RESEARCH ARTICLE OPEN ACCESS

A Trident Extensible Multiprocessor Network

[1]Shaifali Agarwal,[2]ShivangiAgarwal,[3]Amit Kumar,[4]Dr. Nishant Srivastava
[1,2]Pre-Final Student ,[3,4]Assistant Profressor
[3,4] Computer Science & IT ,Faculty of Engineering & Technology, Jaypee University Anoopshahr,India
Corresponding Auther: 1shaifali Agarwal

ABSTRACT- Recently the thrust for higher and higher computing power is increasing day by day, user don't miss an opportunity to use even there smaller and smaller laptop for this purpose to get their requirement full filled . As a first attempt version of 4 node multiprocessor architecture has been proposed in compact form connected to fulfill the properties of multiprocessor network .It has been found that the architecture is linearly extensible and its performance can be compared with the existing commercial multiprocessor architecture. The proposed 4 node multiprocessor network performs equally good as compare to the existing our reported multiprocessor network .This economical compact multiprocessor can be used for higher computation purposes .

General Terms Parallel and Distributed Systems, Scheduling & Load Balancing.
Keywords: Linearly, Multiprocessor, Scheduling and Extensibility, Two Round Scheme ,Minimum Distance Scheme.

Date of Submission: 08-06-2018 Date of acceptance:23-06-2018

I. INTRODUCTION

Research is to design a multiprocessor network (interconnection) network with lesser no of node having better characteristics then the existing network .Lesser no of node means economical. The other important characteristics of a multiprocessor network a diameter, connectivity, extensibility, fall tolerance, The efficient management of parallelism on an interconnection network involves optimizing conflicting performance indices, like the minimization of communication and scheduling overheads and uniform distribution of load among the nodes[4]. In such a system more than one nodes process the various jobs concurrently. Each job may consist of various tasks that could be executed independently. The number of tasks allocated to each processor has to be controlled in such a way that a high speed execution of processes may occur while maintaining high processor utilization. In such a system, if some nodes remain idle while others are extremely busy, system performance will be degraded drastically. Therefore, scheduling of tasks becomes an important problem for multiprocessor system architectures and consequently it has a substantial effect on the system performance and utilization. It is required that all the processors should share the load evenly that would lead to complete the job in minimum possible time

Scheduling may be performed at the local level or global level based on the information they use to make load balancing decisions . In the global schemes, the scheduling decision is made using global knowledge: i.e. all the processors take part in the synchronization and send their performance profiles to the scheduler. Scheduling algorithms can be classified as either static or dynamic. The static algorithm performs by a predetermined policy, whereas, the dynamic algorithm makes its decision at run time according to the status of the system[11],[12],[13] .

The important parameter when dynamic scheduling algorithms are implemented on a parallel system is the configuration of the interconnection network. The parallel system generally uses a regular point-to-point interconnection network, instead of a random network configuration. Over the years, many different interconnection networks have been used in commercially available concurrent systems and numerous research prototypes have been proposed and evaluated in the literature[1],[2] . Prime examples are found in tree network, Hyperloop network, novel extensible network(NEW)[3],[4],[15],[16],Trident extensible multiprocessor network(TEN)[5],[6],[7],[8]. The choice of the topology of the interconnection network is critical in the design of massively parallel computer systems. Interconnection networks may be categorized into two major groups on the basis of their complexity and scalability. The first category includes high complex networks because of their exponential expension and hence posses poor scalability[7],[8],[9]. Some examples are hyperloop networks etc. The second category of multiprocessor systems is of Linearly Extensible Networks, which are lesser complex[1].,[2] These networks are highly scalable networks i.e. the size of the system (e.g., the number of nodes) can be increased with minor or no change in the existing configuration. In this paper two linearly extensible multiprocessor interconnection networks having similar topological properties are considered for the purpose of simulation (Fig. 1.2 to Fig.1. 3). In addition the performance is also evaluated for standard TEM architecture (Fig. 1.1) and a comparative study of FOUR node network is carried out and shown below in the table. The important properties of these interconnection networks are given in Table 1.

Parameter	**Hyperloop**	**Novel extensible**	**Trident Extensible**
No. of Processor	N=3+n+1	N=4+(5n/2)	N=E-2*n
Degree	4	4	4
Extensibility	3	Invert mirror image	1
Diameter	O(n)	O(2)	O(1)

Figure: Table(1)

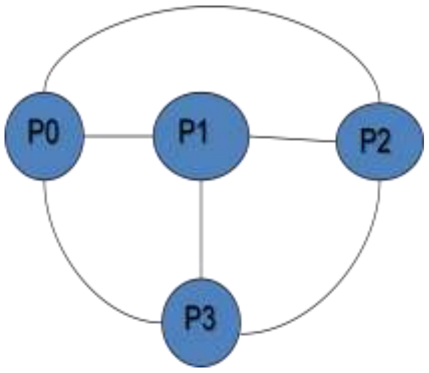
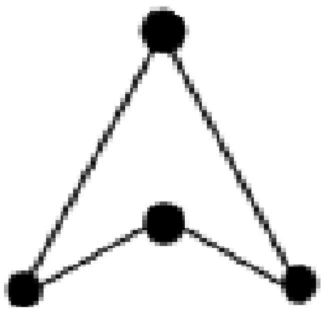
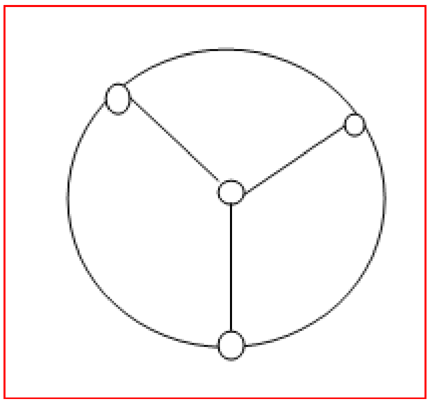


Figure (1.1): TEN


Figure(1.2) NEW


Figure(1.3) HYPERLOOP

II. DYNAMIC TASK SCHEDULING PROBLEM

The performance of a multiprocessor system can be characterized by communication delay, distribution of load among the processors and scheduling overhead[8],[9],[10],[11]. There are many schemes which are based on the principle of minimum distance feature Minimum distance is the property which assures the minimization of the communication in distributing subtasks and collecting partial results. A scheduling scheme operates with this property such as Minimum Distance Scheduling (MDS) minimizes overhead and ensures the maximum possible speedup, however, at the cost of idle unconnected node[4]. In this scheme, the adjacency matrix of the network is used to satisfy the minimum distance property. A „one" in the matrix indicates a link between two nodes whereas a „zero" indicates there is no link between nodes. For load balancing, the MDS algorithm determines the value of Ideal Load (IL) at various stages of the load (task generation). IL is calculated by summing the load of each node in the network divided by the total number of nodes available in the network. The processors having a load value greater than the IL are considered as overloaded processors. Similarly, processors having lesser load than the value of IL are termed as underloaded processors. In other words the overloaded (donors) and underloaded (acceptors) processors are identified based on a threshold value known as IL. Each donor processor, during balancing, selects tasks for migration to the various connected and underloaded processors (i.e. the processors having a „one" in the adjacency matrix) and thus maintaining minimum distance. Mostly any load balancing algorithm considers the overall load on the network. However, in this algorithm the load is mapped through various stages of the task structure. Each stage represents a particular state of the task structure which consists of finite number of tasks.

2.1 Dynamic Load Model

For the purpose of simulation we assume a simple problem characterization in which the load is partitioned into a number of tasks. Each task can be an independent program or partitioned modules of a single program. However, all the tasks are independent and may be executed on any processor in any sequence. The scheduling performance of the strategy has been tested on the three different networks by simulating artificial dynamic load. In order to simulate the load on the given networks, it is characterized into two groups of task structures i.e. uniform and non-uniform load. For a meaningful simulation, tree structures that forms a representative sample of programs are needed which are to be executed on the network. The tree is considered as a test problem on which the schemes are to be applied. In case of uniform load, tasks are generated in a deterministic manner in the form of a regular tree. Each node of the tree represents a task, and executed in parallel in breadth-first manner starting from the root task which is assigned to some given nodes of the network. The total number of nodes in the task tree at a level represents a particular stage of the load.

In order to characterize non-uniform load (non-deterministic load), the total problem is conceived to be an arbitrary tree which unwind itself level by level. A task scheduled on a processor spawns an arbitrary or random number of subtasks, which are part of the whole problem tree. Thus the load on each processor is varying at run time creating unbalance, and balancer/scheduler has to be invoked after each stage.

Using the above pattern of task structure (load), the performance of the networks has been tested for various scheduling schemes as well as with a new scheduling scheme. The performance is measured in terms of Load Imbalance Factor (LIF) i.e. the load imbalance left after a balancing action at each stage of the load. The above simulation has been performed on various similar multiprocessor networks using IBM server X series 226 having Intel Xeon 3.0 GHz processor.

III. DESIGN & ANALYSIS

The TEM network grow linearly as an inverted image of its previous step/level. Let Q be a set of N identical processors represented as

Q= {P_0, P_1……, P_{n-1}}

The number of processor N in the network is given by

N= no. of node + 2*n

Where, n is the level or steps of network (n ϵ Z and n>0) & 4 is the no of nodes of 0th level or the basic NEM network itself. For n=1, an TEM architecture of 4+2*1= 6 interconnected processor can be obtained. Similarly for n=2 there will be 9 interconnected processor.

In order to define the link function we donate each processor in set Q as P_n. They are numbered anticlock wise. The arrangement is shown in Figure (3.1)

The link function can be determined by adjacency matrix of order N*N where N is the number of processor Figure () show the adjacency matrix for proposed network of four processor, where '1' indicates a connection and '0' indicates no connection between nodes.

$$\begin{vmatrix} 0 & \textcircled{1} & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}$$

Figure (3.1): Adjacency matrix for Figure (1.1)

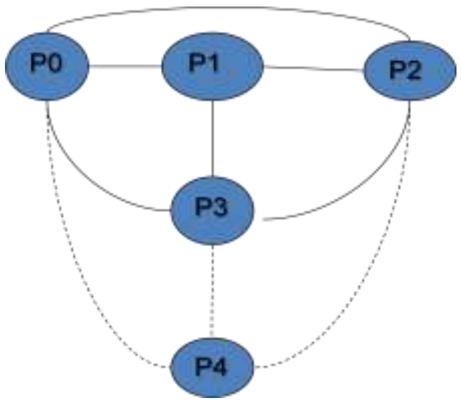
IV. PROPERTIES OF THE TEM NETWORK

This section defines the various methods of connecting processor in a parallel computer. A processor organization can be represented by a graph in which the nodes (vertices) represent processor and the edges represent communication channels between pairs of processors. These processors organization could be evaluated based on certain criteria's or properties of the organization the various properties are:

- **Number of Nodes (N):** The no of nodes in a multiprocessor network plays an important role to evaluate the performance of a multiprocessor system. Lesser the no of nodes, lesser is the system complexity and it is more economical. Therefore, number of nodes should be optimal. The no of nodes in TEN network is N=no.of node +2*nfor n>0 whereas no of nodes in TEN.Due to lesser no of processor in NEW network it may be considered more economical than other networks.
- **Diameter (D):** The diameter of a network is the measure of the maximum inter-node distance in the network. This property is important in determining the distance involved in communication and hence the performance of multiprocessor systems. In the simple words diameter of a network is bound to increase as the size grows unless there is no limit on the no of links.In simple words diameter of a

network is the maximum shortest path between sources and destination node. The path length is measured by the number of links traversed. The network diameter indicates the maximum number of distinct hops between any two nodes, thus provides a figure of communication point of view.

- **Degree (d):** The degree in a network ids defined as the number of connections required at each node. It is the connectivity among different nodes in a network. The degrees of nodes determine the complexity among different nodes in a network. The degree of nodes determine the complexity of the network. Therefore, the node degree should be kept as low, as possible in order to reduce cost. It is best if the number of edges per node is a constant independent of network size, because the processor organization then scales more easily to systems with large number of nodes.
- **Extensibility:** It is the property which facilitates large sized system out of small ones with minimum changes in the configuration of nodes. It is the smallest increment by which the system can be expanded in useful way. In order to avoid the increasing complexity of the system, the expansion must be linear.
- **Bisection Width (b):** The bisection width of a network is the minimum number of edges that must be removed in order to divide by the bisection width puts a lower bound on the complexity of the parallel algorithm.`



Its extensibility is depicted in figure (1.1a) .

V. PERFORMANCE STRATEGY (TRS – TWO ROUND SCHEDULING SCHEME⁾

The basic approach in MDS is to optimize the load balancing among processors under the constraint of the need to keep message path lengths to one hop and thus satisfying the minimum distance property. Migration from donor processor is done to directly connected acceptors only. Thus for every donor, there is a set of acceptors which are outside this set. Referring to Figure (1.1) of NEW network, MDA (P_0)= {P_1, P_3} ,which indicates that even if the processor P_2 is under loaded, it would not be considered as a part of the balancing process. Therefore, a more dynamic nature of algorithm is required to make the networks fully balanced, which takes into consideration those processors also, which are not directly connected.

A new scheme has been proposed for solving load balancing problem with unpredictable load estimates. The proposed algorithm works as an extension of MDS and named as Two Round Scheduling scheme. It is dynamic in the sense that no prior knowledge of load is assumed. TRS scheme takes donor node. There may be more than one path between the donor and acceptor processors which are not connected directly to processors which require multi-hop. However, large number of hopes gives minimum load imbalance and hence, LIF is smaller (i.e., less than the standard range of 40%). The proposed TRS algorithm has a constraint in the scheduling to consider only one processor as intermediate node between donor and acceptor nodes. To perform the load balancing, the algorithm calculates ideal load value for each stage of task structure, which is used as a threshold to factor for kth stage, denotes as $LIF_{k'}$ which is

LIF_k= [max {$load_k$ (Pi)}-(ideal_load) $_k$]/(ideal_load) $_k$

Where (ideal_load) $_k$ = $load_k$ (P_0) + $load_k$ (P_1) +…+ $load_k$ (P_{n-1})]/N,

and max ($load_k$ (Pi)) denotes the maximum load pertaining to stage k on a processor Pi, 0<=i<=N-1, and $load_k$(Pi) stands for the load on processor Pi due to kth stage. Each stage of the task structure (load) represents a finite number of tasks. Based on the IL value, the donor (overloaded) processors and acceptors (under loaded) processors are identified. Migration of task can take place between donor and acceptor processors only.

Trs Algorithm

```
trs( )
{
/* Generate task at 0th processor, tgs indicates task generation at a particular stage*/
/* Consider LMAX is the maximum load on a processor at a particular load stage */
tgs[0] = 1;
while (it_count1 < LMAX)
{
/* calculate IL and RIL */
IL = Calculate_IL (tgs);
RIL = ceil (IL);
printf (tgs);
/* For all processors check whether the load on a particular processor is exceeding the RIL (Rounded IL). If so then migrate the load*/
/* Let the total number of processors are equal to PMAX */
for (it_count2 = 0; it_count2 < PMAX; ++ it_count2)
{
if (tgs [it_count2] > RIL)
{
/* Migrate till load at processors become equal to or less then RIL */
while (true)
{
migrate (it_count2)
if (tgs [it_count2] < = RIL ) break;
} } }
printf (trs)
/* calculate LIF */
LIF = (max(tgs) – IL) / IL;
/* Enter into the next level of the task generation (ts indicates task structure)*/
tgs = ts * tgs;
it_ count ++;
} }
/* Functions used by the algorithm */
Calculate_IL (X[ ])
{
sum = 0; /* x[i] indicates load at ith processor */
for (i = 0; i < PMAX; ++i )
sum = sum + x[i];
return (sum / PMAX);
}
/* Perform migrations */
migrate (p_number)
{
/* Get the set of connected processors to the processor for which migration is being called i.e. p_number */
for (i =0; i < PMAX; ++i )
{
if (connect ed (i, p_number, level))
temp [k++] = i ;
k--;
}
/ * Get the small loaded processor number */
small = temp [0];
for (i = 0 ; i < PMAX; ++i)
if (tgs [temp[i] ] < tgs [small] )
small = temp [j];
/* Transfer the load from p_number to the smallest loaded and connected processors */
while (tgs[p_mumber] != IL || tgs[small] != IL)
{
tgs [p_number] --;
tgs [small] + =1; }
}
/* Check the under loaded processors which are not connected. If any repeat the above procedure for the next level of connectivity */
}
/* Function used to find the maximum load on a processor */
max (X [ ] )
{
max = x [0];
for (i =0; i < PMAX; ++ i)
if (x [i] > max ) max = a [i] ;
return (max);
}
/* Function to check the connectivity of processor i with processor j. Assume the level of connectivity is given (1 or 2)*/
int connected (int i, int j, int level) /* returns true if processors i, j are connected */
{
/* printf("\n node %d is connected to %d: %d", i, j, adj [i][j]); */
if (level = = 1)
return adj [i][j];
for(int k = 0; k < PMAX; k++)
{
if (k = = i || k = = j) continue;
if (connected (i ,k , 1) && connected (k, j, 1 ))
{
/* printf("\n node %d is connected to %d through %d", i, j, k); */
return 1;
} }
return 0; }
end of procedure.
```

VI. SIMULATION RESULTS

The above mentioned TRS scheme has been implemented on IBM server X series 226 having Intel Xeon 3.0 GHz processor in the same environment. The stimulation run consists of generating tasks and executing them on the network of processors i.e. Four processor TEN network under the proposed Two Round scheduling scheme. The result are computed based upon the various types of load as well as for non-uniform load (on random load) generation. To evaluate the performance, the average percentage of LIF is computed, which indicates the load imbalance after a balancing action at each stage of the task structure.

VII. TRS SCHEME ON TEN NETWORK

The load is generated based upon the different stages of the task structures and the balancing action Take place for every stage. A particular stage of task structure represents some fixed amount of tasks. Table and show output of computer generated.

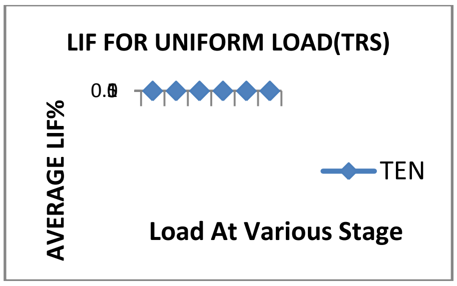
	P_0	P_1	P_2	P_3
TGS[1]:	1	0	0	0
TRS[1]:	1	0	0	0
IL=0.25	RIL=1.000		LIF=0.000	
TT=1				
TGS[2]:	2	0	0	0
TRS[2]:	1	1	0	0
IL=0.5	RIL=1.000		LIF=0.000	
TT=2				
TGS[3]:	2	2	0	0
TRS[3]:	1	1	1	1
IL=1.0000	RIL=1.000		LIF=0.000	
TT=4				
TGS[4]:	2	2	2	2
TRS[4]:	2	2	2	2
IL=2.0000	RIL=2.000		LIF=0.000	
TT=8				

Table(a): Load Migration for Uniform Load on TEN (TRS) up to stage 4(sample output 1).

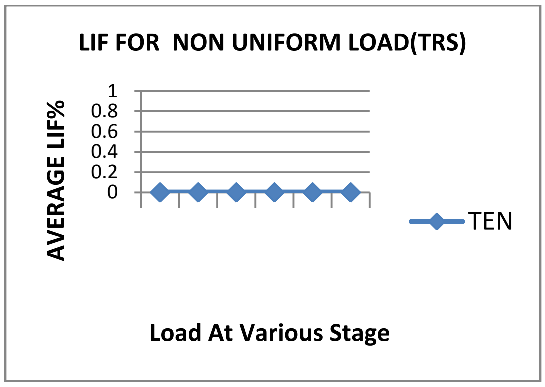
	P_0	P_1	P_2	P_3
TGS[1]:	1	0	0	0
TRS[1]:	1	0	0	0
IL=0.25	RIL= 1		LIF=0.000	
TT=1				
TGS[2]:	1	0	0	0
TRS[2]:	1	0	0	0
IL=0.25	RIL=1		LIF=0.000	
TT=1				
TGS[3]:	2	0	0	0
TRS[3]:	1	1	0	0
IL= 0.5	RIL= 1		LIF=0.000	
TT=2				
TGS[4]:	2	0	0	0
TRS[4]:	1	1	0	0
IL=0.5	RIL= 1		LIF=0.000	
TT=2				

Table (b): Load Migration for non-uniform Load on TEN (TRS) up to stage 4(sample output 2)

Progress of load migration for uniform and non-uniform load respectively on TEN network of four processors (up to stage 4). In each row the entries are: processors (donors and acceptors), TGS (tasks generated at a particular load stage), TR schedule, IL, Rounded IL (RIL), LIF (%) and total tasks (TT) available at a particular stage of load. The behavior of load imbalance is evaluated for both the above mentioned types of load. The average value of LIF is obtained and the curves are plotted as the average percent LIF against the load at various stages (i.e. the problem size) shown in Figure(7.1).



Figure(7.1): Performance of TEN network for uniform load



Figure(7.2): Performance of TEN network for non-uniform load.

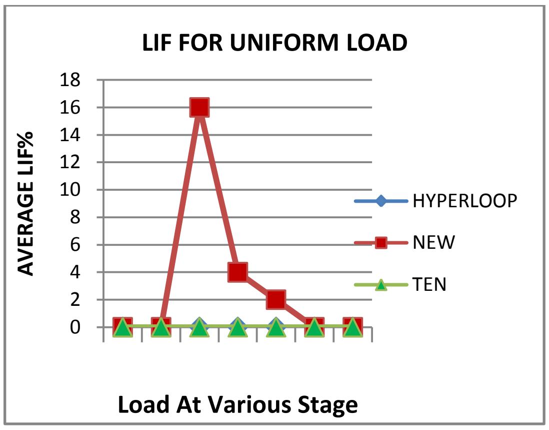


Figure (7.3): TRS scheme on various multiprocessor networks

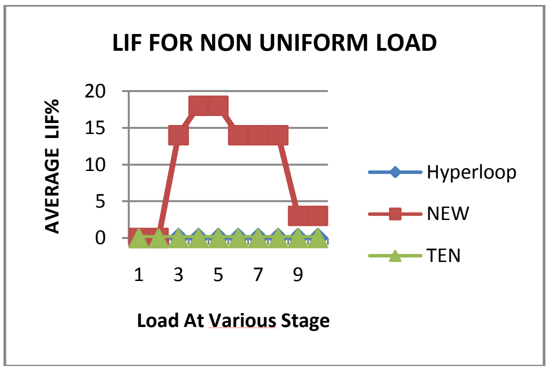


Figure (7.4): TRS scheme on multiprocessor networks

The trend of curves obtained in figure indicates the behavior of the load imbalance factor with respect to load at various stages for uniform task structures. It is observed that LIF initially rises from zero to its high value and then reducing asymptotically. When the number of tasks at a particular stage is lesser than the number of nodes, the LIF shows a higher value and hence a high load imbalance. However, as the number of task increase, the LIF starts reducing (as balancing activity starts its effect) and finally approaches to zero value. For non-uniform load (Figure) value of

LIF starts from zero, reaches to peak and remains same for several stages of the tasks generation. The reason for this high value of LIF for several load stages may be due to imbalance of load which result due to unpredictable load, which is smaller than the number of processors in the network during these stages. In this situation some of the processors may remain idle and hence lacks the efficient utilization of the processing elements. On the other hand when sufficient numbers of tasks are available, the LIF starts reducing. This reduction however, is not that smoother as in the case of uniform load. Figure (7.3) and figure (7.4) shown above also depict the same scenario when compared with other existing linear extensible networks.

VIII. MDS ALGORITHM:

Following are the steps of scheduling algorithms applied and corresponding comparative graph are drawn:(MDS)

1. Mapping of the load to the root processor in the network.
2. Calculate I at any particular load stage.
3. Migrate the task to other processor of the network.
4. Create the subset of acceptors and donors on the basis of IL.
5. Transfer the load from donors to the acceptors on the basis of connectivity of the processor.
6. Repeat step 5.
7. Repeat step 2 to 5 until each and every processor has the same load.

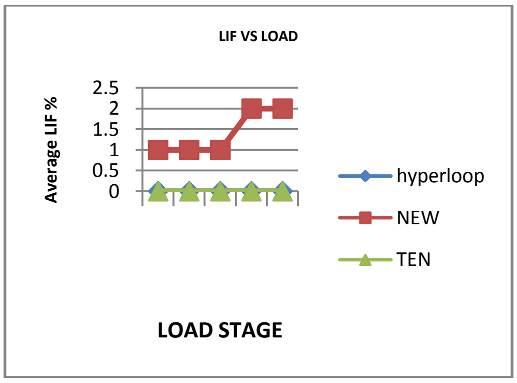


Figure (8.1): Implementation results of MDS algorithm.

IX. CONCLUSION

The overall performance of the multiprocessor system is affected by a number of factors, such as communication delays, imbalance of load among the processor and scheduling overheads. Scheduling plays a vital role to improve the performance of the system and hence a Two Round scheduling algorithm has been proposed and implemented on various similar multiprocessor systems. The performance evaluated in terms of load imbalance and the balancing time. The performance of the TRS algorithm is highly dependent on the connectivity of the various nodes available in the network. However, the algorithm allocates the tasks to the available processors in the network whether they are connected directly or indirectly. From the comparison made on the graphs based on various simulation results, it may be concluded that TRS scheme is performing well on linearly extensible multiprocessor type systems in general and on TEM network in particular while considering the factor of LIF and its balancing time. The proposed TRS scheduling scheme is performing better, degree of balancing is higher and the network utilization is efficient. Therefore, it can be concluded that proposed TRS scheme is ideally suited for linearly extensible multiprocessor networks. The proposed TRS scheme may be applied to other similar multiprocessor network for better network utilization..

ACKNOWLEDGMENT

Author is deeply obliged to the Professor & Head(CSE) M. Qasim Rafiq , Jaypee University Anupshahr giving the idea and helping in implementing this project.

REFERENCE:

[1]. Samad.A,(2009) "Performance Evaluation of Linearly Extensible Multiprocessor Architectures For Networking ", PhD thesis submitted at AMU
[2]. Manaullah, (2013)"A Δ-Based Linearly Extensible Multiprocessor Network",Vol.4(5) pp:700-707 IJCSIT;ISSN:0975-9646
[3]. Samathan , M. R. and Pradhan, D. k.(1989). " The de Brujin multiprocessor network: A versatile parallel processing and sorting network for VLSI " Vol 38 number 4 ,pp 561-581 IEEE Transaction on Computers
[4]. Rafiq, M.Q.,Padam Kumar and Gupta J.P (1999) , " A Novel Tree Structured Multiprocessor Network " International conference on robotics Vision and Parallel Processing for Automation July 16-18 Malasiya.
[5]. A. Samad and M. Q. Rafiq, "A Novel Server Architecture for Networking", In Proceedings of Int"l Conference on Robotics, Vision Information and Signal Processing, Malaysia, 2005, 1029-1032.
[6]. B. Towles and W. Dally. Principles and Practices of Interconnection Network. Morgan Kaufmann Press, san Francisco.
[7]. A. Ishfaq and A. Ghafoor, "Semi-Dostributed Load Balancing For Massively

Parallel Multicomputer Systems", IEEE Transaction on Software Engineering, vol. 17, no. 10, 1991, 987-1004.

[8]. W. M. H. LeMair and A. P. Reeves, "Strategies for dynamic load balancing on highly parallel computers", IEEE Transaction on Parallel and Distributed Systems, vol. 4, no. 9, 1993, 979-92.

[9]. H. Attiya, "Two phase Algorithm for Load Balancing in Heterogeneous Distributed Systems", In Proceedings of 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (Euro-PDP"04), 2004, 434-439.

[10]. M. Bertogna., M. Cirinei, and G. Lipari, "Schedulability analysis of Global scheduling algorithm on multiprocessor platforms", IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 4, 2009, 553-566.

[11]. M. Dobber, R. V. D. Mei and G.Koole, "Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison" IEEE Transaction on Parallel and Distributed Systems, vol. 20, no. 2, 2009, 207-218.

[12]. Yiqiu Fang, Fei Wang, Junwei Ge, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing" Lecture Notes in Computer Science, Issue: 6318, Publisher: Springer-Verlag, 2010, 271-277.

[13]. Bertogna, M., Cirinei, M., and Lipari, G., "Schedulability analysis of Global scheduling algorithm on multiprocessor platforms", IEEE Transaction on Parallel and Distributed Systems, volume 20, number 4, 2009, 553-566.

[14]. D.I. George Amalarethinam and G.J. Joyce Mary, "A new DAG based Dynamic Task Scheduling Algorithm (DYTAS) for Multiprocessor Systems". International Journal of Computer Applications (0975 – 8887) Vol. 19, No. 8, 2011, 24-28.

[15]. Shubham chaudhary,Gaurav Raj,N.K.gupta,"Hyperloop:An Efficient Linearly Extensible Network",ijtrd volume4(6),ISSN:2394-9333.

[16]. Anjali Sharma"A Novel Extensible Multiprocessor Network"2015.

Shaifali Agarwal "A Trident Extensible Multiprocessor Network " International Journal of Engineering Research and Applications (IJERA) , vol. 8, no.6, 2018, pp.46-53