International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 10 – No.5, February 2016 – www.ijais.org

Task Allocation on Linearly Extensible Multiprocessor System

Abdus Samad
University of Women's Polytechnic
Aligarh Muslim University
Aligarh, India

Jamshed Siddiqui
Department of Computer Science
Aligarh Muslim University
Aligarh, India

Zaki Ahmad Khan
Department of Computer Science
Aligarh Muslim University
Aligarh, India

ABSTRACT
A novel dynamic scheduling scheme that supports task unbiased structure approach is proposed for multiprocessor networks. The significance of proposed scheduling scheme is remedying the communication overhead, delay in task execution and in efficient processor utilization and hence improves the total execution time. The proposed algorithm is implemented on a set of processors known as nodes which are linked through certain interconnection network. In particular, the recital is appraised for a linear kind of multiprocessor interconnection network known as Linear Crossed Cube (LCQ) multiprocessor system. In addition, a comparison is also made by implementing the same algorithm on other similar standard multiprocessor systems. The performance is examined with regards to the performance indexed known as Load Imbalance Factor (LIF), which corresponds to the variant of load among processors. The comparative simulation research reveals that the proposed scheme provides more advantageous performance in terms of task scheduling on numerous linear along with on cube-based multiprocessor networks.

Keywords
Distributed Control, dynamic load balancing, scheduling scheme, Parallel Systems, Task Scheduling.

1. INTRODUCTION
The beneficial curing of parallelism on an interconnection network involves optimizing sporadic efficiency indices, for instance the cutback of communication as well as scheduling overheads and uniform distribution of load among the nodes. Within this sort of a system numerous nodes method the many jobs at the same time. Each job might consist of an assortment tasks that can be carried out autonomously. The huge number of tasks issued to every processor needs to be managed in these a means that an excellent speed performance of procedures may happen although sustaining large processor utilization.

The Scheduling trouble is usually to sustain a proper signatory of most the tasks among the numerous existing nodes in a multiprocessor network. A collection of independent tasks originate on the root processors. A scheduling policy assumes a set of processors and a set of task mapped on it which are to be serviced by these processors according to a specific policy. Scheduling could be carried out at the local level or global level depending on the information they choose to render load balancing verdicts [1] [2] [3] [4]. Local scheduling is performed by the operating system on the basis of the time-slices of the processor. Global scheduling, however, decides the processor in a multiprocessor system on which a process is to be executed. Scheduling algorithms will also be categorized like possibly static or dynamic [5] [6]. The static algorithm works by a fixed strategy, however, the dynamic algorithm tends to make the selection at operate time based on the position of the system. The predetermined policy generally does not give effective process behavior and increases the execution time [17] [18] [19] [20].

Over the time, many scheduling policies were introduced which are designed to achieve their goals. Some techniques are specific to a particular type of multiprocessor architecture. All these strategies are created employing distinct tactics for instance Minimum Distance Scheduling (MDS), Hierarchical Balancing Method (HBM), Two Round Scheduling (TRS) and Multi-stage Scheduling Scheme [7] [8] [9] [10]. There are algorithms which operate and optimize the task scheduling based on the prediction of process behavior. These algorithms consider the process behavior extraction, classification and prediction. Iterative greedy approach is also a notable scheme to minimize the total execution time and communication cost. The primary concept in such an algorithm is usually to enhance the top notch of the assignment in an iterative way utilizing outcomes from earlier iteration [11].

The Superior of the topology of the interconnection network accompanied by suitable scheduling scheme is a crucial problem in the enactment of massively parallel systems. There are numerous other interconnection networks which are used in commercially existing parallel systems. Some examples are hypercube, Crossed Cube (CC), Star Graph, Linearly Extensible Cube (LEC), and Star Crossed Cube [12] [13] [14] [15]. The choice of interconnection network not only depends upon its architecture but also on the scheduling approach. Some organizational models have their own specific algorithm, on the other hand some are independent. In this paper a recently designed multiprocessor system based on linear architecture namely Linear Crossed Cube (LCQ) is taken into consideration [16]. The performance on the LCQ is evaluated with random load by implementing the proposed algorithm. A comparative study is made by applying the proposed algorithm on other similar multiprocessor networks.

This paper is organized in sections. Describes the proposed algorithm and its implementation on LCQ network in Section 2. Sections 3, the results of the proposed algorithm when implemented on other multiprocessor system are discussed. In section 4 a comparative study is made when other algorithm is implemented on the same networks. Section 5 concludes the paper.

2. PROPOSED SCHEDULING SCHEME

2.1 Scheduling Algorithm

The proposed Scheduling Algorithm is dynamic in the logic that no previous grasp of the load is expected. Decision of migration is taken on the fly based on the current system utilization. Adequate number of nodes with multiple paths are available that may be considered as donor and acceptor nodes. Section of nodes for task migration is challenging and directly effect the communication cost. In order to reduce the communication cost, the proposed scheme selects directly connected nodes at first step for the purpose of task migration. Though multi-hope section gives minimum load imbalance and results minimum value of LIF but with greater communication overhead. Therefore, the proposed scheduling identifies highly imbalance nodes irrespective of their connectivity. After identification it searches those paths between these nodes which require only a single intermediate node that could be involved for task migration. The other nodes though are imbalance, however, are not taken into consideration for task migration. This strategy helps to control the communication cost and overhead on the scheduler. In order to make simulation, the tasks are generated incrementally at various stages of task structure. Each stage of the task structure (load) represents a finite number of tasks.

The load imbalance factor for k_{th} stage, denoted as LIF_k, is defined as:

$LIF_k = [\max \{loadk\ (Pi)\} - (ideal\_load)_k] / (ideal\_load)_k$ ……… (1)

$(ideal\_load)_k = [loadk\ (P0) + loadk\ (P1) + \ldots + load_k\ (PN-1)]/N$……. (2)

In equation (1) max (loadk (Pi)) denotes the maximum load pertaining to stage k on a processor Pi, $0 \leq i \leq N-1$, and load k (Pi) stands for the load on processor Pi due to kth stage. On the basis of ideal load (IL) value, the donor (overloaded) processors and acceptors (under loaded) processors are identified. Migration of tasks can take place between donor and acceptor processors only. The algorithm is implemented in "java" language. A pseudo code of the algorithm is shown in Table 1.

Table 1. The Proposed Scheme

```
Proposed Algorithm ( )
{
/* Task Generation on ith processor*/
/* Let L_Load = Maximum load on ith processor at load stage K */
Let i=1, K=1;
I_DS [0] = 1;
While (initial_Load < L_Load)
{
/* Calculate AL and RAL */
AL = Total_L / T_Proceesor;
RAL = ceil (AL);
Printf (I_DS);
/*perform Migration when load on a specific node is beyond RAL*/
FOR (initial_Load= 0; initial_Load < L_Load; ++ initial_Load)
{
IF (I_DS [initial_Load] > RAL   {
/* perform migration until load on processors becomes = > then RAL */
WHILE (true) {
Migrate (initial_Load)
IF (I_DS [initial_Load] < = RAL) break;
} } }
/* Migration Continued */
/* get the connected node to the node to that migration is being known i.e. PNR */
Migrate (PNR)
{
FOR (i =0; i < L_Load; ++i)
IF (connected (i, PNR, level)
Temp [j++] = i;
J-- ;   }
/* check and list the under loaded processors which are not connected.*/
/ * the lowest loaded node number */
{
Lesser = temp [0];
FOR (i = 0; i < L_Load; ++i)
IF (I_DS [temp[i] < I_DS [Lesser])
Lesser = temp [j];
/* the load transfer   from PNR to the minimum encumbered and linked nodes */
/*perform final (MS) Scheduling*/
MS_S [PNR] --;
MS_S [small] + =1;
 }
Close of process
```

3. SIMULATION AND ANALYSIS OF RESULT

In order to evaluate the performance of proposed scheduling scheme on different multiprocessor networks the tasks are generated and mapped on to multiprocessor systems. These networks have been simulated to test the enactment of the proposed scheme. The tasks are generated in random pattern and mapped onto the multiprocessor network in iterative manner. The imbalance of load is evaluated and tasks are migrated based on the value of ideal load. Tasks are running concurrently regardless of their precedence. When all the tasks are mapped and no further migration is feasible the imbalance on particular stage i.e. LIF is evaluated. The same procedure is repeated for next level of task structure and so on.

3.1 Simulation Results on LCQ

The proposed scheduling scheme is implemented and tested on the recently designed multiprocessor network namely LCQ network with eight processors which is fully connected. The LCQ network consists of N= ∑K nodes with a constant node degree equal to 4. It is an economical topology with lesser diameter (└√N┘) [16]. The major advantage of LCQ network is the linear extension that can be carried out by adding one or two nodes per extension. The behavior of LIF at various levels of task structures are evaluated and depicted in Figure 1. The figure shows that initial value of load imbalance factor (LIF) start from 20 as well as spreads to zero. Initially, the LIF at lesser number of tasks is higher as compare to larger number of tasks. The value of the LIF remain lesser expect for initial stage of task structure. The results show that the algorithm produces good solutions with larger number of task. It is because the larger tasks are balanced efficiently on larger number of processors.

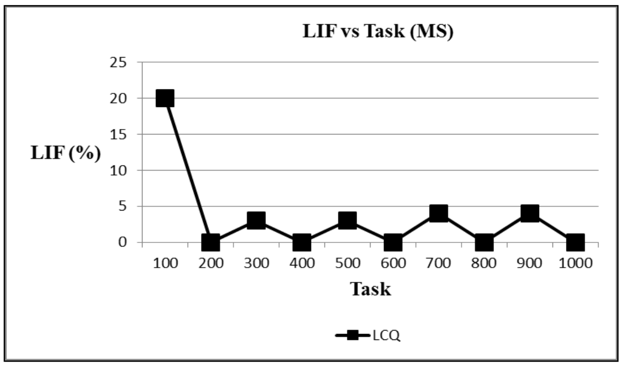

Fig 1: Propose Scheduling Scheme (MS) on LCQ

3.2 Simulation Results on Other Multiprocessor System

Processor topology also has significant impact on the performance on scheduling algorithm. Therefore, the same algorithm is implemented on other architectures namely Hypercube (HC) and Star Crossed Cube (SCQ) networks. In HC more processors are connected directly as compare to LCQ. When the algorithm is implemented the values of LIF are obtained at various level of task structure. The curves are plotted as LIF against number of task and shown in Figure 2 for HC network. The initial value of Load imbalance factor begins from 80 as well as spreads to minimum value of 20 and also then starts increasing. This behavior may be due to the pattern of task structure when mapped on hypercube which is having processors as 2n.

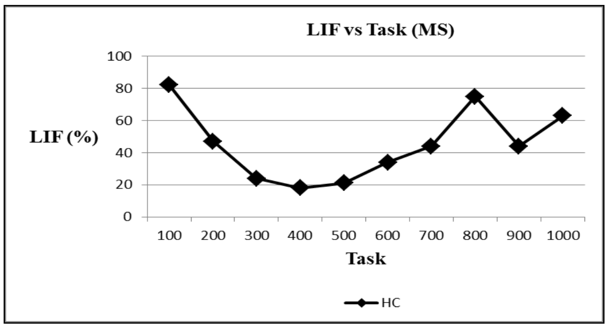

Fig 2: Propose Scheduling Scheme (MS) on HC

The star crossed cube is a recently reported topology in which we considered with eight processing elements for implementation and the results are evaluated with the proposed algorithm. Usually fully connected processor topology future improves the performance of scheduling algorithm. Figure 3 presents the behavior of LIF with SCQ network. The initial value of load imbalance factor (LIF's) begins from 35 and to minimum value of 10 and also then starts increasing. The results are similar to hypercube architecture.

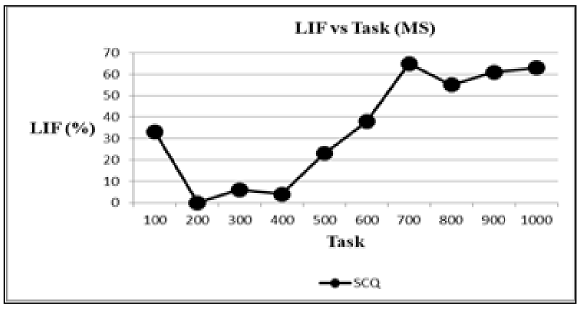

Fig 3: Propose Scheduling Scheme (MS) on SCQ

When comparing the simulation results it is observed that the proposed algorithm producing similar results in cube based networks. The LIF is increasing at higher levels of task structures and tasks are not mapped efficiently. On the other hand when implemented on linear multiprocessor the algorithm gives better results in term of LIF. The initial value of LIF is lesser as well as it reduces with increase in the number of tasks. This trend is depicted in Figure 4.

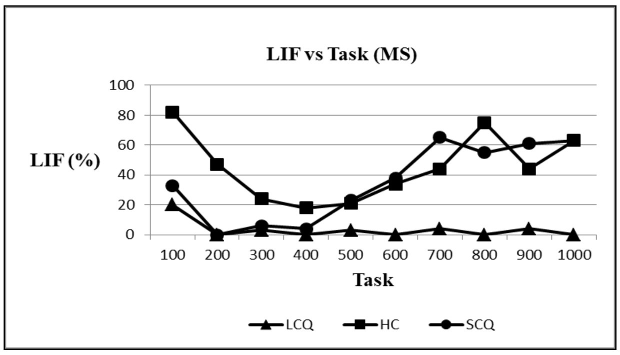

Fig 4: Comparison of proposed scheduling scheme on various networks

4. COMPARATIVE STUDY

The performance of the proposed scheduling scheme, it is desirable to implement other standard dynamic scheduling schemes and also with different networks. In the present work, in addition to the proposed algorithm, one standard reported dynamic scheduling scheme has also been implemented on the LCQ network under the same environment. The scheme known as MDS is considered and implemented to test the performance of LCQ network. This Scheme was designed especially for fully connected networks such as mesh topology. It works on the principle of minimum distance property which considers the directly connected processors for migration.

When MDS algorithm is implemented on the LCQ network the task are scheduled with purely random task structures. The mapping of task is performed at various levels of task structures and behavior is shown in the curves given in Figure 5. It demonstrates that values of LIF initially start reducing with the increase in number of tasks. However, at large number of tasks the scheduler unable to map the task efficiently and hence LIF value become higher. On the other hand when proposed scheduling scheme is implemented on LCQ network the LIF is continuously reducing and become zero at one thousand tasks.

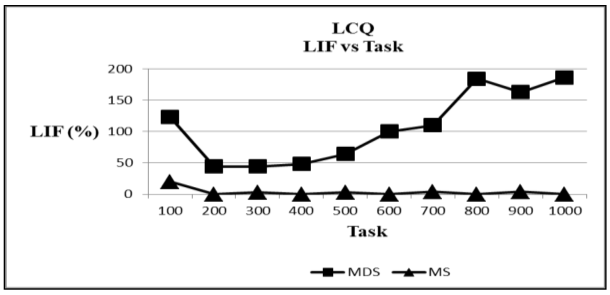

Fig 5: Performance of MDS &Propose Scheme (MS) on LCQ Network

Similar results are evaluated with the MDS algorithms with same task structure for HC and SCQ networks. The curves are plotted and shown in Figure 6 and Figure 7 respectively. The results obtained indicate that the proposed scheduling scheme is performing well when implemented on the HC network, as compare to LCQ network. The initial value of LIF is lesser and it continuously decreasing as larger number of tasks is generated. The major drawback is that MDS scheme produced initial higher LIF value in comparison of proposed scheduling scheme. Therefore, it can argue that the performance of proposed scheduling scheme is comparatively better from the MDS scheme on the same task structure.

Similar results are obtained for SCQ network and shown in Figure 7. The value of LIF becomes higher on SCQ network by MDS scheme. On the other hand when proposed scheduling scheme is implemented on SCQ network the behavior of LIF is similar to those when MDS is implemented but with lesser values of LIF's.

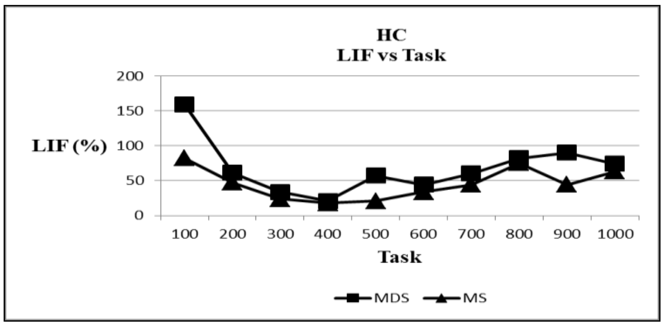

Fig 6: Performance of MDS & Propose Scheme (MS) on HC Network

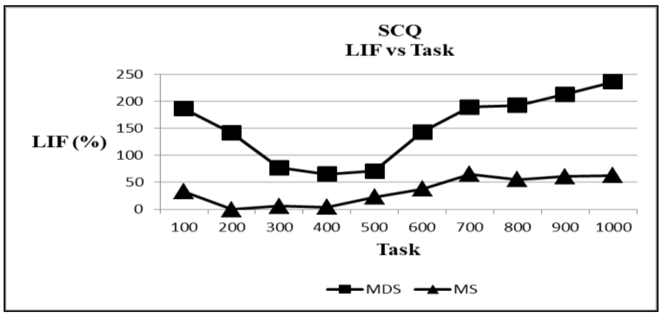

Fig 7: Performance of MDS &Propose Scheme (MS) on SCQ Network

From the above discussion the MDS scheme is carrying out superior on hypercube network as it has better connectivity. The scheme is not suitable for linear type architectures which have lesser complexity.

5. CONCLUSION

This Paper proposed a new scheduling algorithm and implemented on different types of parallel systems. The performance examined by way of load imbalance left after a balancing action. The efficiency of the proposed algorithm is extremely influenced by the connection of the numerous nodes obtainable in the network. Nevertheless, the scheduling scheme allows the tasks to the offered processors in the network whether these are linked instantly as well as partially to not directly linked nodes. From the comparison created on the curves depending on numerous demonstration outcomes , it could be worked out that proposed scheduling scheme is carrying out nicely apropos LIF on linear kind multiprocessor systems usually as well as on LCQ network particularly . The proposed scheduling scheme is superior, degree of balancing is greater and the network usage is economical which is excellent for linear multiprocessor networks.

6. REFERENCES

[1] J. Jia, B. Veeravalli and J. Weissman, "Scheduling Multiprocessor Divisible Loads on Arbitrary Networks", IEEE Transactions on Parallel and Distributed Systems, vol. 21, No. 4, pp. 520-531, 2010.

[2] F. Martelli and A. M. Bonuccelli, "Minimum Message Waiting Time Scheduling in Distributed Systems", IEEE Transactions On Parallel And Distributed Systems, vol. 24, no. 9 pp. 1797–1806, 2013.

[3] A. F. Omara and M. M. Arafa, "Genetic Algorithms for Task Scheduling Problem", Journal Parallel Distributed Computing, vol. 70, pp. 13–22, 2010.

[4] E. Dodonov and D. F. R. Mello, "A Novel Approach for Distributed Application Scheduling Based On Prediction of communication events", Future Generation Computer Systems, vol. 26, pp. 740–752, 2010.

[5] Z. Lan, E. V. Taylor and G. Bryan,"A Novel Dynamic Load Balancing Scheme for Parallel systems", Journal of Parallel and Distributed Computing, vol. 62, pp. 1763–1781, 2002.

[6] R. Hwang, M. Gen and H. Katayama,"A Comparision of Multiprocessor Task Scheduling Algorithms with Communication Costs", Computers and Operations Resaerch, vol. 35, pp. 976-993, 2008.

[7] H. W. M. 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, pp. 979-992, 1993.

[8] A. Samad, M. Q. Rafiq and O. Farooq, "Two Round Scheduling (TRS) Scheme for Linearly Extensible Multiprocessor Systems", International Journal of Computer Applications, vol. 38 no. 10, pp. 34-40, 2012.

[9] 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, pp. 553-566, 2009.

[10] 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, pp. 207-218, 2009.

[11] Q. Kang, H. He, and H. Song, "Task Assignment in Heterogeneous Computing Systems Using an Effective Iterated Greedy Algorithm", The Journal of Systems and Software, vol. 84, pp. 985–992, 2011.

[12] K. Efe, "The crossed cube architecture for parallel computation", IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 5, pp. 513–524, 1992.

[13] C. R. Tripathy, "Star-cube: A New Fault Tolerant Interconnection Topology For Massively Parallel Systems", IE (I) Journal, ETE Div, vol. 84, no. 2, pp. 83-92, 2004.

[14] A. Samad, M. Q. Rafiq, and O. Farooq, "LEC: An Efficient Scalable Parallel Interconnection Network", In proceeding International Conference on Emerging Trends in Computer Science, Communication and Information Technology, pp 453-458, 2010.

[15] N. Adhekari and C. R. Tripathy, "Star Crossed Cube: An Alternative to Star Graph", Turkish Journal of Electrical Engineering and Computer Science, vol. 2, no. 3, pp. 719- 734, 2014.

[16] Z. A. Khan, J. Siddiqui and A. Samad, "Linear Crossed Cube (LCQ): A New Interconnection Network Topology for Massively Parallel System", International Journal of Computer Network and Information Security, vol. 7, no. 3, pp 18-25, 2015.

[17] R. F. DeMello, L. J. Senger and L. T. Yang, "Performance Evaluation of Route", A Load Balancing Algorithm for Grid Computing. RITA, vol. 13, no. 1, pp 87-108, 2006.

[18] A. J. Andrade, R. F. De Mello, E. Dodonov, L. J. Senger, L. T. Yang and K. C. Li, "Toward an Efficient Middleware for Multithreaded Applications in Computational Grid", In proceeding of 11th IEEE International Conference on Computational Science and Engineering, CSE-08, Sao Paulo, Brazil, 2008.

[19] R. P. Ishii, R. F. De Mello and L. T. Yang, "A Complex Network Based Approach for Job Scheduling in Grid Environments", HPC, in: Lecture Notes in Computer Science, Vol. 4782, Springer, pp. 204-2015, 2007.

[20] H. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L.Huang and B. Chapman, "High Performance Computing Using MPI and Open MP on Multicore Parallel System", Parallel Computing, Vol. 37, pp. 562-575, 2011.