I J C T A, 9(40), 2016, pp. 485-490 ISSN: 0974-5572

Optimal Dynamic Scheduling Algorithm for Cube Based Multiprocessor Interconnection Networks

Abdus Samad* and Jamshed Siddiqui & Zaki Ahmad Khan**

ABSTRACT

The Scheduling and mapping of tasks on a set of processors is considered as a critical problem in parallel and distributed computing system. The scheduling problem is to maintain a balanced execution of all the tasks among the various available nodes in a multiprocessor network. In this paper a dynamic scheduling algorithm named as Optimal Multi-Step Scheduling Algorithm (OMSS) has been proposed for scheduling the load on various multiprocessor interconnection networks. In particular, the performance of the OMSS algorithm is evaluated for a special class of multiprocessor system known as cube based multiprocessor networks. A comparison is also made by applying other standard scheduling algorithms on the same networks. The comparative simulation study shows that the proposed OMSS algorithm gives better performance in terms of task scheduling on various cube based multiprocessor networks.

Keywords: Cube Networks, Scheduling Algorithm, Minimum Distance Property, Scheduling Performance Parameter, Multiprocessor Interconnection Networks

1. INTRODUCTION

Over the time, many scheduling policies were introduced which are designed to achieve their goals such as efficient utilization of process elements, minimization of resource idleness or decreasing the total execution time. Some techniques are specific to a particular type of multiprocessor architecture. These approaches are developed using different strategies such as Minimum Distance Scheme (MDS) [1], Hierarchical Balancing Method (HBM) [2] etc. 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 [3]. Iterative greedy approach is also a notable algorithm to minimize the total execution time and communication cost [4]. The main idea in this algorithm is to improve the quality of the assignment in an iterative manner using results from previous iteration [4-5]. These algorithms are applied on specific parallel system and the performance has not been extensively studied on a cube type of multiprocessor system. This paper is devoted to investigate the scheduling problem on a cube multiprocessor architecture [6-7]. The standard dynamic algorithms namely MDS algorithms which was designed originally for tree types multiprocessor networks has been modified to overcome its drawback of large execution time and greater imbalance at earlier stages of load generation. In the proposed algorithm two different parameters are improved. First the execution time is reduced by involving intermediate processor in the balancing process and the second is to reduce the values of load imbalance by extending load migration into two hopes.

The choice of the topology of the interconnection network is critical in the performance of massively parallel computer systems. In this paper four cube based multiprocessor interconnection networks are

* University Women's Polytechnic, Aligarh Muslim University, Aligarh, U.P, India, *Email: abdussamadamu@gmail.com*

** Department of Computer Science, Aligarh Muslim University, Aligarh, U.P, India, *Emails: jamshed_faiza@rediffmail.com & jmi.amu1@gmail.com*

considered for the purpose of simulation. Simulation results are evaluated and a comparative study based on various performance parameters is carried out on the results obtained by the algorithms. The performance is also evaluated for standared hypercube (HC) [8], Folded Hypercube (FHC) [9], Cross Cube (CQ) [10] and Star Crossed Cube (SCQ) [11] architecture and a comparative study is made. The important properties of these interconnection networks are given in Table 1.

Table 1
Summary of Cube Interconnection Network

Parameter	*HC*	*FHC*	*CQ*	*SCQ*
Nodes	2^n	2^n	2^n	n!2m
Diameter	n	n+1	n	(m+1)/2⌉+—⌊3(n-1)/2⌋
Degree	n	n/2	⌈n+1/2⌉	m+n-1
Cost	n^2	n/2*n+1	n⌈(n+1)/2⌉	(m+n-1)(⌈(m+1)/2⌉ +—⌊3(n-1)/2⌋)

This paper is organized in five sections. In Section II, the dynamic scheduling model is described in brief. The proposed algorithm with Pseudo codes is described in Section III. In section IV the simulation and result analysis of the proposed algorithm is implemented on the various networks with same fashion. Section V concludes the paper.

2. DYNAMIC SCHEDULING MODELS

We presume an effective problem characterization wherein the load is partitioned into a large number of tasks needed for simulation. Each task tends to be ‘ program or even partitioned components of a single program. Nevertheless, almost all the tasks are unbiased and also may be executed on virtually any processor at a sequence. The scheduling performance of the technique continues to be analysed on the three different networks by simulating artificial dynamic load. To simulate the load on the given networks, it is characterized into two groups of task structures. Uniform and non-uniform load [12-13]. For a worthwhile simulation, tree structures that forms a representative sample of programs are required that are to be executed on the network. The tree is seen as a test problem whereby the algorithms are to be applied. In the event 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 persuasion beginning with the root task that is allocated to certain given nodes of the network. The total number of nodes in the task tree at a level represents a particular stage of the load. To be able to depict non-uniform load (non-deterministic load), the total problem is invented to be an arbitrary tree which relax by itself level by level [14]. 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 [15-17].

3. OPTIMAL MULTI-STEP SCHEDULING ALGORITHM

There are many algorithms 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 algorithm operates with this property such as Minimum Distance Scheme (MDS) minimizes overhead and ensures the maximum possible speedup. The OMSS is an extension of MDS algorithm. In this algorithm, 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 overloaded (donors) and under

loaded (acceptors) processors are identified based on the value of IL, which is being treated as threshold. Migration of load takes place from overloaded processor to underloaded by applying the connectivity check. Each donor processor, during balancing, selects tasks for migration to the various connected and under loaded processors with the help of adjacency matrix and thus maintaining minimum distance. Comparing execution times is fair only if we compare the best algorithm on each system for overall load. However, to make the algorithm cost efficient the load is characterized into different stages. In this particular algorithm the performance is evaluated for varying the load in each iteration and keeping the system size constant. Each stage represents a particular state of the task structure which consists of finite number of tasks. The load im balance factor for k^{th} stage, denoted as LIF_k, is defined as:

$$LIF_k = [\max \{load_k (P_i)\} - (ideal\_load)_k] / (ideal\_load)_k \quad (1)$$

Where,

$$(ideal\_load)_k = [load_k (P_0) + load_k (P_1) + \ldots + load_k (P_{N-1})] / N, \quad (2)$$

And max ($load_k$ (Pi)) denotes the maximum load pertaining to stage k on a processor P_i, $0 \le i \le N-1$, and $load_k (P_i)$ stands for the load on processor P_i due to k^{th} stage. Based on the IL value, the donor processors and acceptor processors are identified. Migration of task can take place between donor and acceptor processors only after checking connectivity between them. A pseudo code of the algorithm is shown in Table 2.

Table 2
The Optimal Multi-Step Scheduling (OMSS) Algorithm

```
Algorithm: OMSS
Proposed Algorithm ( )
/* The processor i with processor j. Assume the level of connectivity is given (multiple level)*/
Int connected (int i, int j, int level) /* returns true if processors i, j are connected */
{
If (level = = 1)
Return adj [i] [j];
For (int k = 0; k < no_proc; k++)
{
If (k = = i || k = = j) continue;
If (connected (i, k, level-1) = = 1 && connected (k, j, level-1) = = 1)
{
Return 1;
}
}
Return 0;
End of procedure
```

4. SIMULATIONS AND ANALYSIS OF RESULTS

To draw general conclusion about the effectiveness of the Optimal Multi-Step Scheduling Algorithm (OMSS), the simulation run consists of generating various types of load and mapping them on the cube networks namely Hypercube (HC), Folded Hypercube (FC), Cross Cube (CQ) and Star Crossed Cube (SCQ). The estimation of LIF is obtained for various numbers of tasks and the task migration takes place from one node to another node in the form of packets of size such as one, four and eight. The algorithm implemented and tested on the various multiprocessor networks under the same environment. The Minimum Distance Scheduling is considered and implemented to test the performance of cube based network as it was originally

designed especially for fully connected networks such as mesh topology. The minimum distance property is taken into account only the directly connected processors available in the network for migration.

When MDS algorithm is implemented on the cube based 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 1. 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.

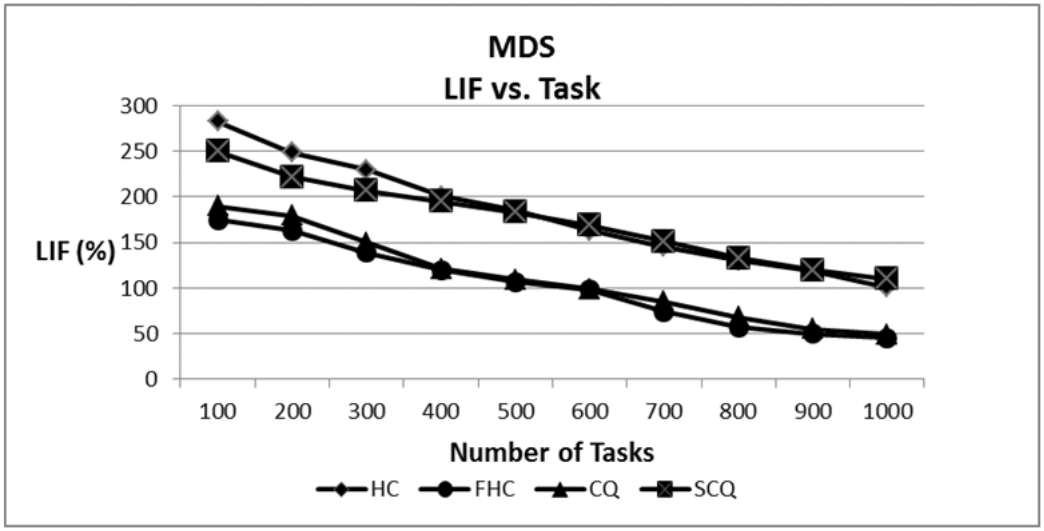

Figure 1: MDS algorithm on Cube Based Networks

In the same pattern proposed algorithm is applied which produces similar results in cube based networks. The time varies on MDS and Other Scheduling Scheme on HC, FHC, CQ and SCQ network. The time is continuously reducing and becomes one at one thousand tasks. However, MDS Scheme shows the lesser balancing time. This trend is depicted in Figure 2.

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. 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 3.

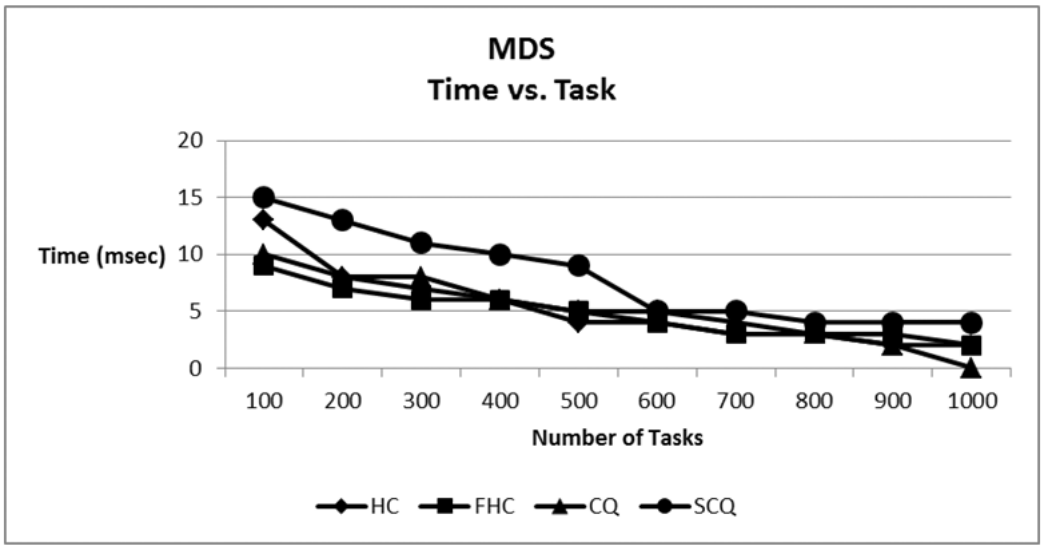

Figure 2: Time Graph of MDS Algorithm on Cube Based Networks

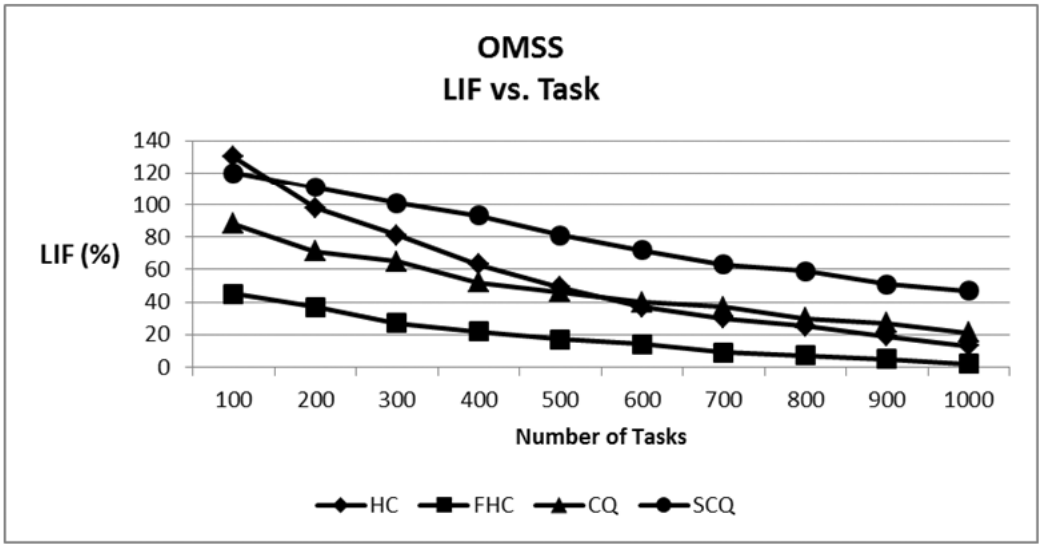

Figure 3: OMSS Algorithm on Cube Based Networks

It is widely recognized that one of the major benefit of parallel computing is to offer a shorter time to solution than the fastest uniprocessor system. The performance results shown in Figure 4 indicate the behavior of balancing time of OMSS algorithm i.e. Time verses Load. The figure shows the speedup for the proposed OMSS algorithm. It is to be noted that there is regular pattern in the balancing time with load. The behavior of the tasks is unpredictable; therefore, the balancing time varies on OMSS on the cube network. The total execution time of OMSS initially starts reducing with the increase in number of tasks. The time is continuously reducing and becomes constant at higher stages of tasks.

The comparison made from the graphs based on various simulation results demonstrates that OMSS scheme is performing well on FHC network considering the factor of LIF and its balancing time. The scheduling scheme is giving better results for FHC in comparison to other tested networks for different types of loads. Therefore, it can be concluded that OMSS scheme and FHC network is a better organization and performance is significantly better particularly for unpredictable load. The overall performance of the OMSS Scheme 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 and partially to indirectly connected nodes. The

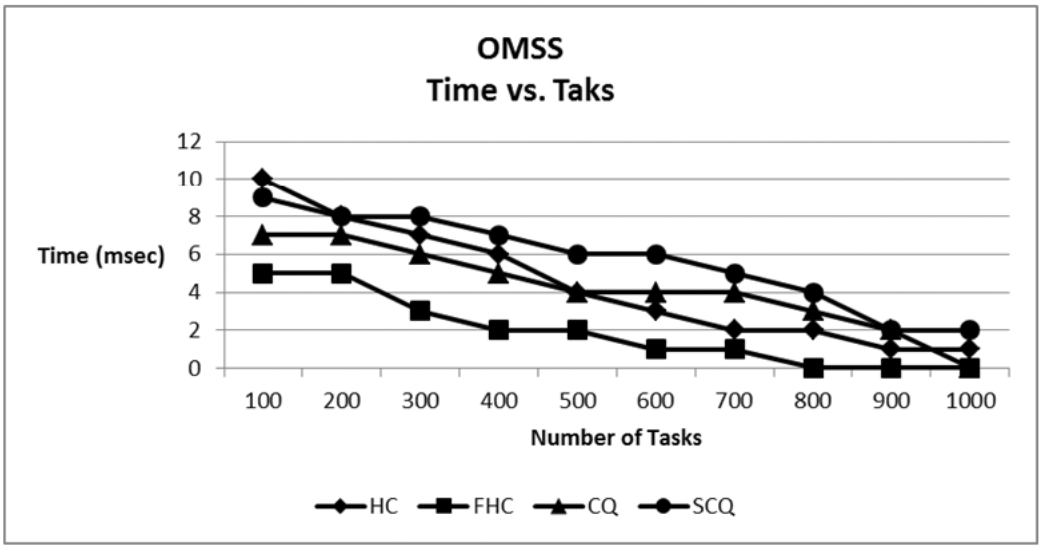

Figure 4: Time Graph of OMSS Algorithm on Cube Based Networks

OMSS scheme is cost effective; degree of balancing is higher and has significant impact on overall parallel performance.

4. CONCLUSION

This Paper proposed a new scheduling algorithm and applied on various cube based multiprocessor interconnection networks in terms of load imbalance left after a balancing action and execution time. The performance of the OMSS algorithm is extremely influenced by the connectivity of the numerous nodes obtainable in the network. Nevertheless, the algorithm allows for the tasks to the available processors in the network whether these are linked instantly and statically. From the comparison created on the graphs depending on numerous simulation results, it may be concluded that OMSS algorithm is carrying out nicely from MDS algorithm in terms of LIF on cube multiprocessor interconnection networks. The OMSS algorithm is more effective with higher degree of balancing and the network usage is economical. The algorithm shows significant performance when run on cube based multiprocessor systems.

REFERENCES

[1] A. Samad, M. Q. Rafiq, O. Farooq, "A Novel Algorithm For Fast Retrival Of Information From A Multiprocessor Server," in proceeding of 7th WSEAS International Conference on software engineering, parallel and distributed systems (SEPADS '08), University of Cambridge, UK, pp. 68-73, 2008.

[2] N. Preve, "Balanced Job scheduling Based on ant algorithm for Grid Network," International Journal of Grid and High Performance Computing; vol. 2, no. 1, pp. 34-50, 2010.

[3] M. Alam, A. Kumar., "A Comparative Study of Interconnection Network," International Journal of Computer Applications, vol. 127, no. 4, pp.37-43, 2015.

[4] A. Samad, J. Siddiqui, Z.A. Khan "Properties and Performance of Cube-based Multiprocessor Architectures," International journal of applied evolutionary computation, vol. 7, no.1, pp. 67-82, 2016.

[5] E. Dodonov, R. F. d. Mello, "A novel approach for distributed application scheduling based on prediction of communication events," Future Generation Computer Systems, vol. 26, pp. 740–752, 2010.

[6] Q. Kang, H. He, 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.

[7] N. Rajak, A. Dixit, R. Rajak, "Classification of list task scheduling algorithms: A short review paper," Journal of Industrial and Intelligent Information, vol. 2, no. 4, pp. 320-323, 2014.

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

[9] A. Chandra , P. Shenoy, "Hierarchical Scheduling for Symmetric Multiprocessors," IEEE Transactions On Parallel And Distributed Systems, vol. 19, no. 3, pp. 418-431, 2008..

[10] Y. Saad, M. H. Schultz, "Topological properties of hypercubes," IEEE Trans. Computer, vol. 37 no. 7, pp. 867–872, 1988.

[11] El. A. Amway, S. Latifi, "Properties and performance of folded hypercubes," IEEE Transactions on Parallel and Distributed Systems, vol. 2, no.1, pp. 31– 42, 1991.

[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] N. Adhikari, C. R. Tripathy, "Star Crossed Cube: an alternative to star graph," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 22, no.3, pp.719-734, 2014.

[14] Z. A. Khan , J. Siddiqui , A. Samad, "A novel multiprocessor architecture for massively parallel system," Proceeding of the 2014 IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 68-73, 2014.

[15] M. Shanmugasundaram, R. Kumar , K. H. Mallikarjun , " Approaches for Transient Fault Tolerance in Multiprocessor- A State of Art," Indian Journal of Science and Technology, vol. 8, no. 15, pp. 1-9, 2015.

[16] K. Lakshmanan, D. D. Niz, R. Rajkumar, "Coordinated Task Scheduling, Allocation and Synchronization on Multiprocessors," In Proceeding of 30th IEEE Real-Time Systems Symposium, pp. 469-478, 2009.

[17] Z. A. Khan, J. Siddiqui, A. Samad, "A novel task scheduling algorithm for parallel system," In Proceeding of 3rd International conference computing for sustaniable global development, pp. 3983-3986, 2016.