Effective Load Balance Scheduling Schemes for Heterogeneous Distributed System

Zeba Khan[1], Mahfooz Alam[2], Raza Abbas Haidri[3]
[1]Department of Computer Science & Engineering, Institute of Technology & Management, Aligarh, India
[2]Department of Computer Science, Al- Barkaat College of Graduate Studies, Aligarh, India
[3]School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India

Article Info

Article history:

Received Feb 10, 2017
Revised Jun 3, 2017
Accepted Sep 11, 2017

Keywords:

Distributed system
Heterogeneous system
LIF
Load balancing
Resource utilization

ABSTRACT

Importance of distributed systems for distributing the workload on the processors is globally accepted. It is an agreed fact that divides and conquers is the effective strategy for load balancing problems. In today’s time, load balancing is the major issue for scheduling algorithm such as in Parallel and Distributed Systems including Grid and Cloud Computing and many more. Load Balancing is the phenomena of spreading or distributing the workload among the processors so that all processors keep busy for all the time, in order to prevent ideal time of processors. In this work, presents a load balancing algorithm for heterogeneous distributed system (HeDS) with aim of minimizing the load imbalance factor (LIF). The proposed algorithm is using optimization techniques such as Max-Max and Min-Max strategy and applied on Folded Crossed Cube (FCC) network. Makespan, speedup and average resource utilization are also evaluated for performance matrices. The experimental results of the proposed algorithms have showed better in comparison to with previous work under various test conditions.

Corresponding Author:

Mahfooz Alam,
Department of Computer Science,
Al-Barkaat College of Graduate Studies,
Anoopshahr Road, Aligarh, India. Tel:+91-9634423982.
Email: mahfoozalam.amu@gmail.com

1. INTRODUCTION

Distributed System (DS) is a form in which Hardware and Software components situated at computer network communication manage their actions only by passing message. DS is fast emerging field, it keeps evolving and changing to meet user demands. The goals of DS are making resources accessible, distribution transparency, open, scalable, communication and coordination etc. A distributed system is described as collection of either homogeneous system or heterogeneous system [1-2]. Homogeneous Distributed System (HoDS) is that distributed system where collections of identical processors are linked to a high speed network for completing some tasks. Identical processors in the sense that all processors possessed same computational speed, complexity, cache size, equivalent frequencies and functions etc. The benefit of HoDS is that the communication and computation cost are constant in any type of task scheduling algorithm. Heterogeneous distributed system (HeDS) is that DS when all processors work different computational speed, complexity, cache size, frequencies and function etc., are connected with different speed links in order to completing those tasks or solving problems which needs different non identical processors. So, for implementing HeDS is to very difficult as compared to HoDS [3-4]. Distributed System is important to distributing the work load on the processors [5]. The distributed of loads to the processing element is basically known as load balancing problem. However, Load balancing algorithm is plays a important role in homogeneous and heterogeneous distributed system in order to distribution of the tasks with better performance in terms of minimizing LIF, execution time, migration time, maximizing speedup and many more. Today, tasks scheduling is the major issue to be considered for the researchers and various tasks scheduling method are needed in heterogeneous distributed system. This tasks scheduling can be done both the DS that is HoDS and HeDS. It can be categorized into dependent (i.e., Directed Acyclic Graph) and independent task scheduling. In independent tasks scheduling, scheduling of tasks can be run without any dependencies of all other tasks, here tasks can be processed whether other tasks finished or not [6-8]. There are various independent task scheduling algorithms such as Min-Min [9], Max-Min [9-10], Opportunistic Load Balancing [10-11], Dynamic Load Balancing Strategy (DLBS) [2], a heuristic based load balanced scheduling model [12] algorithm etc. In this paper, our contributions are enhancement of the previous work [2] and proposing load balancing scheme for heterogeneous distributed system. The proposed algorithm is using optimization techniques such as Max-Max and Min-Max strategy on the FCC interconnection network. In section 2, it describes the problem formulation and presents the proposed load balancing algorithm named as Independent Task Scheduling with Load Balancing using two techniques Max-Max and Min-Max for heterogeneous distributed system with illustration. In section 3 presents the results and analysis for this work. Finally presents the concluded paper with future work in section 4.

2. RESEARCH METHOD

The proposed work considers load balancing a batch of independent tasks where each task has dissimilar expected time to compute value in order to minimize the LIF on the FCC network. The FCC is a best multiprocessor interconnection network (MIN) as compared with other MINs in terms of diameter [13]. The balancing of load is used to load balancer scheduler for the proposed work as shown in Figure1. Load Balancer places a vital role in fulfilling the request of the clients through servers (i.e., FCC Interconnection Network) and for this work load balancer routes requests to those servers, which has the capability of doing its job in an effective way that is maximization of speed, maximum utilization of capacity and can fulfill the client's requests. For the proposed load balancing algorithm are using FCC network as server which have n number of processor. In this work load balancer checks, which processors are overloaded and underloaded. After determining overloaded and underloaded processors load balancer sends tasks from overloaded to underloaded processor in order to moderating the processors. Thus, all the processors in FCC interconnection network are approximately moderates by migrating the load from overloaded to underloaded processor if the connection exist between the processors. The detail of the scheduler is discussed in the next section. The load balancing algorithms is proposed to schedules a batch of independent tasks for HeDS whereas achieving minimum LIF, makespan, maximum speedup and resource utilization.

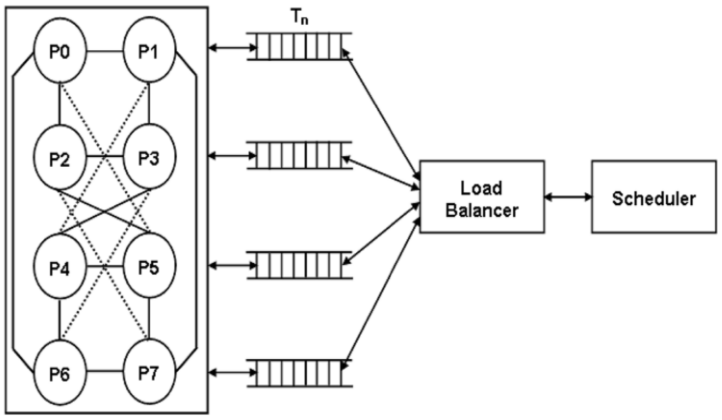

Figure 1. Scheduler model with Load Balancer and FCC3 network

Let us consider that the FCC interconnection network under assumptions has $\boldsymbol{N}$ processors. The set of processors are $\rho = \{P_0, P_1, P_2, \dots\dots.., P_{N-1}\}$ such that $\{N = 2^i \; \forall \; i \in Z \; \& \; i \geq 4\}$ where Z is a natural number. The batch of independent tasks are $\tau = \{T_1, T_2, T_3, \dots\dots\dots.., T_K\}$ randomly allocated on the processors $\{P_m : 0 \leq m \leq N-1\}$ and each task has dissimilar expected time to compute $\{ETC_{ij} : i = 1,2, \dots\dots., K \; \& \; j = 0,1, \dots\dots., N-1\}$. In this part it shows the various components interconnected to the proposed work like notation used which are as follows: $\boldsymbol{N}$ is the number of processors in the FCC network and $\boldsymbol{\tau}$ is set of tasks where as $\boldsymbol{P_m}$ is used for *m*th processor. $\boldsymbol{ETC_{ij}}$ and $\boldsymbol{LEP_m}$ are the Expected Time to Compute of *i*th task on *j*th processor and Load on Each *m*th processor respectively. ***OL***, ***UL***, ***MOD***, ***MOL*** and ***MUL*** are overloaded, underloaded, moderate, maximum overloaded and maximum underloaded processor respectively whereas ***CC***, ***IL***, ***TL*** and ***LIF*** check connectivity, ideal, total load and load imbalance factor respectively. ***SMT***, ***EMT*** and ***MT***, are used for start migration, end migration and migration time. ***MS*** and ***RU*** are makespan and average resource utilization respectively. The batch of independent task is randomly allocated on the processors. After allocation of task, the scheduler computes the load on each P_m by summation of all expected time to compute of ith tasks on *j*th processor. The load on each P_m is calculated as

$$LEP_m = \sum_{i=1\ \&\ (T_i \to P_m)}^{K} ETC_{i,m} \tag{1}$$

Ideal load shows the average load should on each processor as possible which is calculated by equation (2)

$$IL = \frac{TL = \sum_{m=0}^{N-1} LEP_m}{N} \tag{2}$$

The overloaded, underloaded and moderate processors are compared with ideal load. The overloaded underloaded and moderate processors can be calculated as

$$\begin{cases} if(LEP_m > IL) \\ Overloaded\ Processor\ ie., OL(P_m) \\ elseif(LEP_m < IL) \\ Underloaded\ Processor\ ie., UL(P_m) \\ else(LEP_m = IL) \\ Moderate\ Processor\ ie., MOD(P_m) \end{cases} \tag{3}$$

The load distribution of tasks is firstly from maximum OL (MOL) to maximum UL (MUL) processor. The MOL and MUL processor is computed as

$$\begin{cases} MOL > \max_{0 \le i \le m}\{OL(P_i)\} \\ and \\ MUL < \max_{0 \le i \le m}\{UL(P_i)\} \end{cases} \tag{4}$$

The FCC interconnection network is a cube shape network. The degree of each processor is four. For checking connection among the processors are used to adjacency matrix (Adj). Since, its network cube shape so the number of rows (R) and columns (C) will be equal. To check connectivity between any two processors is defined as

$$Adj[R][C] = \begin{cases} 1, & connection\ exist \\ 0, & no\ connection \end{cases} \tag{5}$$

The LIF is important parameter for balancing of load. The LIF is calculated as

$$LIF = \frac{MOL - IL}{IL} \tag{6}$$

The migration time is the time to move of the tasks from one processor to another processor. Migration time is always less for give better performance of the system. The migration time can be estimated as

$$MT = EMT - SMT \tag{7}$$

Makespan is the total completion time of latest task among all the processors in the system. The makespan can be calculated as

$$MS = \max_{0 \le m \le N-1} LEP_m \tag{8}$$

Speedup is defined as the ratio of the time taken by job in serial manner to the time taken by job in parallel. The speedup of the distributed system is calculated as

$$SpeedUp = \frac{TL}{MS} \tag{9}$$

The allocation of resources for completion/tasks should be effective and optimized. The average resource utilization of the heterogeneous distributed system for the batch of independent tasks for a given allocation can be computed as

$$RU = \frac{TL}{N \times MS} \tag{10}$$

In this section, we present the a load balancing schemes ITSLB (Max-Max) and ITSLB (Min-Max) with objective of minimizing the LIF and computing the makespan, speedup and resource utilization for performance evaluation. The objective of this algorithms is to improve our previous work i.e. DLBS algorithm [2]. The DLBS algorithm worked for homogeneous system it means all tasks have identical execution time. So, it is easy to reduce the load imbalance factor. But, the proposed work is designed for heterogeneous distributed system on the same multiprocessor interconnection network. Our approach is to reduce the LIF despite that each task has dissimilar execution time. To perform for this work the LIF can be rewritten of equation (1) as

$$LIF = \frac{MOL}{IL} - 1$$

Since, '1' is a constant factor. So, LIF is dependent on MOL and IL. But IL is also constant variable throughout all iteration. Therefore, for minimum LIF must dependent on MOL.

$$LIF_{min} \ = \frac{MOL}{IL}$$

Due to this reason, firstly, load transfer should be from maximum overloaded processor because lesser MOL will give lesser LIF. Therefore, it is new optimization problem is MOL.

$$LIF_{min} \ \propto MOL_{min}$$

Thus, the proposed ITSLB algorithm is a new strategy for minimization of LIF.

2.1. ITSLB (Max-Max)

Working of ITSLB (Max-Max) algorithm initiates with generation of random tasks, which allocates the processors in randomly fashion with dissimilar ETC of tasks. The scheduler sorts the ETC of all tasks in ascending order on each processor and computes LEP. Computes TL and IL of the system and then indentifies the OL, UL and MOD by comparison with the IL. After calculating all these OL and UL, scheduler determines MOL and MUL then checks for connectivity between the MOL and MUL, if the connection found between the MOL and MUL, migration time starts. The load is transferred through load balancer (i.e., already shown in Figure 1) from the MOL which will have maximum ETC value of the task and goes to MUL then mapped between these processors. Now, next load transfer take place between MOL and MUL, if the MUL has sufficient capacity for receiving the next highest ETC value, otherwise it will transfer to another MUL and continues till the capacity exhausted. After accomplishment of first MOL we take second MOL and this process continues like former process and so on. When all the load transfer finished then migration time stops. If the scheduler does not found connection between MOL and MUL then will go for next MUL and then checks connectivity between these two processors, and connectivity existing, migration will takes place from MOL to MUL, and this step will repeat again and again until and unless all the available processors become approximately moderated and migration time ends. The pseudo code of ITSLB algorithm is given by following steps:

```
1. Generate random ETC matrices
2. Sort ETC in ascending order
3. Compute the LEP and Idea Load using equations (1 & 2)
4. Compute OL, UL and MOD using equation (3)
5. Evaluate MOL and MUL from a set OL & UL respectively using equation (4)
6. Check connection using equation (5)
7. for P :=0 to n do
        if CC ==1
        Start migration time
        Migration start using Max-Max strategy
        Migration start using Min-Max strategy // for section 2.2
        Mapping ( )  // Update LEP
        End migration time
            else
                    Determine next MUL
        Repeat step 8
        end if
        end for
8.  Repeat steps 6-8 unit MOL & MUL is not empty
9.  Compute LIF, MT, Makespan, Speedup and RU using equations (6, 7,8,9&10)
```

As this Table 1 is generated by ITSLB algorithm for 92 numbers of tasks on FCC_3 interconnection network. The FCC_3 interconnection network has 8 processors and 32 connections exist among these processors. By DLBS algorithm for this illustration LIF is **45.45%**. ITSLB (Max-Max) algorithm is generated same number of random tasks and allocated same number of processors but each tasks having dissimilar ETC value and these ETC value are also generated randomly by the scheduler. For better view the illustration of ITSLB (Max-Max) algorithm as shown in Table 1.

Table1. Initial Random Generation Tasks Matrix by ITSLB

P_m	T	ETC Values																			
P0	5	0.125	56.358	19.330	80.874	58.500															
P1	10	47.987	35.029	89.596	82.284	74.660	17.410	85.894	71.050	51.353	30.399										
P2	20	1.49	9.14	36.44	14.73	16.58	98.85	44.56	11.90	0.466	0.891	37.78	53.16	57.11	60.17	60.71	16.62	66.30	45.07	35.21	5.703
P3	3	60.768	78.331	80.260																	
P4	17	51.988	30.195	87.597	72.667	95.590	92.571	53.935	14.233	46.208	23.532	86.223	20.960	77.965	84.365	99.679	99.969	61.149			
P5	11	39.243	26.621	29.728	84.014	2.374	37.586	9.262	67.720	5.621	0.878	91.879									
P6	14	27.588	27.28	58.79	69.11	83.76	72.64	48.49	20.53	74.37	46.84	45.79	94.91	74.44	10.82						
P7	12	59.904	38.523	73.500	60.896	57.240	36.133	15.155	22.510	42.515	80.288	51.71	98.998								

For the analysis of the proposed ITSLB (Max-Max) algorithm, after random generation of ETC values for tasks on the processors, schedulers sorts the all ETC values for each processors in an ascending order then scheduler calculates the LEP for each processor which are 215.187, 585.662, 672.973, 219.359, 1098.826, 394.926, 755.422 and 637.372 respectively. The total load and ideal load calculated by as per equation (2) which is 4579.727 and 572.465. Scheduler also identifies that processor P1, P2, P4, P6 and P7 are overloaded by 13.197, 100.508, 526.361, 182.957 and 64.907 respectively and processors P0, P3 and P5 are underloaded by 357.278, 353.106 and 177.539 respectively. Then scheduler calculates MOL and MUL processors which are P4 and P0. After checking for connectivity between P4 and P0 processors scheduler found that there is no connection between the P4 and P0, then find next MUL i.e., is P3. In P4 and P3 processors exists connection and then load transfer begins. P4 and P3 are overloaded and underloaded by 526.361 and 353.106 respectively. Since P4 has next maximum load and P3 has minimum load to take, that's why migration between P4 and P3 will not takes place in this situation, so in this case P3 becomes partially moderated by 514.597. Therefore, scheduler determines next MUL processor that is P5, now scheduler checks connectivity between P4 and P5, if connectivity exists between these two processors P4 and P5 then migration will takes place from P4 to P5. Similarly, it's continue as per ITSLB (Max-Max) schemes. After that, ITSLB (Max-Max) algorithm calculates LIF is 24.20% as shown in Table 2.

Though we have used ITSLB (Max-Max) algorithm for heterogeneous distributed system environment still we are achieving lesser LIF as **24.20%** as comparison to DLBS algorithm. The migration time and makespan of ITSLB (Max-Max) are 2 msec and 711.017 msec respectively.

Speedup =4579.727/711.017=6.44 and Resource utilization =((4579.727)/(711.017*8))*100=80.51%.

Table 2. Complete Migration Matrix using ITSLB (Max-Max)

P_m	T	ETC Values																		
P0	08	0.125	19.330	56.358	58.500	80.874	98.852	94.915	83.761											
P1	10	17.410	30.399	35.029	47.987	51.353	71.050	74.660	82.284	85.894	89.596									
P2	19	0.466	0.891	1.498	5.703	9.140	11.908	14.731	16.589	16.623	37.788	35.21	36.44	44.56	45.07	53.16	57.11	60.17	60.71	66.304
P3	06	60.768	78.331	80.260	99.969	99.679	95.590													
P4	13	14.233	20.96	23.532	30.195	46.208	51.988	53.935	61.149	72.667	77.965	84.365	86.223	87.597						
P5	12	0.878	2.374	5.621	9.262	26.621	29.728	37.586	39.243	67.720	84.014	91.879	92.571							
P6	12	10.827	20.535	27.289	27.588	45.796	46.845	48.493	58.790	69.118	72.649	74.373	74.443							
P7	12	15.155	22.510	36.133	38.523	42.515	51.710	57.240	59.904	60.896	73.500	80.288	98.998							

2.2. ITSLB (Min-Max)

Working of ITSLB (Min-Max) strategy is same as of ITSLB (Max-Max) but the only difference is migration, which take place from MOL to MUL. The load is transferred (i.e., migration of the task) from the MOL which will have minimum to maximum ETC value of the task alternatively. From ITSLB (Min-Max) algorithm is also achieving lesser LIF as **24.84%** as comparison to DLBS algorithm. The migration time and makespan of ITSLB (Min-Max) are 2 msec and 714.668 msec respectively.

Speedup =4579.727/714.668=6.40 and Resource utilization =((4579.727)/(714.668*8))*100= 80.10%.

After comparing the result of these two strategies, that is ITSLB (Max-Max) and ITSLB (Min-Max) in terms of performance matrices such as LIF, speedup, makespan and resource utilization. The analysis of ITSLB (Max-Max) algorithm gives better values **24.2%, 711.017 msec** and **80.51%** of LIF, makespan and resource utilization respectively but ITSLB (Min-Max) algorithm gives better value **6.40** unit of time of speedup. The migration time of both ITSLB (Max-Max) and ITSLB (Min-Max) are equal but better than DLBS algorithm.

3. RESULTS AND ANALYSIS

To simulate and compute the performance of ITSLB (Max-Max) and ITSLB (Min-Max) is to design on software Code: Blocks in C language using Intel Core i5-6200, x64-based processor with 4GB RAM. The experimental results were concluded to monitor the allocation of the batch of independent tasks on the FCC interconnection network. The tasks are randomly generated between 0-100, 0-1000, 0-25000 etc., and each task's ETC values taken are also between 0.0-100.0 msec in the simulation. The batch of independent tasks is scheduled on the FCC networks as per the ITSLB algorithm which is discussed in section 2.1 and 2.2. The experimental results are to compare our previous work DLBS algorithm. The experimental evaluation is carried out to compare the performance of the algorithm on the performance metrics such as LIF, makespan, speedup and resource utilization as follows:

a. Observing the LIF, Makespan, Speedup and Average Resource Utilization of the processors represent the FCC network while keeping the number of the processors equal but varying the number of independent tasks.
b. Observing the LIF, Makespan, Speedup and Average Resource Utilization of the processors represent the FCC network while keeping the number of independent tasks equal but varying the number of the processors.

3.1. Observations

In Figures 2, 3, 4, and 5 show the case 1 which is representing the variation of LIF, Makespan, Speedup and Average Resource Utilization while keeping the number of the processors equal to eight.(i.e., FCC_3 interconnection network) but varying the number of tasks.

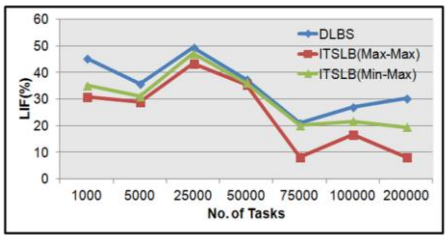

Figure 2. LIF v/s independent tasks

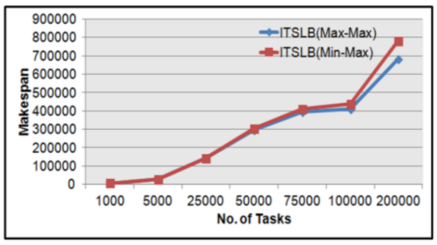

Figure 3. Makespan v/s independent tasks

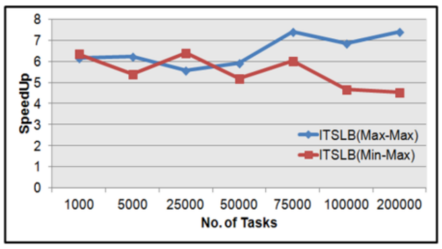

Figure 4. Speedup v/s independent tasks

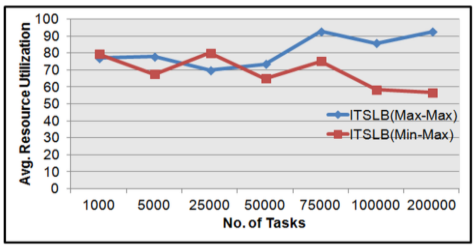

Figure 5. Resource Utilization v/s independent tasks

ITSLB (Max-Max) shows the best performance followed by ITSLB (Min-Max) and DLBS. This is because of the potentially of ITSLB (Max-Max) to simultaneously allocate each independent tasks of the batch on the best processors available resulting in the minimizing the LIF offered to the independent tasks. ITSLB (Min-Max) shows also better perform than DLBS. Makespan, Speedup and RU are also increasing when keeping the number of processors fixed while the number of tasks is increased for all load balancing strategy viz. ITSLB (Max-Max) and ITSLB (Min-Max). ITSLB (Max-Max) shows better Makespan, Speedup and RU as compare to ITSLB (Min-Max).

3.2. Observations

Furthermore, in Figure 6, 7, 8 and 9 show the case 2 which is representing the variation of LIF, Makespan, Speedup and Average Resource Utilization while keeping the number of independent tasks equal but varying the number of the processors are eight and sixteen. As we can see in Figure 6, ITSLB (Min-Max) also shows better LIF from DLBS. LIF goes on decreasing when the number of processors is increased whereas keeping the batch of tasks fixed for all the scheduling strategies viz. ITSLB (Max-Max), ITSLB (Min-Max) and DLBS as expected in such a case on FCC interconnection network as shown in Figure 6. Makespan, Speedup and RU are also decreasing when the increase the number of processors while keeping the number of tasks fixed for all load balancing strategy viz. ITSLB (Max-Max) and ITSLB (Min-Max) as predictable in such a case. ITSLB (Max-Max) shows better Makespan, Speedup and RU as compare to ITSLB (Min-Max).

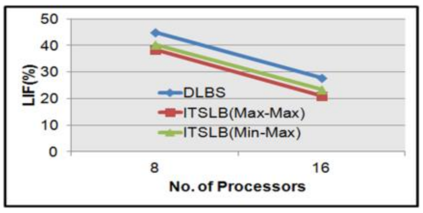

Figure 7. LIF v/s independent tasks

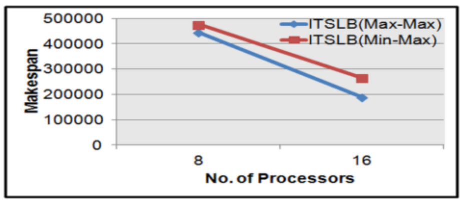

Figure 8. Makespan v/s independent tasks

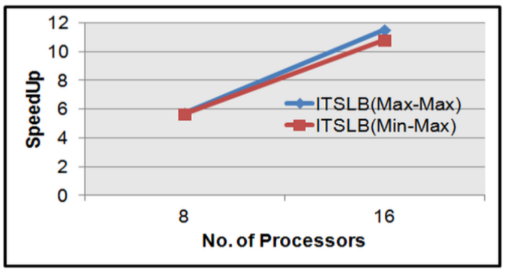

Figure 9. Speedup v/s independent tasks

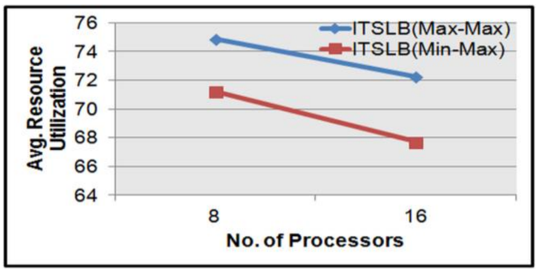

Figure 10. Resource Utilization v/s independent tasks

4. CONCLUSION

Load Balancing (LB) is the phenomena of distributing the approximately equal amount of workload of the processors so that all processors keep busy all the time in order to prevent ideal time of processors. The aim of LB algorithm is to sustain the load to each processing element (PE) such that all the PEs becomes neither underloaded nor overloaded. Therefore, the proper design of a LB algorithm may notably improve the performance of the system. In this paper, a load balancing schemes ITSLB (Max-Max) and ITSLB (Min-Max) algorithms have been proposed to address LIF in the scheduling for heterogeneous distributed system and its application on FCC interconnection network. To the experimental results, ITSLB (Max-Max) algorithm is framed for better performance in terms of different parameters like LIF, makespan, speedup, resource utilization etc. With this ITSLB algorithm, we would be able to construct a fine speedup, reduced LIF and makespan, which consequently can save energy of the systems.

REFERENCES

[1] Singh K, Alam M, Sharma S. “A Survey of Static Scheduling Algorithm for Distributed Computing System.” *International Journal of Computer Applications*. 2015; 129(2): 25-30.
[2] Alam M, Varshney AK. “A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System.” *International Journal of Applied Evolutionary Computation (IJAEC)*. 2016; 7(2): 61-75.
[3] Daoud MI, Kharma N. “A High Performance Algorithm for Static Task Scheduling in Heterogeneous Distributed Computing Systems.” *Journal of Parallel and Distributed Computing*. 2008; 68(4): 399-409.
[4] Jiang Y. “A Survey of Task Allocation and Load Balancing in Distributed Systems.” *IEEE Transactions on Parallel and Distributed Systems*. 2016; 27(2): 585-99.
[5] Potluri S, Rao KS. “Quality of Service based Task Scheduling Algorithms in Cloud Computing.” *International Journal of Electrical and Computer Engineering (IJECE)*. 2017; 7(2).
[6] You T, Li W, Fang Z, Wang H, Qu G. “Performance Evaluation of Dynamic Load Balancing Algorithms.” *Indonesian Journal of Electrical Engineering and Computer Science*. 2014; 12(4): 2850-9.
[7] Yang ZX. “Load Balancing Algorithm of GPU Based on Genetic Algorithm.” *Indonesian Journal of Electrical Engineering and Computer Science*. 2014; 12(6): 4361-7.
[8] Rafsanjani MK, Bardsiri AK. “A New Heuristic Approach for Scheduling Independent Tasks on Heterogeneous Computing Systems.” *International Journal of Machine Learning and Computing*. 2012; 2(4): 371.
[9] Etminani K, Naghibzadeh M. “*A min-min max-min Selective Algorihtm for Grid Task Scheduling*.” In Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on 2007; 1-7. IEEE.
[10] Freund RF, Gherrity M, Ambrosius S, Campbell M, Halderman M, Hensgen D, Keith E, Kidd T, Kussow M, Lima JD, Mirabile F. “*Scheduling Resources in multi-user, Heterogeneous, Computing Environments with SmartNet*.” In Heterogeneous Computing Workshop, 1998. (HCW 98) Proceedings. 1998 Seventh 1998; pp. 184-199. IEEE.
[11] Sang A, Wang X, Madihian M, Gitlin RD. “Coordinated Load Balancing, handoff/cell-site selection, and Scheduling in multi-cell Packet Data Systems.” *Wireless Networks*. 2008; 14(1): 103-20.
[12] T. Sasidhar, V. Havisha, S. Koushik, M. Deep, VK. Reddy, "Load Balancing Techniques for Efficient Traffic Management in Cloud Environment. *International Journal of Electrical and Computer Engineering (IJECE).*, vol. 6, no. 3, pp. 963-973, 2016.
[13] Alam M, Varshney AK. “A Comparative Study of Interconnection Network.” *International Journal of Computers and Applications*. 2015; 127(4): 37-43.

BIOGRAPHIES OF AUTHORs

ZEBA KHAN is pursuing M.Tech. in Computer Science & Engineering from Institute of Technology & Management (ITM) Aligarh India. She received her B.Tech. degree in Computer Science & Engineering from ACN College of Engineering and Management Studies Aligarh India in 2014. Her research areas are Parallel and Distributed System, Load Balancing. She has attended national workshop on Internet and its Applications in Aligarh Muslim University in 2009.

Mahfooz Alam is an Assistant Professor in department of Computer Science at Al- Barkaat College of Graduate Studies (ABCGS), Aligarh, India. He received his M.Tech. degree in Computer Science & Engineering from Dr. A. P. J. Abdul Kalam Technical University, Lucknow, India in 2015. He also completed his M.C.A. and B.C.A. in Computer Applications from Indira Gandhi National Open University (IGNOU) New Delhi, India in 2012 Dec. and 2011 Jun. respectively. His research areas are Parallel and Distributed Systems, Grid and Cloud Computing. He has attended national and international conference in India, and published papers in international journals. You can contact him at: ABCGS, Aligarh, 202021, India.

Raza Abbas Haidri is pursuing Ph.D. at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. Before this he obtained his M.Tech. in Computer Science from the same school. He also completed his M.C.A. in Computer Applications and B.Sc. Hons. (Maths) from Aligarh Muslim University (AMU) Aligarh, India in 2010 and 2006 respectively. His research interests include Cloud Computing, Parallel and Distributed Computing, and soft computing.