ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 10(25), DOI: 10.17485/ijst/2017/v10i25/105344, July 2017* ISSN (Online) : 0974-5645

Dynamic Scheduling Algorithm for Variants of Hypercube Interconnection Networks

Zaki Ahmad Khan[1*], Jamshed Siddiqui[2] and Mahfooz Alam[3]

[1]College of Life Science Nanjing Agricultural University Nanjing, Jiangsu, China; t2016041@njau.edu.cn
[2]Department of Computer Science, Aligarh Muslim University, Aligarh – 202001, Uttar Pradesh, India; Jamshed_faiza@rediffmail.com
[3]Department of Computer Science, Al- Barkaat College of Graduate Studies, Aligarh – 202002, Uttar Pradesh, India; mahfoozalam.amu@gmail.com

Abstract

Objectives: Topropose analgorithm for better performance interms of scheduling and the network usage is economical. **Method/Statistical Analysis:** The dynamic task scheduling algorithm has been proposed for scheduling the load on numerous cube based multiprocessor interconnection networks. Especially the efficiency of the proposed algorithm is examined in terms of performance parameters for instance Load Imbalance Factor's as well as Execution Time for cube based multiprocessor networks; Nevertheless, a comparison is created with other standard scheduling algorithm. **Findings:**The comparative simulation study shows that the proposed algorithm gives better performance in terms of task scheduling on various cube based multiprocessor networks. **Application:** The study in such a direction implies that the variety of processors in folded hypercube has been decreased thus minimizing the cost as well as intricacy of the network without reducing the efficiency of the network. Therefore, a combination of scalable folded hypercube architecture and efficient proposed algorithm is a better organization model that supports variety of informatics applications.

Keywords: Dynamic Scheduling Algorithm, Hypercube Networks, Interconnection Networks, Minimum Distance Property, Scheduling Performance Parameter

1. Introduction

Numerous scheduling schemes were created that are made to attain their aims for instance economical use of process elements, cutback of source idleness or reducing the overall execution time. Certain strategies are particular to a specific kind of multiprocessor architecture. Those methods are introduced utilizing various techniques for example Minimum Distance Strategy (MDS)[1], Hierarchical Balancing Method (HBM)[2] etc. There are algorithms which use as well as enhance the task scheduling based on the forecast of procedure conduct. Most of these algorithms deem the procedure conduct extraction, category and prediction[3]. Iterative greedy strategy can be a vital algorithm to reduce the whole execution time and communication cost[4]. The primary concept within this algorithm is always to enrich the top notch of the assignment in an iterative fashion employing outcomes from earlier iteration[5]. All these algorithms are employed on particular parallel system along with the performance has not been widely analyzed on a cube kind of multiprocessor system. This paper is centered on examine the scheduling issue on a cube multiprocessor architecture[6-7]. The regular dynamic MDS algorithms are intended initially for cube dependent multiprocessor networks. Simulation outcomes are examined. The comparison study based on numerous performance parameters is performed on the outcomes received by the algorithms.

The number of topology of the interconnection network is vital in the layout of substantially parallel computer systems. With in this paper top three cubes based multiprocessor interconnection networks are thought meant for simulation. Furthermore the performance is usually examined for standard hypercube (HC)[8], Folded Hypercube (FHC)[9] as well as Cross Cube

* *Author for correspondence*

(CQ)[10-11-12] architecture and a comparative examine is created. The vital properties of such interconnection networks are provided in Table 1.

Table 1. Comparison of Cube Interconnection Network

Parameter	HC	FHC	CQ
Nodes	2^n	2^n	2^n
Diameter	n	n+1	N
Degree	n	n/2	⌈n+1/2⌉
Cost	n^2	n/2*n+1	n ⌈(n+1)/2⌉

2. Dynamic Scheduling Models

It presumes an effective problem characterization through which the load is partitioned into numerous tasks required for simulation. Every task is often regime or maybe partitioned aspects of an individual program. Virtually all the tasks are neutral to implement practically in any processor at a series. The scheduling performance of the method remains examined on the three distinct networks by simulating imitation dynamic load. In order to simulate the load on the given networks, it is epitomized into two groups of task structures uniform and non-uniformload[4-5]. 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 created in a deterministic fashion as a regular tree. Every node of the tree represents a task, and carried out in parallel in breadth-first influencing via the root task which is earmarked to particular provided nodes of the network. The amount of nodes in the task tree at level shows a specific phase of the load. To manage depict non-uniform load (non-deterministic load), the total problem is invented to be an arbitrary tree which relax by itself level by level[13-14-15]. A task scheduled on a processor spawns an arbitrary or random number of subtasks, which are part of the whole problem tree[16].Thus the load on each processor is varying at run time creating unbalance, and balancer/scheduler has to be invoked after each stage[17-18].

The version of parallel system whereby the task assignment is performed includes couple of completely linked processors or nodes. One can find no precedence interactions between tasks as well as any task tend to be executed cost. The total cost relies up on the mapping of application as well as the communication cost sustained in the network. In the proposed algorithm the tasks are created in a kind of randomly on the processor. We consider the pattern of structure namely purely random task structure.

3. Proposed Dynamic Scheduling Algorithm

The performance of a multiprocessor system may be identified by communication holdup, syndication of load among the processors as well as scheduling over-head. There are plenty of algorithms that are influenced by the theory of minimum distance characteristic. Minimum distance is the character which guarantees the minimization of the communication in distributing subtasks and gathering narrow outcomes. A scheduling algorithm works using this property for instance Minimum Distance Scheduling (MDS) reduces over-head as well as confirms the optimum achievable speedup, however, at the cost of idle unconnected node(s). The Dynamic Task Scheduling (DTS) Algorithm 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). The IL is figured out by summing the load of every processor in the network separated by the amount of processors obtainable 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 under loaded processors. In other words the overloaded (donors) and under loaded (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 under loaded 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.

The proposed Dynamic Task Scheduling (DTS) 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 instantly moods the communication cost. To be able to decrease 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 Dynamic Task Scheduling (DTS) 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 technique assists to manage the communication cost as well as over-head on the scheduler. In order to render simulation, the tasks are generated incrementally at various stages of task structure. Each stage represents a particular state of the task structure which consists of finite number of tasks.

The Load Imbalance Factor to z^{th} stage, represented as LIFz,

$$LIFz = [\{Lz (Pi)\} Max - (IL) z] / (IL)z \quad (1)$$

Where,

$$(IL)z = [Lz (P0) + Lz (P1) + \ldots + Lz (PN-1)] / N, \quad (2)$$

A pseudo code of the algorithm is shown in Table 2.

Table 2. The Pseudo Code of Proposed Dynamic Scheduling Algorithm

```
Algorithm: DTS
Proposed Algorithm ( )
/* Theprocessor i with processor j. Assume the level of
connectivity is given (multiple level)*/
Int connected (inti, int j, int level) /* returns true if
processors i, j are connected */
{
If (level = = 1)
Returnadj [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 the effectiveness of the Dynamic Task Scheduling (DTS), the simulation operate includes numerous kinds of load as well as mapping these on the cube networks for instance Hypercube (HC), Folded Hypercube (FC) and Cross Cube (CQ). The approximation of LIF is received for several amounts of tasks and the task migration from one node to another node is the form of packets namely one, four and eight. The DTS and Minimum Distance property are directly connected processors for migration.

Once MDS algorithm is executed on the cube based network the task are slated with simply haphazard task structures. The mapping of task is conducted at numerous stages of task structures demonstrated in the shape provided in Figure 1. It demonstrates that values of LIF initially start minimizing with the rise in variety of tasks. Even so, at many more tasks the scheduler unable to map the task efficiently and hence LIF value becomes higher.

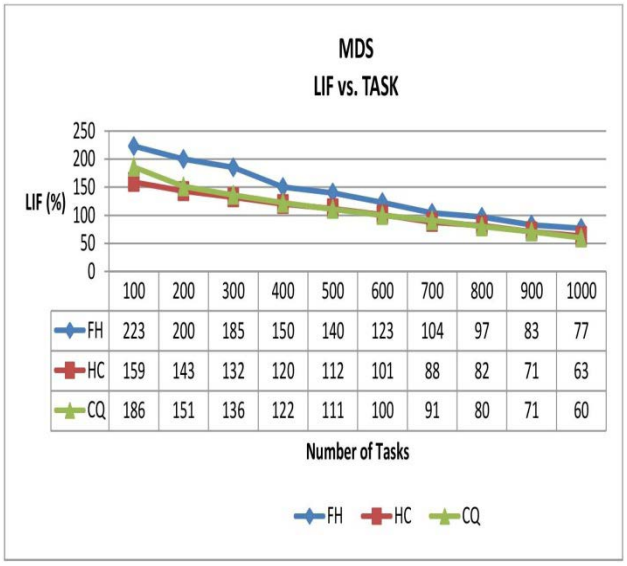

Figure 1. MDS Algorithm on Cube Based Network.

It is observed that the proposed algorithm producing similar results in cube based networks. The time varies on MDS and Other Scheduling Scheme on HC, FHC and CQ networks. The time is continuously reducing and become one at one thousand tasks. However, MDS Scheme shows the lesser balancing time. This trend is depicted in Figure 2.

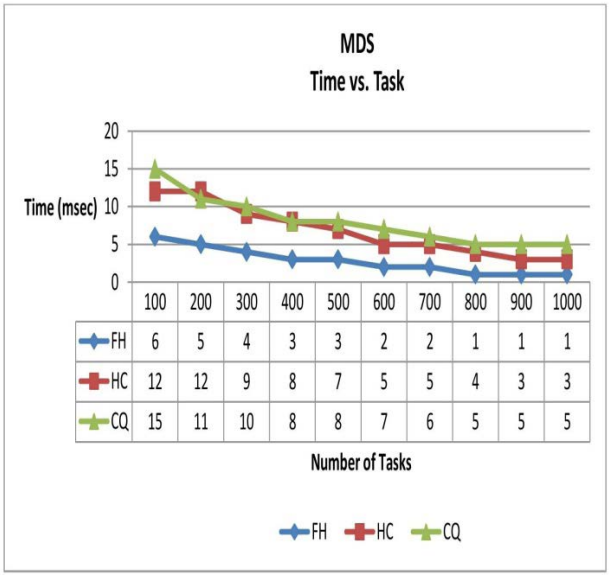

Figure 2. Time Graph of MDS Algorithm on Cube Based Networks.

When comparing the simulation results it is observed that the proposed Dynamic Task Scheduling (DTS) algorithm producing similar results in cube based networks. At higher levels of task structures the LIF is increasing as well as tasks are not mapped efficiently. When the numbers of tasks are increasing then initial value of LIF is lesser as well as reducing. This trend is depicted in Figure 3.

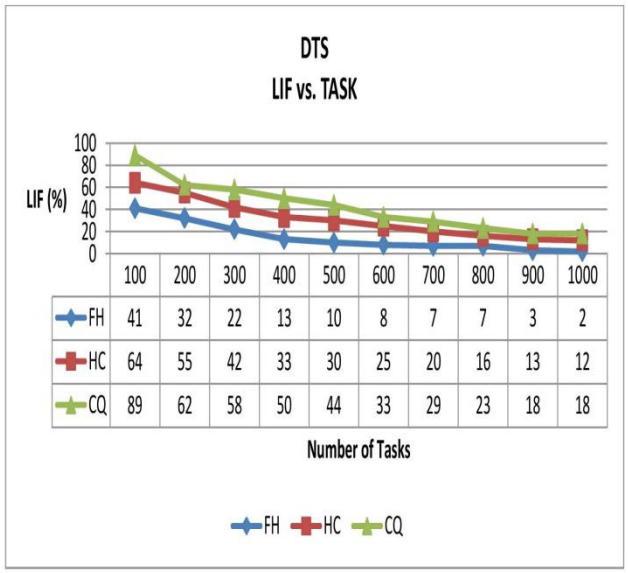

Figure 3. Proposed the Dynamic Task Scheduling Algorithm on Cube Based Networks.

The performance results indicate the behavior of balancing time of DTS algorithm Time verses Load (Figure 4). It is observed from the curves that there is regular pattern in the balancing time with load. The tendencies of the tasks are irregular; consequently, the balancing instance differs on DTS on the cube network. The total execution time of DTS first begins minimizing with the rise in number of tasks. The time is continuously reducing and become one at one thousand tasks. However, DTS Scheme shows the lesser balancing time.

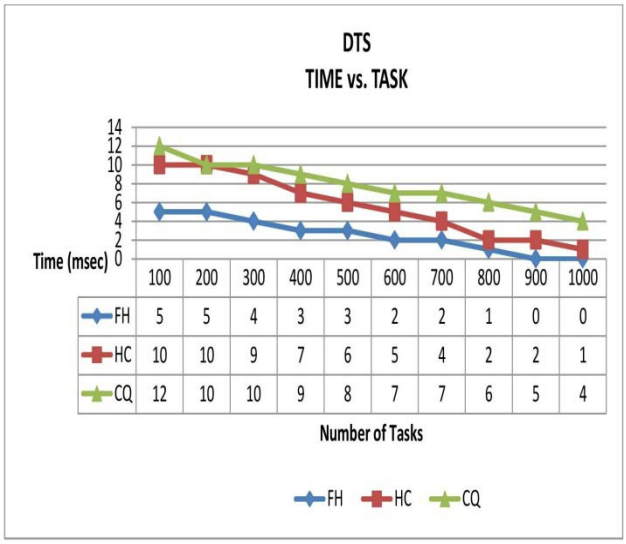

Figure 4. Time Graph of DTS Proposed Dynamic scheduling on Cube Based Networks.

The comparison made from the graphs based on various simulation results, it may be concluded that DTS 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, this can be concluded that DTS scheme and FHC network is the better organization. The organization is found to be performing better particularly for unpredictable load.

The overall performance of the DTS Scheme is highly dependent on the connectivity of the various processors accessible in the network. Nevertheless, the algorithm allows for the tasks to offered processors in the network if these are linked directly as well as partially to indirectly linked nodes the network whether they are connected directly and partially to indirectly connected nodes. The DTS scheme is better, degree of balancing is higher and

the network utilization is efficient and is ideally suited for linear multiprocessor networks.

5. Conclusions

We proposed a new scheduling algorithm and applied on various cube type multi processor interconnection networks in terms of load imbalance left after a balancing action and precious time. The efficiency of the DTS algorithm is really based on the connection of the several processors included in the network. Even so, the algorithm permits tasks to the offered nodes in the network whether these are linked instantly and also partially to in a round about way linked processors. From the comparison created on the graphs depending on numerous simulation results, it may be concluded that DTS algorithm is carrying out nicely from MDS algorithm in terms of LIF on cube multiprocessor interconnection networks. The DTS algorithm is more effective, the degree of balancing is greater and the network usage is economical.

6. References

1. Khan ZA, Siddiqui J, Samad A. A novel multiprocessor architecture for massively parallel system. Proceeding of the 2014 IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC), India. 2014; p.68–73. Crossref
2. Dodonov E, MelloRFd. A novel approach for distributed application scheduling based on prediction of communication events. Future Generation Computer Systems. 2010; 26(3): 740–52. https://doi.org/10.1016/j.future.2009.05.004
3. Umarani GS, Uma VM, Shanthi AP, Siromoney A. Task Scheduling Model. Indian Journal of Science and Technology. 2015; 8(S7):33–42. Crossref
4. Kang Q, He H, Song H. Task assignment in heterogeneous computing systems using an effective iterated greedy algorithm. The Journal of Systems and Software. 2011; 84(2):985–92. Crossref
5. Shanmugasundaram M, Kumar R, Mallikarjun KH. Approaches for transient fault tolerance in multiprocessor-A State of Art. Indian Journal of Science and Technology. 2015; 8(15): 1–9. Crossref
6. Rajak N, Dixit A, Rajak R. Classification of list task scheduling algorithms: A short review paper. Journal of Industrial and Intelligent Information. 2014; 2(4): 320–23. Crossref
7. Preve N. Balanced Job scheduling based on ant algorithm for grid network. International Journal of Grid and High Performance Computing. 2010; 2(1): 34–50. Crossref
8. Samad A, Siddiqui J, Khan ZA. Task allocation on linearly extensible multiprocessor system. International Journal of Applied Information Systems. 2016; 10(5):1–5. Crossref
9. Martelli F, Bonuccelli MA. Minimum message waiting time scheduling in distributed systems. IEEE Transactions on Parallel and Distributed Systems. 2013; 24(9):1797–1806. Crossref
10. Alam M, Kumar A. A comparative study of interconnection network. International Journal of Computer Applications. 2015; 127(4):37–43.
11. Saad Y, Schultz MH. Topological properties of hypercubes. IEEE Trans. Computer. 1988; 37(7):867–72. Crossref
12. Amway ElA, Latifi S. Properties and performance of folded hypercubes. IEEE Transactions on Parallel and Distributed Systems. 1991;2(1):31–42. Crossref
13. Efe K. The crossed cube architecture for parallel computation. IEEE Transactions on Parallel and Distributed Systems. 1992; 3(5): 513–24. Crossref
14. Adhikari N, Tripathy CR. Star crossed cube: an alternative to star graph. Turkish Journal of Electrical Engineering and Computer Sciences. 2014; 22(3):719–34. Crossref
15. Samad A, Siddiqui J, Khan ZA. Properties and performance of cube-based multiprocessor architectures. International journal of applied evolutionary computation. 2016; 7 (1): 67–82. Crossref
16. Tandjaoui D, Doudou M. FH-MaC. A Multi-Channel hybrid MaC protocol for wireless mesh networks. International Journal of Grid and High Performance Computing. 2009; 1(4):40–56. Crossref
17. Khan ZA, Siddiqui J, Samad A. A novel task scheduling algorithm for parallel system. Proceedings of 3rd International Conference on Computing for Sustainable Global Development, INDIACom. India, 2016;p. 3983–86.
18. Bokhari MU, Alam M, Hasan F. Performance analysis of dynamic load balancing algorithm for multiprocessor interconnection network. Perspectives in Science. 2016; 8:564–66. Crossref