Load Balancing Problem on Hyper Hexa Cell Interconnection Network

Aryaf Al-Adwan[1]
Department of Computer and Networks Engineering
Faculty of Engineering Technology
Al-Balqa Applied University
Amman, Jordan

Basel A. Mahafzah[2]
Computer Science Department
King Abdullah II School for Information Technology
The University of Jordan
Amman, Jordan

An'aam Aladwan[3]
Department of Management Information Systems
Al-Ahliyya Amman University
Amman, Jordan

***Abstract*—Dynamic load balancing techniques prevents computer nodes from overloading unevenly while leaving other idle. It is considered as one of the most challenging topics in parallel computing. Moreover, it is essential for increasing the efficiency of highly parallel systems especially in solving multitask problems with unpredictable load estimates. Particularly, over each processor in the parallel systems and interconnection networks. This paper focuses on developing an efficient algorithm for load balancing on Hyper Hexa Cell (HHC) interconnection network, namely, HHCLB algorithm. Basically, the Dimension Exchange Method (DEM) approach is used in this paper to construct a new load balancing approach on the network of HHC interconnections. Thus, an algorithm was introduced and simulated using java threads, where the performance of the algorithm is evaluated both analytically and experimentally. The evaluation was in terms of various performance metrics, including, execution time, load balancing accuracy, communication cost. By implementing the proposed load balancing algorithm to the HHC network, a high degree of accuracy and minimal execution time was achieved. It is important to highlight that the algorithm recorded small gap between the execution time for small number of processors and large number of processors. For instance, the algorithm achieved 0.14 seconds for balancing the load of 6 processors while 0.59 seconds for balancing the load of 3072 processors. This proves how effective the algorithm is in balancing the load for different network sizes from small to large number of processors, with a slight difference in execution time.**

Keywords—Parallel computing; load balancing; Hyper Hexa-Cell; interconnection network; Dimension Exchange Method (DEM)

I. Introduction

In parallel systems, during the processing of tasks, the load is dynamically modified. So, during the processing of each task, there is a need to consider the current node of each processor. The literature has suggested many techniques and methodologies for scheduling processes in a distributed or parallel environment. Load balancing is primarily aimed at equalizing the load between the nodes by minimizing execution time and communication delays, optimizing resource efficiency and maximizing throughput [1][2].

To prevent some processors from becoming idle when others have a significant amount of workload, load balancing is concerned with distributing the workload among the processors in a parallel system. Thus, either heavily loaded processors sending loads to other processors or idle processors demanding work from others will perform it. It is critical that a large number of communication steps do not significantly contribute to the overhead of executing the load balancing algorithm.

In any parallel machine, the interconnection network transfers information from the source processor to the destination processor. This task can be achieved with as little latency as possible, enabling a large number of such transfers to occur at the same time. Therefore, in reducing this latency, the topology of the interconnection network plays a significant role. Correspondingly, researchers in this field have proposed many interconnection networks and various parallel algorithms have been used to verify the topological properties of such architectures. One of these interconnection networks is Hyper Hexa Cell, which was built on the advantageous features of the hypercube interconnection network and had attractive topological properties such as diameter, minimum node degree, width of bisection and optical cost [3]. This encouraged the researchers to use this new architecture in solving many parallel algorithms in different fields, such as broadcast communication [4], unicast routing [5] and parallel prefix sum [6]. Moreover the optical links version of this interconnection network namely OTIS-HHC, was exploited by the researchers in order to solve various problems, such as communication algorithms[7], shortest path routing on OTIS-HHC[8], parallel heuristic local search algorithms [9], routing and sorting algorithms[10], parallel quick sort algorithm [11], and traveling salesman problem[12]. Due to the importance of load balancing in parallel architectures, different researches were addressed this problem [13-22]. Among these researches, HHC obtained high speedup as well as efficiency, this is due to the iterative structure that is provided by this interconnection network as well as the high computing capabilities and the minimum communication time that can be achieved by HHC. Therefore, it is worth to solve the load balancing problem on such interconnection network. Based on these observations this paper chose the HHC interconnection network.

This paper focuses on developing an efficient algorithm for load balancing on hyper Hexa Cell interconnection network where a new algorithm is introduced and its performance is evaluated analytically and experimentally, in terms of various performance metrics. To the best of our knowledge, there is no work that has been addressed in the literature to solve the load balancing problem on Hyper Hexa Cell interconnection network, since it is relatively a new optoelectronic architecture.

The structure of this study is as follows: Section 2 shows a brief background about the main concepts that are used throughout this study, Section 3 describes the proposed algorithms, Section 4 describes the analytical evaluation for the two algorithms, Section 5 shows the experimental results and the comparison analysis of our algorithms, and finally the conclusions and future work are presented in Section 6.

II. Background

Hyper Hexa-cell (HHC) which has a dh dimension, can be considered as an undirected graph that can be generated by substituting 2^d nodes of a hypercube by HHC graph with one dimension [3]. A fusion of an HHC of dimension one and a hypercube of dimension d is each dimension of the HHC. One is the smallest dimension of an HHC and it is the basis for the other HHC dimensions. For more clarity, HHC of six processors and their labels is as shown in Fig. 1.

A summary of the topological characteristics of the HHC interconnection network in terms of diameter, maximum and minimum node degree, size, total cost and bisection width is provided in this section, as defined in [3]. These topological properties include the diameter, maximum node degree, size, number of links and bisection width. The diameter is the most important property that distinguish each interconnection from the other, it can be defined by the maximal distance between any two processors in the network is the diameter. The diameter of the HHC interconnection network is d+1, where the maximum distance between one of the top triangle's processors and one of the processors at the bottom of the opposite triangle will always be two steps in each sub-group. For instance, the diameter would be 1 + 1 = 2 in the first dimension, and 2 + 1 = 3 in the second dimension.

The maximum node degree of interconnection network can be defined by the maximum number of links to which it is connected. Consequently, the maximum node degree of the HHC interconnection network is d +2, where each node within each HHC subgroup is connected to three nodes. Also, each node is connected to an equivalent node in an additional dimension by a single link. The maximum node degree, for example, is 2 + 2 = 4 in the second dimension.

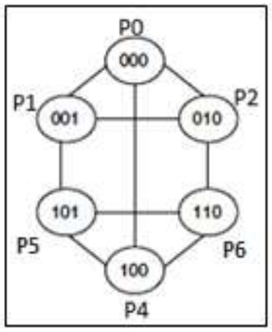

Fig. 1. One Dimensional HHC [3].

Basically, the size of an interconnection network can be considered as the number of processors or nodes in any interconnection network. The size of the HHC interconnection network is $6{\times}2^{(d-1)}$, because in HHC interconnection network, the minimum number of nodes is six. For example, the number of processors in the first dimension will be $6 \times (2^0) = 6$ processors and in the second dimension will be $6 \times (2^1) = 12$ processors.

The total cost is the total number of communication links that bind the processors within each network group, which can be calculated easily using the following equation:

$$\text{Number of links} = ((6 \times 2^{d-1}) \times (d + 2)) / 2 \quad (1)$$

Precisely, the bisection width is the minimum number of communication links to be eliminated in order to break the network into two equal halves, as defined in equation 2. For example, in the second dimension the cost of HHC will be $(6\times 2^1) / 2 = 6$.

$$\text{Bisection width} = ((6 \times 2^{d-1}) / 2 \quad (2)$$

III. Dimension Exchange Method

Balancing is carried out between two processors in the DEM system in such a way that the processor with a greater load sends part of its load to the other processor, so that it has the same load as possible. If the load is indefinitely divisible, exactly the same amount of load will always be taken from each processor. But this is not a realistic assumption of fact in the distribution of tasks [15-16]. By redistributing the tasks through the links of each dimension, this approach goes across all dimensions, from d=1 to d = n, and balances loads. The processors swap their load sizes in the first dimension, where the higher load processor transfers the excess load to the lower loaded processor; in the second , third and fourth dimensions, the same steps are performed between the processors related. Basically, the DEM approach is used in this paper to construct a new load balancing approach on the network of HHC interconnections.

IV. Hyper Hexa Cell Load Balancing Algorithm

In this section , the Hyper Hexa Cell Load Balancing (HHCLB) algorithm is introduced, where the pseudo code for balancing the one-dimensional HHC interconnection network is provided, followed by the application of the DEM algorithm to balance the multidimensional HHC, and the best and worst cases of the algorithm are finally defined.

A. HHCLB Pseudo Code

The pseudo code of HHCLB basically contains two phases as follows:

Input: Unbalanced P-processors Hyper Hexa Cell

Output: Balanced HHC with equal or approximate load on each processor

```
Phase 1: Processors Load balancing
1.   For (j=1; j<=p-1; j++)
2.   for all pairs of processors pi,pj such that the binary
     representation of i and j differ only in the dth bit
     position
3.     do in parallel
4.   avgload = floor (load(pi)+load(pj) ) / 2
5.     if(load(pi)>avgload)
6.   send excess load (load (pi)-avgload) to neighbor
     processor along dimension d
7.   load (pi) = avgload
8.   else
9.   receive excess load (load (pi)-avgload) from neighbor
     processor
10.  load (pi)+= ( load(pj) –avgload )
Phase 2: Triangle Load Balancing
11.  Consider the HHC as two triangles and balance each
     one on parallel
12.  Exchange pi,pj,pk loads sizes
13.  avgload = floor (load(pi)+load(pj)+ load(pk) ) / 3
14.  if(load(pj)>avgload)
15.  pj send excess load (load (pj)-avgload) to pi and pk
     processors
16.    load (pj) = avgload
17.  else
18.  pj receive excess load (load (pi)-avgload and load (pk)-
     avgload) from pi and pk processors
19.    load (pi) = avgload
20.    load (pk) = avgload
```

The main phases of this algorithm are as follows:

Phase1: Perform load balancing on all HHC processors that differs only in the location of the dth bit. It includes the steps that follow:

Step 1: the load will be exchanged between the processors that will communicate to balance their loads. So, load balancing will be applied between the processors who differ in the Least Significant Bit (LSB) as depicted in Fig. 2(a) and (b). Then, the Second Significant Bit (SSB) as shown in Fig. 2(c) and (d). Finally, the Most Significant Bit (MSB) as shown in Fig. 2(e) and (f) (Line 2).

Step2: For each pair of processors directly linked along the dth dimension, the average load is computed. This can be accomplished by considering the floor of the sum of processor's load and dividing it by two (line 4). For example, adding and dividing the load of processors P0 and P1 in LSB step by two and calculate the average load of them as: [LP0+LP1/2]. Also, the load of processor P5 and P4 in LSB step will be added and divided by two to calculate the average load of them [L_{P5}+L_{P4}/2]. Notice that, we assume that if the average load contains fractions, the remainder , which equals to one, will be added to the AvgLoad of the processors as follows: between P0, P1 → P1, between P5, P4 →P5, between P1, P2 →P1, between P5, P6 →P5 in the LSB Step.

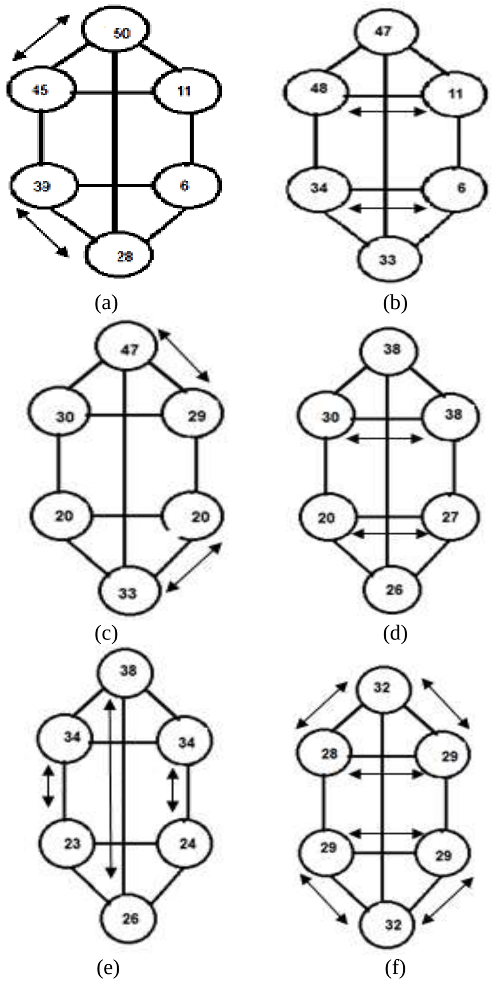

Fig. 2. HHC One Dimensional Load Balancing Example.

Step3: Processors' load redistribution. First, each processor compares its load to the average load (average of the processor's load and its neighbor processor's load). If the processor's load is greater than the average load (line 5), the processor sends the amount of excess load (processor's load minus average load) along the dth dimension (line 6), and the processor's load is adjusted to be equal to the average load (line 7). Otherwise, the processor receives the amount of its neighbor's excess load along the dth dimension (lines 9-10), and its load is incremented by the amount of its neighbor's excess load.

When the algorithm balances the loads for the processors based on the difference in the least significant bit the remainder will be added to the processors with labels (P1 and P5). When the algorithm balances the processors based on the difference in the second significant bit the remainder will be added to the processors with labels (P2 and P6). In the most significant bit balancing the remainder will be added the processor with the lower load. After this phase you will have a balanced rectangle of processors P1, P2, P5, and P6 with maximum difference in weight = 2). And the loads of processors P0 and P4 will be either less than or greater than the loads of the processors in this rectangle.

Phase 2: Triangle Load Balancing

In this phase, the HHC will be considered as two triangles and the average processor's load in each triangle is calculated (lines 11-14). There will be two cases here: *Case 1* (in lines 15-16), in which the load of processor P0 is greater than the load of processors P1 and P2, in this case, processor P0 will send the excess load to both processor P1 and P2, as depicted in Fig. 2(f).

Case 2 (in lines 17-20), the load of processor P0 are less than the load of processors with labels P1 and P2, in this case processor P0 will receive the excess load from both P1 processor and P2 processor. This will be applied in parallel with processors P4, P5, and P6. Finally, after applying the two phases a balanced one-dimensional HHC will be obtained, as depicted in Fig. 3.

Regarding the balancing of the other dimensions, where *d >1*, the balancing will be started by applying the DEM algorithm [17-18] on the connected HHC cells. In the two-dimensional HHC, two steps are only needed for load balancing. The first, is to balance the HHC cells in the first dimension while in the second step each processor will be balanced with its directly connected neighbor in the second dimension.

B. Best and Worst Cases of HHCLB Algorithm

The best case occurred when the HHC interconnection is almost balanced; in this case, the algorithm will perform only global information collection and average calculation without load balancing steps, because each processor will find out that its load is equal or almost equal to the average load between them. On the other hand, the worst case occurred when all the workload is on one processor P1 while other processors are idle or have very minimum load as shown in Fig. 4.

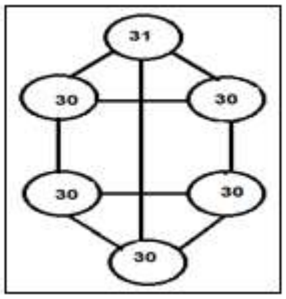

Fig. 3. Balanced HHC.

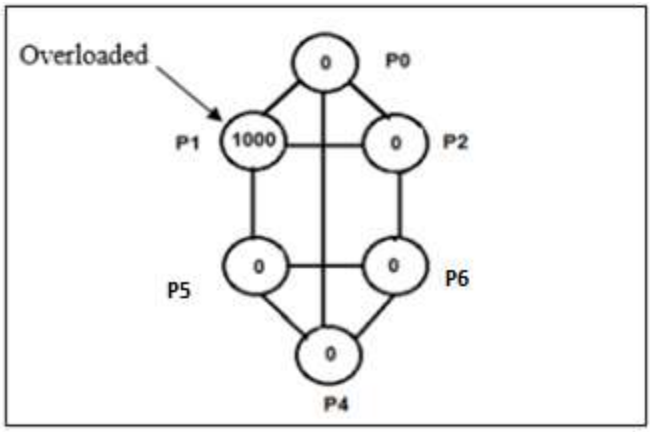

Fig. 4. Worst case of HHCLB Algorithm.

If each processor is assigned at most *M* tasks before the load balancing algorithm is performed, then at each step there are at most *M/2* tasks to be moved across the edges of dimension.

Assuming the maximum workload is *M*. In phase 1, the first step of the algorithm is required to transfer *M/2* excess load from *P1* to *P0*. In step 2, *M/4* load will be transferred from *P1* to *P2*. While in step 3, *M/8* load will be transferred from *P0* to *P2*. In step 4, *M/16* load will be transferred from *P1* to *P2*. In step 5, *3M/16* load will be transferred from *P0* to *P4*. Subsequently, in phase*2*, *M/48* excess load will be transferred by the algorithm. Adding all these loads would result in a first dimension execution time of 1.2M. This will end with every load of M/6 on each processor. On the second dimensions, M/6 will be divided by 2 on the second dimension, resulting in an excess load of M/12 that will be transferred between the two HHC cells. And so forth. Thus, in all dimensions, the total execution time will be as shown in equation (3).

Solving equation (3) results in *O (M + ln (d))* execution time.

$$\text{Total execution time} = 1.2\,M + \sum_{i=2}^{d} \frac{M}{6*i} \quad (3)$$

V. Analytical Evaluation

HHCLB algorithm is evaluated in this section in terms of the following metrics: execution time, accuracy of load balancing, and number of communication steps.

A. *Execution Time for HHCLB*

The execution time metric calculates the time required to achieve load balancing steps. The worst-case time complexity of the proposed load balancing algorithm on HHC, is the time taken to move the excess load M from one processor to another, which can be defined as the difference between the maximum load of one processor and the average load of the processor. When all processor loads are zero and processor P1 has the maximum load, the worst case of HHCLB occurred, this will generate O (M + ln d), as discussed in section IV.

B. *Load Balancing Accuracy*

The accuracy in load balancing represents the difference between the maximum and the minimum number of tasks assigned to any processor in the whole interconnection which also defined as the error rate in the algorithm [14].

In the first dimension, HHCLB algorithm generates a maximum error equal to 2, and in the above dimensions it will achieve error with maximum $e \leqslant d+1$.

C. *Number of Communication Steps*

Communication cost in load balancing is the number of communication steps that are needed for load balancing [14].

The number of communication steps in phase1 is five steps. But in each step, there are two more additional steps for load exchanging and average load calculation. Therefore, this will produce 5*3 = 15 steps. Additionally, in phase2, the number of communication steps will be 2*3 = 6 steps. So, in the first dimension, the total number of steps is 21. On the other hand, the maximum number of steps in all dimensions will be increased by three in each dimension, this will result in 21+3*(d-1) = 3d+18. In the worst case, we may have additional three steps over each dimension, Thus the total number of communication steps will be 6*(d-1) + 21 = 6d+15. The summary of the performance metrics is shown in Table I.

TABLE I. SUMMARY OF PERFORMANCE METRICS FOR EVALUATING HHCLB ALGORITHM

Analysis Metric	**Equation**
Execution Time in the First Dimension	*1.2M→ O(M)*
Total Execution Time	*O (M + ln (d))*
Maximum Number of Steps in all dimensions	*In average 3d+18* *In worst case 6d+15*
Accuracy	e ≤ d+1
Links utilized	All

VI. EXPERIMENTAL RESULTS AND COMPARATIVE ANALYSIS

The experimental results obtained to validate the performance of the proposed algorithms for load balancing in HHC interconnection networks are shown in this section. To implement the algorithm, the simulation environment was set up using the Java Jdk7.2 and Eclipse Java EE IDE environments. All tests were performed on a 16 GB RAM Intel Processor (CPU 3.2 GHz) with 8 MB Cache memory and Windows10 as an operating system.

The implementation was based on the following classes:

- **Topology** class, which connects the multidimensional interconnection of the HHC cells.
- **HHC** class, which according to the HHC interconnection, connects the processors.
- **Node** class, which sets each processor's properties.

The simulation begins by constructing the desired network of interconnections according to the user-determined dimension. To allow parallel execution of the implemented load balancing algorithms, the load balancing mechanism is implemented using multithreading. The library of Java threads is used to build and manage a complex number of threads used to simultaneously perform load balancing steps. Our implementation was done by: load computation, calculation of average load, and transfer of excess load.

A. *Execution Time*

Several experiments have been conducted to compute the time required to execute the proposed load balancing method, on 6, 12, 24, 48, 96, 192, 384, 768, 1536 and 3072-processor HHC. That indicates that the experiments were performed for several dimensions starting from dimension one up to dimension ten. The execution of the algorithm was done using the same random number sequences that represents the load of the processors using specific seeds for our random number generator. For this purpose, four seeds were selected: {1, 2, 8, and 12} to run the experiment. Finally, the results were recorded according to the execution time as shown in Table II.

Table II depicts the average execution time in seconds taken by HHCLB to balance the HHC interconnection in different dimensions, in this experiment, the maximum load which assigned to each processor was at most 200 workload units. A careful examination of this table shows that the execution time is increased as the number of processors increases too. For instance, the average execution time for balancing 24 processors is equal to 0.23 seconds, while it takes 0.56 seconds to balance 1536 processors. This shows the efficiency of the HHCLB algorithm, where a large number of processors only need a very limited amount of time to balance their loads.

The preceding discussion concerned with the effect of the size of the network on the execution time. Currently, it is the time to examine the contribution made by the number of workload units allocated to each processor. So, another experiment had been performed with variable average loads sizes assigned to each processor. The load sizes used are at most {10, 50, 100, 500} workload units assigned to each processor as shown in Table III.

Experiments have shown that the number of workloads units allocated to processors have a huge effect on the execution time for a large number of processors. This is more apparent with a greater number of processors. For example, the execution time for HHCLB algorithm with 200 workload units and 24 processors is 0.26 seconds while it is 1.31 seconds for 1000 workload units and 24 processors. As shown in Table III these findings were revealed.

TABLE II. AVERAGE EXECUTION TIME USING HHCLB ON SEVERAL DIMENSIONS OF HHC INTERCONNECTION NETWORK (MAXIMUM LOAD SIZE IS 200 WORKLOAD UNITS PER PROCESSOR)

No of Processors	HCCLB Algorithm Execution time (Sec)
6	0.14
12	0.19
24	0.25
48	0.29
96	0.35
192	0.39
384	0.46
768	0.53
1536	0.56
3072	0.59

TABLE III. AVERAGE EXECUTION TIME IN SECONDS USING HHCLB ALGORITHM ON SEVERAL DIMENSIONS OF HHC (WITH 10,200,400 AND 1000 WORKLOAD UNITS)

Workload Units	HHC Dimension	No of Processors	HCCLB Algorithm Execution time (Sec)
10	**1D**	**6**	0.01
	2D	**12**	0.015
	3D	**24**	0.019
	4D	**48**	0.023
	5D	**96**	0.026
200	**1D**	**6**	0.19
	2D	**12**	0.24
	3D	**24**	0.26
	4D	**48**	0.29
	5D	**96**	0.36
400	**1D**	**6**	0.29
	2D	**12**	0.41
	3D	**24**	0.54
	4D	**48**	0.59
	5D	**96**	0.63
1000	**1D**	**6**	0.77
	2D	**12**	0.93
	3D	**24**	1.31
	4D	**48**	1.46
	5D	**96**	1.72

B. *Communication Cost*

The average number of communication steps was computed among several runs of HCCLB algorithm on different HHC cells in several dimensions from dimension one to dimension ten.

Table IV shows the average number of communication steps required by the algorithm to balance the loads on HHC interconnection network. It is clearly shown that HHCLB algorithm requires small number of communication steps.

TABLE IV. MAXIMUM NUMBER OF STEPS USING HCCLB ALGORITHM ON SEVERAL DIMENSIONS OF HHC INTERCONNECTION NETWORK (MAXIMUM LOAD SIZE IS 200 WORKLOAD UNITS PER PROCESSOR)

No of Processors	HCCLB Algorithm Communication Steps
6	18
12	24
24	30
48	33
96	39
192	45
384	48
768	51
1536	54
3072	60

C. *Load Balancing Accuracy*

The variation between the maximum amount of workload units in each processor and the minimum amount of workload units in each other is known as the error which is responsible for determining the accuracy of the load balancing. The accuracy of HHCLB algorithm on different HHC cells in several dimensions from dimension one to dimension ten was computed. The experiments show that the error in the algorithm does not exceed $d+1$. For instance, the accuracy for 96 processors is equal to four which is less than the fifth dimension. In addition, Table V showed a very excellent performance, where the accuracy for a very large number of processors such as 3072 processors is only seven. Thus, when the network size becomes very large the error is very small for HHCLB algorithm.

Fig. 5 illustrates a slight increase in accuracy as the number of processors increases.

TABLE V. THE NUMBER OF ERRORS RESULTED AFTER EXECUTING HHCLB OVER DIFFERENT HHC DIMENSIONS

No of Processors	Dimension of HHC	HHCLB Algorithm Load Balancing Accuracy
6	**1D**	2
12	**2D**	3
24	**3D**	3
48	**4D**	3
96	**5D**	4
192	**6D**	5
384	**7D**	5
768	**8D**	5
1536	**9D**	6
3072	**10D**	7

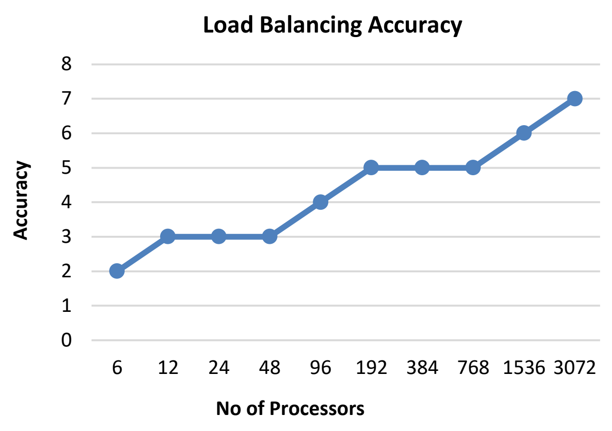

Fig. 5. Accuracy of HHCLB Algorithm.

VII. Conclusions and Future Work

We presented a new algorithm for balancing the load across the network of HHC interconnections in this paper. Moreover, it was evaluated and implemented. The efficacy of the HHCLB algorithm is proved not only by the parameters of the analytical evaluation, but also by the experimental findings. The results show the effectiveness of the HHCLB algorithm as regards to several performance metrics. Limited execution time was achieved, with a high degree of accuracy by applying the proposed load balancing approach to the HHC network. Consequently, with the time needed to perform load balancing on HHC, it was clear from the empirical and analytical results that the algorithm requires limited number of communication steps. So, the main goal of load balancing for equalizing the load among the processors with a minimization of the execution time and the communication delays is satisfied. Future work includes the extension of the proposed scheme for load balancing to be applied in optoelectronic architecture of the HHC interconnections.

References

[1] Willebeek-LeMair, H. Marc, and A. P. Reeves, “Strategies for dynamic load balancing on highly parallel computers”. Parallel and Distributed Systems, IEEE Trans, vol. 4, pp. 979-993, 1993.

[2] P. Shah and SM. Shah, “Load Balancing in Distributed System Using Genetic Algorithm”. Special issues on IP Multimedia Communications, vol. 1, pp. 139-142, October 2011.

[3] B. Mahafzah, A. Sleit, N. Hamad, E. Ahmad, T. and Abu-Kabeer. “The OTIS hyper hexa-cell optoelectronic architecture”. Computing, vol. 94, pp. 411–432, 2012.

[4] B. Mahafzah, and I. Al-Zoubi. "Broadcast communication operations for hyper hexa-cell interconnection network." Telecommunication Systems, vol. 67, pp. 73-93,2018.

[5] J. Al-Sadi. "A New Unicast Routing Algorithm for Hyper Hexa-Cell Interconnection Networks." International Journal of Information Systems and Social Change (IJISSC), vol. 8, pp. 45-57, 2017.

[6] A. Gupta, and B. Sarkar. "Parallel Prefix Sum Algorithm on Optoelectronic Biswapped Network Hyper Hexa-cell." International Journal of Computer Network & Information Security, vol. 10, pp. 27-35, 2018.

[7] A. Akhtar, K. Lucas, “Comparison of communication algorithms on OTIS-HHC and OTIS-ring parallel architectures”. International Journal of Engineering and Computer Science vol. 3, pp. 8741–8745,2014.

[8] A. Gupta, and B. Sarkar. "Shortest path routing on OTIS hyper hexa-cell." In Computing, Communication and Networking Technologies (ICCCNT), 8th International Conference on, pp. 1-6. IEEE, 2017.

[9] A. Al-Adwan, B. Mahafzah, and A. Sharieh, “ Parallel Heuristic Local Search Algorithm on OTIS Hyper Hexa-Cell and OTIS Mesh of Trees Optoelectronic Architectures”, Applied Intelligence, vol. 49, pp. 661-688, 2019.

[10] A. Akhtar, K. Lucas,” Routing and sorting on OTIS-hyper hexa-cell”, International Journal of Engineering & Computer Science **vol. 7**, pp. 7388–7393 ,2014.

[11] A. Al-Adwan, R. Zaghloul, B. Mahafzah, and A. Sharieh, "Parallel quicksort algorithm on OTIS hyper hexa-cell optoelectronic architecture”, Journal of Parallel and Distributed Computing, vol.141, pp. 61-73, 2020.

[12] A. Al-Adwan, R. Zaghloul, B. Mahafzah, and A. Sharieh ,” Solving traveling salesman problem using parallel repetitive nearest neighbor algorithm on OTIS-hypercube and OTIS-mesh optoelectronic architectures”. The Journal of Supercomputing vol. 74, pp. 1–36, 2018.

[13] A. Awwad, and J. Al-Sadi. "Efficient Load Balancing Algorithm for the Arrangement-Star Network." Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016.

[14] B. Mahafzah, B. and B. Jaradat, “The hybrid dynamic parallel scheduling algorithm for load balancing on Chained-Cubic Tree interconnection networks”. The Journal of Supercomputing, vol. 52, pp. 224-252, 2010.

[15] H. Rim, J. Jang and S. Kim, “An efficient dynamic load balancing using the dimension exchange method for balancing of quantized loads on hypercube multiprocessors”. In Parallel Processing, 1999. 13th International and 10th Symposium on Parallel and Distributed Processing, 1999. IPPS/SPDP. Proceedings IEEE, pp. 708-712, April 1999.

[16] G. Jan and Y. Hwang, “An efficient algorithm for perfect load balancing on hypercube multiprocessors”. The Journal of Supercomputing, vol. 25, pp. 5-15, 2003.

[17] H. Yuan-Shin H and E. Gene. “A Simple Algorithm for Optimal Load Balancing on Hypercube Multiprocessors”, The journal of supercomputing, vol. 25, pp. 5-15, 2001.

[18] B. Mahafzah and B. Jaradat, “The load balancing problem in OTIS-Hypercube interconnection networks”. The Journal of Supercomputing, vol. 46, pp. 276-297, 2008.

[19] B. Mahafzah, M. Alshraideh, L. Tahat, and N. Almasri,” Topological Properties Assessment for Hyper Hexa-Cell Interconnection Network”, International journal of computers, vol. 13,pp 115-121, 2019.

[20] J. Sadi, “Factor-Optical-Factor Factor Exchanges Method: a new load balancing method for Extended Optical Transpose Interconnection System-n-Cube networks”, Concurrency and Computation: Practice and Experience, vol. 13, pp.3415-3428,(2015).

[21] AM. Awwad, J. Al-Sadi, “Investigating the Distributed Load Balancing Approach for OTIS-Star Topology”, International Journal of Computer Science and Information Security, vol. 14, pp.163-171, 2016.

[22] AM. Awwad and J. Al-Sadi, J, “The Load Balancing Algorithm for the Star Interconnection Network”, International Journal of Computer, Information, Systems and Control Engineering, vol. 8, pp.1598-1602. 2015.