S) would be in any of the following three states as SC, LS and NS.

### 3.5.1 Fuzzification

In this phase, the input variables WL, CPU speed and distance from the scheduler are fuzzified by representing in membership functions. The membership functions for input variables WL, CPU speed and distance from the scheduler are given in Figures 2, 3 and 4, respectively. Figure 5 represents the membership function of output variable.

**Figure 2** Fuzzy set for WL




### *Fuzzy inference engine*

In fuzzy logic system, the fuzzy rules make up the fuzzy inference system. The defined fuzzy rules associates input, output parameters with membership functions. Fuzzy rules

take if then else rule format. In our system, fuzzy inference system is constructed considering the following 13 rules in Table 2.

**Figure 3** Fuzzy set for CPU speed

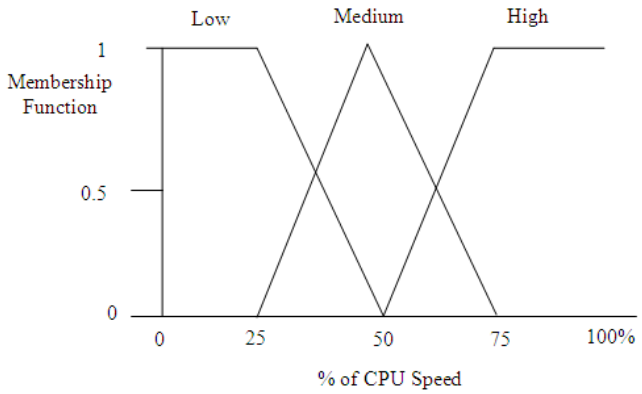



**Figure 4** Fuzzy set for distance

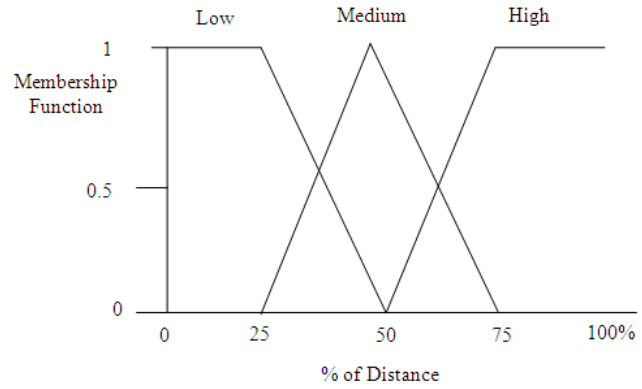
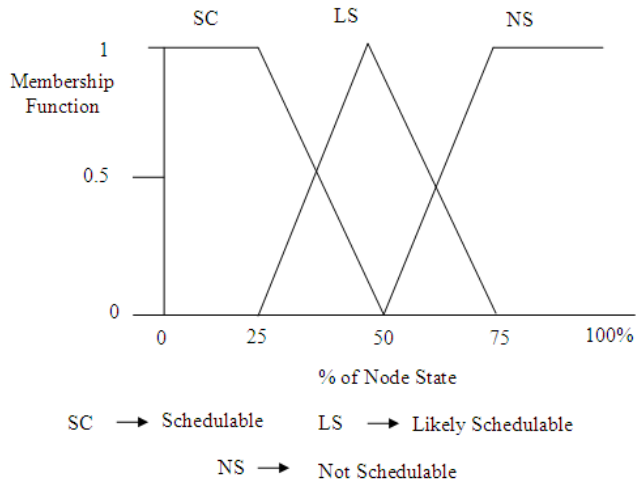



**Figure 5** Fuzzy set for node state




In Table 2, SC denotes schedulable state, NS stands for not schedulable state and LS represents the state of likely to be scheduled. Among 13 rules, we intend to explain the following two cases,

1 If (Work Load = Low && CPU Speed = High && Distance = Low) Then

*"Node state is schedulable"*

*End if*

2 If (Work Load = High && CPU Speed = Low && Distance = High) Then

*"Node state is not schedulable"*

*End if*

**Table 2** Fuzzy rule set

| *Rule* | *Work load* | *CPU speed* | *Distance* | *Node state* |
|---|---|---|---|---|
| 1 | Low | Low | Low | LS |
| 2 | Low | High | Low | SC |
| 3 | Low | High | Medium | LS |
| 4 | Low | Low | High | NS |
| 5 | Low | High | High | LS |
| 6 | Medium | High | Low | SC |
| 7 | Medium | High | Medium | LS |
| 8 | Medium | Low | Medium | NS |
| 9 | Medium | Low | High | NS |
| 10 | High | Medium | Low | NS |
| 11 | High | Medium | Medium | NS |
| 12 | High | High | Low | LS |
| 13 | High | Low | High | NS |

### *3.5.2 Defuzzification*

A precise output value from fuzzy set is extracted during defuzzification phase. We have enormous methods for an efficient defuzzification. From those we choose to utilise the weighted mean method. The formulation for defuzzification process is,

$$O^* = \sum \eta_o(\overline{O}) \Big/ \sum \eta_O(\overline{O}) \tag{3}$$

where $O^*$ is the derived output value, $\eta_O(\overline{O})$ denotes the strength of output membership function and $\overline{O}$ is the centroid of membership function.

### *3.5.3 Task allocation*

As a result of fuzzy logic, server nodes (S's) will be in the following three states as SC, NS and LS. While allocating tasks, the CH checks task length of non-flexible tasks and compare the buffer length of S nodes in schedulable state. Then, CH allocates non-flexible tasks to S nodes in schedulable state. When number of S nodes in schedulable state is less than task length of non-flexible tasks, then the CH looks for nodes in LS state. Once, the allocation of non-flexible tasks are gets over, the CH goes for flexible tasks. The task allocation strategy is given in Algorithm 1.

**Algorithm 1**

*Let S be the set of server nodes and C be the set of client nodes*
*Consider nTi be the set of non-flexible tasks and fTi be the set of flexible tasks.*
*C node receives a new task*
*C transmits task details to CH*
*CH looks for flexible field*
*If (flexible field = 1) Then*
  *6.1 The task is flexible*
  *6.2 Else if (flexible field = 0) Then*
  *6.3 The task is non-flexible*
*End if*
*For (non-flexible tasks)*
  *8.1 CH collects work load, CPU speed and distance information's of S' nodes*
  *8.2 CH forwards these information's into fuzzy system*
  *8.3 Nodes are separated into three states as SC, LS and NS*
  *8.4 CH compares nTi task length with buffer availability of S' nodes*
  *8.4.1 If (task length (nTi) < S (buffer availability)) Then*
    *8.4.1.1 Allocates nTi among S' nodes in SC state*
  *8.4.2 Else if (task length (nTi) > S (buffer availability)) Then*
    *8.4.1.2 Allocate nTi among S' nodes in LS state*
  *8.4.3 End if*
*For (flexible tasks)*
  *9.1 CH allocates tasks among S' nodes in LS state*

Our proposed technique is well suited for all real time applications, where buffer capacities and number tasks to be computed vary over time. This technique schedules the tasks adaptively and improves the quality of network. Using fuzzy logic system, this technique provides accurate load balancing and scheduling.

## 4 Simulation result

### 4.1 Simulation model and parameters

In this section, we examine the performance of our FLBTS with an extensive simulation study based on NS-2 (Network Simulator, http:///www.isi.edu/nsnam/ns). The simulation topology is given in Figure 6. We compare our results with queue-based scheduling (QBS) with normal scheduling (Viswanathan et al., 2007). Various simulation parameters are given in Table 3.

### 4.2 Performance metrics

In our experiments, we measure the following metrics.

- *Scheduling delay:* It measures the average delay occurred while scheduling given tasks.
- *Throughput:* It is the amount of work received in terms of Mbit/s at the receiver.
- *Data loss:* It is the total number of data packets dropped at the receiver side.
- *Success ratio:* It is the ratio of number of tasks completed successfully to the total number of requested tasks.

**Figure 6** Simulation setup (see online version for colours)

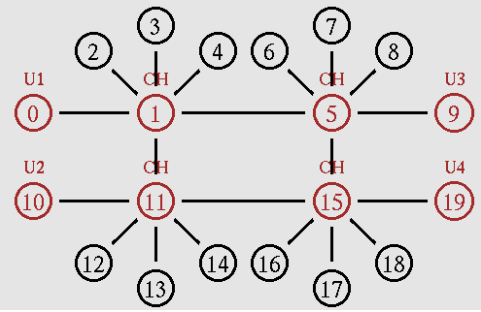



**Table 3** Simulation settings

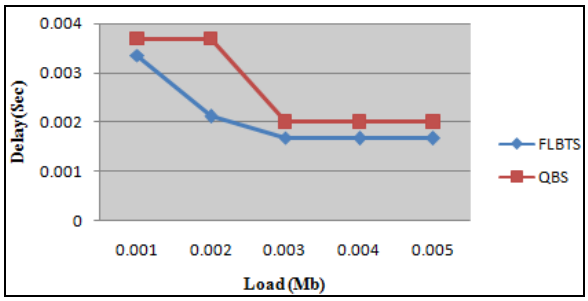
| | |
|---|---|
| Mobile nodes | 12 |
| Users | 4 |
| Area size | 1,000 × 1,000 |
| Mac | 802.11 |
| Radio range | 250 m |
| Simulation time | 50 sec |
| Traffic source | CBR |
| Packet size | 512 |
| Rate | 1, 1.5, 2, 2.5 and 3 Mb |

### 4.3 Results

#### 4.3.1 Based on load

In the initial experiment, we vary the WL of the nodes by varying the size of the tasks as 1, 1.5, 2, 2.5 and 3 Mb.

**Figure 7** Load vs. scheduling delay (see online version for colours)




Figures 7 and 8 show that delay and packet drop of FLBTS is significantly less than the existing QBS technique respectively.

Figures 9 and 10 show that the throughput and success ratio of FLBTS is higher than the existing QBS method respectively.

**Figure 8** Load vs. data loss (see online version for colours)

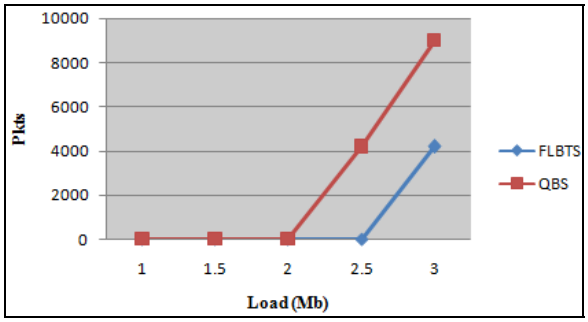



**Figure 9** Load vs. throughput (see online version for colours)

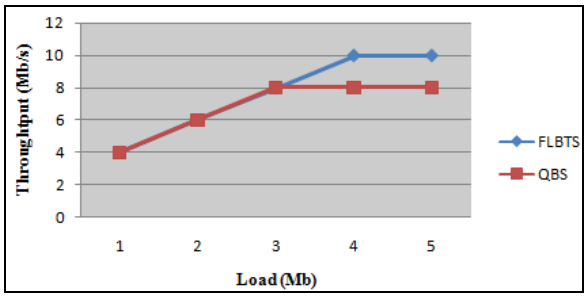
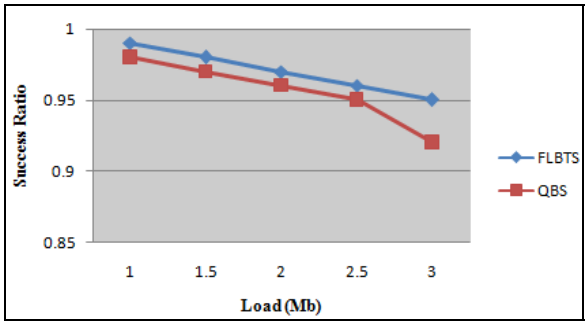



**Figure 10** Load vs. success ratio (see online version for colours)

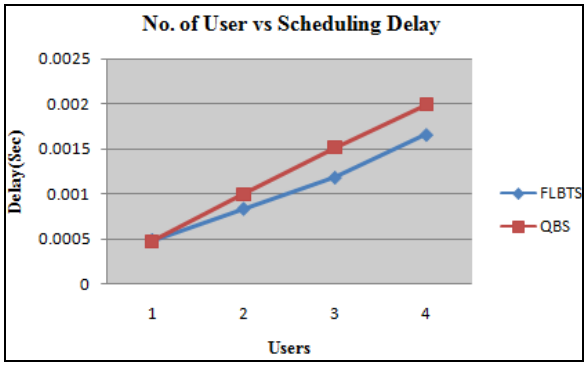



### *4.3.2 Based on users*

In our second experiment, we vary the number of users from 1 from 4 each requesting variable number of tasks with size 1 Mb.

Figures 11 and 12 show that delay and packet drop of FLBTS is significantly less than the existing QBS technique respectively.

**Figure 11** Users vs. delay (see online version for colours)




Figures 13 and 14 show that the throughput and success ratio of FLBTS is higher than the existing QBS method respectively.

**Figure 12** Users vs. drop (see online version for colours)

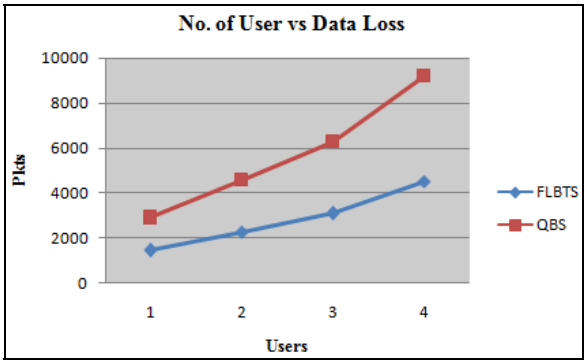



**Figure 13** Users vs. throughput (see online version for colours)

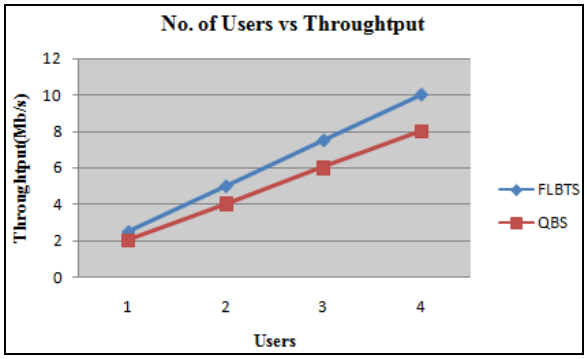
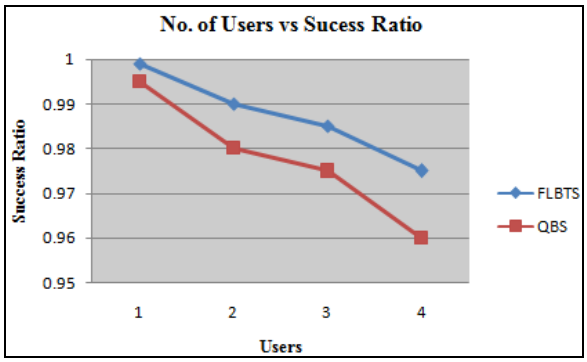



**Figure 14** Users vs. success ratio (see online version for colours)




## 5 Conclusions

Load balancing and task scheduling are old problem. But new solutions are required in modern distributed computing system. Load balancing and task scheduling algorithm plays very important role in distributed system for managing load, and assigning them to appropriate resources. An efficient load balancing and task scheduling algorithm can reduce the total execution time and increases throughput of system. In this paper, we have proposed a FLBTS in distributed systems to improve the performance of distributed system. Fuzzy logic technique has very much potential in solving load balancing and task scheduling problem. Fuzzy logic

make absolute output from uncertainties input. Our technique dynamically handles scheduling and load balancing at hand. The simulation result shows that efficiency of our technique is better than existing QBS technique. In future we will present new load balancing and task scheduling technique using genetic algorithm and neuro-fuzzy technique.

## References


Aida, K. and Casanova, H. (2009) 'Scheduling mixed-parallel applications with advance reservations', *Cluster Computing*, Vol. 12, No. 2, pp.205–220.

Basu, A., Bensalem, S., Peled, D. and Sifakis, J. (2011) 'Priority scheduling of distributed systems based on model checking', *Formal Methods in System Design*, Vol. 39, No. 3, pp.229–245.

Bharadwaj, V., Ghose, D. and Robertazzi, T.G. (2003) 'Divisible load theory: a new paradigm for load scheduling in distributed systems', *Cluster Computing*, Vol. 6, No. 1, pp.7–17.

Chow, K.P. and Kwok, Y.K. (2002) 'On load balancing for distributed multiagent computing', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 8, pp.787–801.

Grosu, D. and Chronopoulos, A.T. (2005) 'Noncooperative load balancing in distributed systems', *Journal of Parallel and Distributed Computing*, Vol. 65, No. 9, pp.1022–1034.

Grosu, D., Chronopoulos, A.T. and Leung, M.Y. (2001) 'Load balancing in distributed systems: an approach using cooperative games', in *Parallel and Distributed Processing Symposium, Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, IEEE, April, 10pp.

Guo, J. and Bhuyan, L.N. (2006) 'Load balancing in a cluster-based web server for multimedia applications', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 17, No. 11, pp.1321–1334.

Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K. and Goldberg, A. (2009) 'Quincy: fair scheduling for distributed computing clusters', in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ACM, October pp.261–276.

Jing, Z., Xuefen, C., Guan, L. and Hongxia, L. (2008) 'Service-aware multi-constrained routing protocol with QoS guarantee based on fuzzy logic', in *22nd International Conference on Advanced Information Networking and Applications-Workshops*, *AINAW 2008*, IEEE, March, pp.762–767.

Kameda, H., Fathy, E.Z., Ryu, I. and Li, J. (2000) 'A performance comparison of dynamic vs. static load balancing policies in a mainframe-personal computer network model', in *Proceedings of the 39th IEEE Conference on Decision and Control*, IEEE, Vol. 2, pp.1415–1420.

Lin, X., Lu, Y., Deogun, J. and Goddard, S. (2007) 'Real-time divisible load scheduling for cluster computing', in *13th IEEE on Real Time and Embedded Technology and Applications Symposium*, *RTAS'07*, IEEE, April, pp.303–314.

Nadiminti, K., De Assunçao, M.D. and Buyya, R. (2006) 'Distributed systems and recent innovations: Challenges and benefits', *InfoNet Magazine*, Vol. 16, No. 3, pp.1–5.

*Network Simulator* [online] http:///www.isi.edu/nsnam/ns.

Nikravan, M. and Kashani, M.H. (2007) 'A genetic algorithm for process scheduling in distributed operating systems considering load balancing', in *Proceedings 21st European Conference on Modelling and Simulation Ivan Zelinka, Zuzana Oplatkova, Alessandra Orsoni*, ECMS, June.

Sharma, S., Singh, S. and Sharma, M. (2008) 'Performance analysis of load balancing algorithms', *World Academy of Science, Engineering and Technology*, Vol. 38, No. 3, pp.269–272.

Viswanathan, S., Veeravalli, B. and Robertazzi, T.G. (2007) 'Resource-aware distributed scheduling strategies for large-scale computational cluster/grid systems', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 10, pp.1450–1461.


## Websites


http://www.wikipedia.org.