Modelling and Evaluation of Multiprocessor Architecture

Preeti Rajput
Department of Computer Engg.
Aligarh Muslim University
U.P., India

Varsha Kumari
Department of Computer Engg.
Aligarh Muslim University
U.P., India

ABSTRACT
Load balancing involves assigning tasks to each processor proportional to its performance and to minimize communication overhead. The assignment can be static-done at compile time, or it may be dynamic- done at run-time. Many load balancing polices achieve high system performance by increasing the utilization of CPU, memory, or a combination of CPU and memory [3]. In this paper Modified Triangle Scheduling Scheme (MTSS) is proposed which modifies the Minimum Distance Scheduling (MDS). This scheme has been implemented on Linearly Extensible Triangle (LE∆) and Linearly Extensible Tree (LET) which reduces the Load Imbalance Factor (LIF) and also the execution time of parallel tasks assigned to the processors.

General Terms
MDS, MTSS, Load balancing, time, Linearly Extensible Tree, Linearly Extensible Triangle, Load Imbalance Factor.

1. INTRODUCTION
Multiprocessor system is a single computer incorporating a number of independent processors that work together to solve a given problem. There are two types of multiprocessor models: shared-memory and message passing system. The shared memory model has single address space and provides a global memory shared by all processors. However, message-passing model has multiple address space and each processor has access to its own local memory. There are number of techniques and methodologies for scheduling processes of a distributed system. These are task assignment, load-balancing, load-sharing approaches. In task assignment approach, each process is viewed as a collection of related tasks and these tasks are scheduled to suitable nodes so as to improve performance. In load sharing approach simply attempts to assure that no node is idle while processes wait for being processed. In load balancing approach, processes are distributed among the nodes of the system so as to equalize the workload among the nodes at any point of time [11].

2. PAPER ORGANIZATION
In this paper we have proposed a load balancing algorithm Modified Triangle Scheduling Scheme (MTSS) for multiprocessor architecture that tries to reduce LIF and execution time of tasks. Section III presents related work in this field. Section IV describes the load distribution. Section V explains the proposed algorithm for LE∆ and LET. Section VI gives the simulation setup and experimental results. Section VII concludes the paper. Section VIII contains the references.

3. RELATED WORK
Various metrics for comparing the load balancing algorithms have been identified in [11]. It also discusses the components of dynamic load balancing algorithms and various dynamic load balancing algorithms. The algorithm adopted for load balancing is closely related to the type of network, number of nodes, number and weight of links which connect the nodes and job size. In order to balance the load uniformly over a grid one has to choose a mix of centralized, decentralized, sender-initiated and receiver- initiated approach. Communication overhead and load balancing time depend upon the approach selected in the algorithm. [3] Considers a cluster computing platform of heterogeneous system in which a set of N nodes are connected via a high speed network. Each node in this model composed of a combination of various resources including processor, memory, disk, network connectivity. Here, a load manager or master node is responsible for load balancing and monitoring available resources of node. [2] Carries out an overview of a six node multiprocessor server, Linearly Extensible Cube, to achieve both load balancing and downloading information efficiently. It implements a new proposed algorithm on the server which uses store and forward like technique that reduces the resource download time. [4] Carries out the study and comparison of six load balancing algorithms, various parameters are used to check the results. It concludes that static load balancing algorithms are more stable in comparison to dynamic and it is also easy to predict the behavior of static, but the dynamic distributed algorithms are always considered better than static algorithms. [5] Concludes that the load balancing algorithm developed leans on a structure of data of network type WAN, what guarantees its portability on any grid computing. The distribution of loads indeed assures the convergence of the algorithm in acceptable time [11].

4. LOAD DISTRIBUTION
Load distribution is the problem of distributing workload among physically dispersed nodes during run time. It is carried out in such a way that a set of independent jobs are distributed among the computing nodes of the multiprocessor architecture so that the jobs are uniformly distributed and none of the nodes are overloaded or underloaded. This is the load balancing problem. A load
balancing algorithm, in general, improves the system performance. However, the degree of improvement not only depends on the specific load balancing algorithm used, but also on the degree of uneven distribution of load over the nodes. Many scheduling and load balancing solutions have been proposed for traditional distributed computing systems [1].

Load balancing strategies may be static or dynamic [1][3]. In static scheduling, the assignment of the tasks to the nodes is done before the execution of the program. A task is always executed on the node to which it is assigned. Dynamic scheduling is based on the re-distribution of processes among the processors during execution time. This redistribution is performed by transferring tasks from heavily-loaded processors to lightly-loaded processors with an aim to minimize the processing time of the application. The flexibility inherent in dynamic load balancing allows for adaptation to unforeseen application requirements at run-time. In general, load-balancing algorithms can be broadly categorized as centralized or decentralized, dynamic or static, periodic or non-periodic, and those with thresholds or without thresholds [3].

5. PROPOSED ALGORITHM FOR LOAD BALANCING

5.1 Linearly Extensible Tree (LET)

.
The LET network combines linear extensibility with small number of processing elements per extension. In LET network the number of nodes at level j is (j+1) [8] as shown in Figure 1.

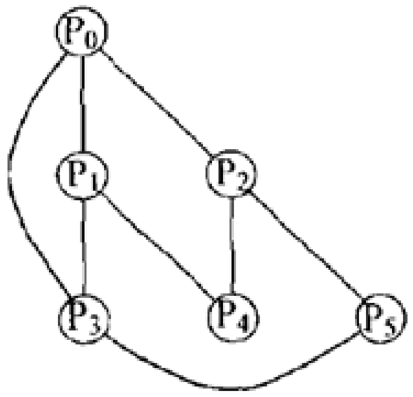

Fig 1: LET Network with 6 Processors

Load balancing starts out at the node with smallest degree i.e. node which has minimum number of links. The load balancing algorithm is given below.

5.2 MTSS for LET network

```
1.  Select the root processor
2.  Allocate ideal load to this processor
3.  Distribute the remaining load on the processors
    connected to the 2*(no of processors in step 1) links
4.  Repeat n (no. of levels) - 1 times to perform load
    balancing on the remaining processors
    4.1  Distribute the remaining load among these
         processors with each processor getting
         (IL)/ step 3
    4.2  Assign same amount of load on processors
         in next subsequent level
    4.3  Add IL to these processors
5.  Calculate final value of load for P
    5.1  $P_0 = P*_0 – 2(N/2) P*_0$
    5.2  $P'_{N-n} = P*_{N-n} + 2(N/2) P*_0$
    5.3  $P_{N-n} = P'_{N-n} - P'_{N-n}/(N/2)$
    5.4  $P_N = P*_N + P*_{N-n}/(N/2)$
    where P* is load obtained from 2, 3 and 4
6.  Repeat till LIF can be minimized no further
    6.1  Find the node with maximum load
    6.2  Assign that node ideal load and distribute
         the remaining load equally on links from
         that processor
```

5.2 Linearly Extensible Triangle (LEΔ)

This triangle-based multiprocessor network has concept of simple geometry and its interconnections topology exhibits the properties of linearly extensible multiprocessor architecture [6][11]. An LEΔ network with four processors is shown in Figure 2. Load balancing starts at the (N-1)th node.

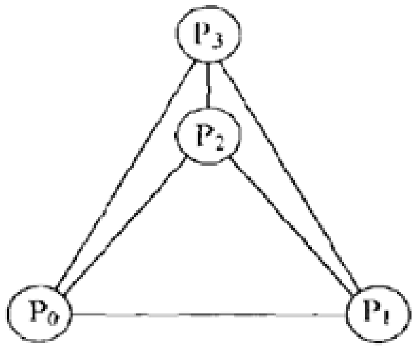

Fig. 2: LEΔ Network with 4 Processors

5.4 MTSS for LEΔ network

```
1.  Assign ideal load to the processor at nth level in an
    N(N=n+3) processor network
2.  Distribute the remaining load equally among the N-
    1 processors with each processor getting (load-
    IL)/N of the remaining load
3.  Load accumulates on the processors forming the
    base of the triangle
4.  Repeat for the base nodes m=1,2,...,(N-1)-2 times
    4.1  Distribute the load on these processors as
         load on P/(i+3): i = 0,1,2,...,(N-1)-2
    4.2  Repeat 4 with remaining load on P
5.  Calculate the final load on P
    5.1  $P_{N-1} = P_{N-1} – (P*_{N-1}/2)$
    5.2  $P_{N-2} = P_{N-2} – (P*_{N-2}/2)$
```

6. RESULTS

The proposed algorithms for LET and LEΔ were implemented in Matlab and the curves obtained were compared with various other algorithms for load balancing as shown in Figure 3 & 4.

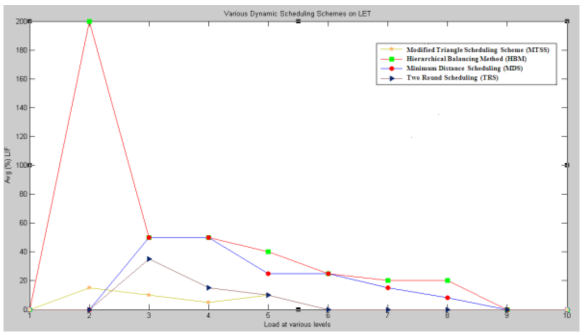

Fig. 3: Load vs LIF for a LET Network

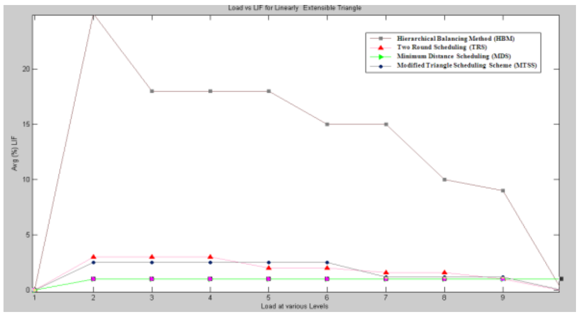

Fig. 4: Load vs LIF for a LEΔ Network

The results of the algorithm for LET & LEΔ are compared with results for Hierarchical Balancing Method (HBM), Minimum Distance Scheduling (MDS), Two Round Scheduling (TRS) and the results obtained are shown above.

Figure 5 shows the average LIF (%) at various stages against no. of processors and Figure 6 shows load balancing time against no. of processors. It can be seen that in comparison to the LET network, LEΔ network gives better results.

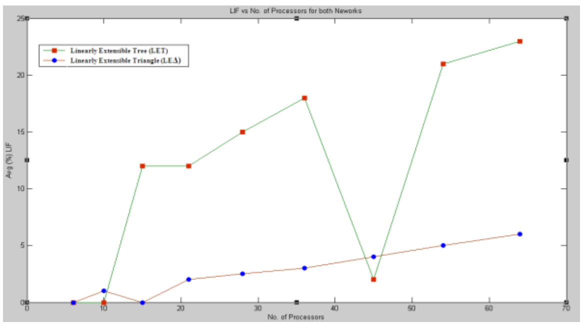

Fig. 5: No. of Processors vs LIF

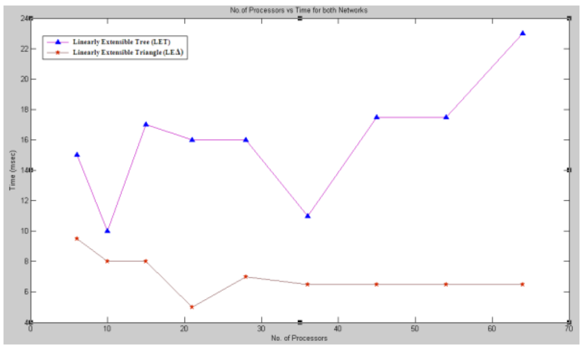

Fig. 6 No. of Processors vs execution time

7. CONCLUSION

From the above results and discussion it is shown that an efficient scalable algorithm for load balancing in the LET and LEΔ networks has been designed. It also reduces the response time of tasks running in parallel. It also reduces the Load Imbalance Factor (LIF) to less than 25 %.

8. REFERENCES

[1] Janhavi B., Sunil Surve, Sapna Prabhu, “Comparison of Load Balancing Algorithms in a Grid”, 2010 International Conference on Data Storage and Data Engineering, pp 20-23, 2010

[2] Abdus Samad, M. Q. Rafiq and Omar Farooq, “A NovelAlgorithm for Fast Retrieval of Information from a Multiprocessor Server”, *7th WSEAS Int’l* Conf. on SOFTWARE ENGINEERING, PARALLEL AND DISTRIBUTED SYSTEMS (SEPADS ’08), University of Cambridge, UK, pp 68-73, 2008

[3] Chandra, Pushpendra Kumar, Sahoo, Bibhudatta, “Dynamic Load Distribution Algorithm Performance in Heterogeneous Distributed System for I/O Intensive Task”, TENCON 2008 - 2008, TENCON 2008, IEEE Region 10 Conference, pp 1-5, 2008.

[4] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, “Performance Analysis of Load Balancing Algorithms”, World Academy of Science, Engineering and Technology 38, pp 269-271, 2008.

[5] Abdallah Boukerram, Samira Ait Kaci Azzou, “Implementation of Load Balancing Algorithm in a Grid Computing”, American Journal of Applied Sciences, 2006.

[6] Manaullah, “Performance Evaluation of Multiprocessor Architectures”, Ph.D. thesis, Jamia Millia Islamia, 2002.

[7] Abdus Samad, “Performance Evaluation of Linearly Extensible Multiprocessor Architectures for Networking”, Ph.D. thesis, Aligarh Muslim University, 2009.

[8] M. Q. Rafiq, “Studies on the Performance Evaluation of a Linearly Extensible Multiprocessor Network”, Ph.D thesis, Univ. of Roorkee, 1995.

[9] D. Acker, S. Kulkarni, “A Dynamic Load Dispersion Algorithm for Load-Balancing in a Heterogeneous Grid System”, Sarnoff Symposium IEEE, pp 1- 5, 2007.

[10] A. Chhabra, G. Singh, “Qualitative Parametric Comparison of Load Balancing Algorithms in Distributed Computing Environment”, 14th International Conference on Advanced Computing and Communication, IEEE, pp 58– 61, 2006.

[11] Ambreen Ahmad, M. Qasim Rafiq, “Design and Development of a Scalable Multiprocessor Architecture”, International Conference on emerging Trends in Technology (ICETT 2011).