Performance Improvement in Multiprocessors using Two Row Boundary Allocation Method and Online Dynamic Compaction Algorithm

Akram Reza
Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University
Tehran, Iran

Mahnaz Rafie
Department of Computer Engineering, Ramhormoz Branch, Islamic Azad University
Ramhormoz, Iran

ABSTRACT
In this paper, two row boundary (TRB) allocation algorithm and limited top-down compaction (LT-DC) migration method are proposed. The first scheme, attempts to allocate the free nodes in the center of the mesh and decrease the problem of external fragmentation. The next mechanism use task migration to improve the performance of existing sub-mesh allocation strategies. It should be noted that in this process three key metrics are considered. They are average execution time, average response time, and average wait time. In fact, we perform rigorous simulation experiments based on practical workloads as reported in the literature to quantify all our proposed schemes and compare them against standard schemes existing in the literature. Based on the results, we make clear recommendations on the choice of the strategies.

General Terms
Network on Chip

Keywords
Allocation, Fragmentation, Migration, Two row boundary

1. INTRODUCTION
Network on chip (NoC) is a developing and promising on chip communication paradigm that improves scalability and performance of system on chips. NoC design flow contains many problems from different areas, for example networking, embedded design and computer architecture [1]. Application mapping is one of the most important dimensions in NoC research. It maps the cores of the application to the routers of the NoC topology, affecting the overall performance and power requirement of the system [2]. Furthermore, allocation is used for mapping development. Indeed, for optimal use of the computing power of a large multicomputer network, having a processor allocation algorithm is very vital. Processor allocation is responsible for selecting a set of processors in order to run parallel work on them. Also, minimization of allocation time in Grid multi-computers is a fundamental issue because the main purpose of parallel execution is to minimize the total time that a task spends upon the entry to the exit moment in the system. With increase in system size, time for finding sub-meshes for the allocation to input task may be equal to the task execution time. Hence, development of strategies for minimizing search time is very important. Methods of processor allocation can be divided into two general categories: continuous and discontinuous. In continuous allocation methods, a set of free continuous processors available in the network is allocated to execute the input task. Allocation method (as shown in [3]) results in high fragmentation. Excessive fragmentation degrades performance parameters of the system. In order to resolve the fragmentation that occurred in the continuous allocation, discontinuous allocation methods were proposed [4-8]. Discontinuous allocation is able to execute a task on several sub-meshes smaller than that the input task has requested and will not wait to release a continuous sub-mesh. It should be noted that a discontinuous allocation increases the conflicts between messages in the system. In fact, this strategy can create an overall traffic that leads to an increase in the message delay. In this paper, continuous allocation is considered. Also, task migration is used to solve external fragmentation of this type of allocation. Indeed, the continuous allocation algorithm has been designed for two-dimensional mesh network because it has always been popular owing to its simplicity, regularity and scalability.

In this article, for the online mapping the following steps have been done

The first step is to find the appropriate size of sub-mesh for input task. The second step is to find a sub-mesh place in integrating the mesh for online task allocation. In addition, the task migration has been continuously used to solve external fragmentation in allocation. The third step is to find a main place in sub-mesh for online task mapping. In order to reduce the overhead time of online mapping, second and third steps must be performed simultaneously. These steps will be discussed in the next sections. At first, previous studies related to the processor allocation algorithms in mesh networks will be reviewed. In a review of literature, studies conducted on improvement in efficiency of allocation and migration algorithms will be investigated and the manner of these algorithms performances will be summarized. In part 3, the proposed algorithm will be described and the manner of sub-mesh dimensions, allocation and migration of this algorithm will be exemplified. And finally, the results of the previous studies are compared from the viewpoint of several important parameters in performance.

2. REVIEW OF LITERATURE
Definitions and methods of continuous allocation and task migration used for multi-computers mesh networks have been reviewed in this section.

2.1 Definitions
A two-dimensional mesh M (w, h) is a rectangle of nodes with dimensions of w × h where w is width and h is the height of the rectangle. Each node of mesh is a processor that is known with the address of its characteristics [9]. A node in column c and row r has the coordinate of <c, r> where $\mathbf{0} \leq c < \boldsymbol{w}$ and $\mathbf{0}$

≤ r < ***h***. Node <i , *j*> that is not in borderlines of mesh approximates and connects directly with other four nodes: <*i* ± 1>, *j* and *i*, *j*±1 so that 0<*i*<*w*−1 and 0<*j*<*h*−1. In borderlines, each node approximates and connects to other two or three nodes according to its situation.

Definition 2-1-1: two-dimensional sub-mesh S (c, r) in the mesh M (w, h) is a sub-mesh M (c, r) that 0 ≤ c ≤ *w* and 0 ≤ r ≤ *h*. When a task requests a sub-mesh with dimensions c × r, this task is expressed via T (c, r). Address for sub-mesh S is known by its end and base node that is a four-parameters variable as <x_b , y_b , x_e , y_e> where, < x_b , y_b > shows the lower left corner and <x_e, y_e> shows the upper right corner of sub-mesh S. It is clear that c = $x_e - x_b + 1$ and r = $y_e - y_b + 1$ and base node of sub-mesh, is <x_b , y_b> and the sub-mesh area is the number of nodes inside it that is equal to c×r.

Definition 2-1-2: Busy sub-mesh β is a sub-mesh that all its nodes are assigned to a task at that moment. A set of busy sub-meshes B is the set that set includes all the busy sub-meshes available in the mesh that is called busy list. For example, in figure (1), three busy sub-meshes exist in the mesh M (6, 6); therefore, $B = \{\beta 1, \beta 2, \beta 3\}$ where $\beta 1$ = <4,0,5,2> , $\beta 2$ = <0,0,2,2>, $\beta 3$=< 0,3,1,4> are the members of this set.

*Definition 2-1-3:*Coverage sub-mesh for busy sub-mesh β is expressed according to the input T that is a sub-mesh that none of its nodes can be selected as the basis node of a free sub-mesh for allocation to task T with respect to busy sub-mesh $\vartheta_{\beta,T}$. Coverage sub-mesh $\vartheta_{\beta,T}$ is equal to <x_{cs} , y_{cs} , x_e ,y_e> for β<x_b, y_b, x_e, y_e> and the task β where, x_{cs} = max(0, x_b − c+1) and y_{cs} = max(0, y_b − r + 1). A according to the input task T, coverage set C_{ST} is a collection of coverage sub-meshes for the task T where, C_{ST}= {$\vartheta_{\beta,T}|\beta \in B$}. For example, for the input task T (2, 3) in figure (1), we have: $\vartheta_{\beta 1,T}$ = <3,0,5,2>، $\vartheta_{\beta 2,T}$ = <0,0,2,2>، $\vartheta_{\beta 3,T}$ = <0,1,1,4> ،C_{ST}= {<3,0,5,2> , <0,0,2,2> , <0,1,1,4>}

Definition 2-1-4: According to the input task T, reject δ_T sub-mesh is a sub-mesh including some processors that is a sub-mesh that none of its processors can be regarded as the basis node of a free sub-mesh for allocation to task T with respect to its dimensions. There are two reject sub-meshes for each T: horizontal (δ_{TH}) and (δ_{TV}) vertical. It is simple to calculate them i.e. δ_{TV} = <r′, 0, *w*, *h*> and δ_{TH} = <0, c′, *w*, *h*> and r′= *w* - c + 1 *and* c′= *h* - r + 1 where, *w* × *h* is sub-mesh size. A set of reject sub-meshes Δ_T is calculated by adding δ_{TH} and δ_{TV}. For example, δ_{TH} = <0,4,5,5> and δ_{TV} = <5,0,5,5> in figure (1).

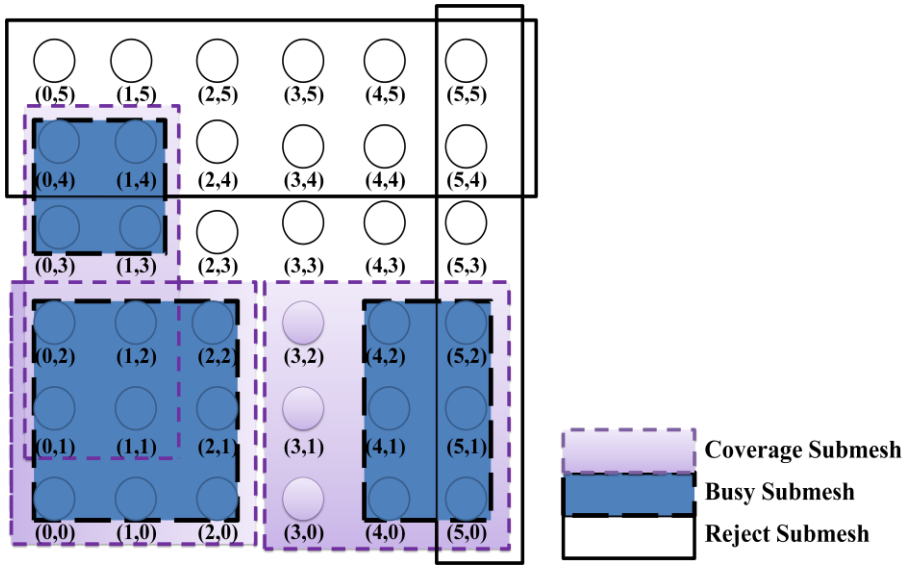

Figure 1 An example of allocation for T (2, 3)

2.2 Processor Allocation

Continuous allocation has been proposed for mesh multicomputer networks. In this approach, the tasks of an application are mapped onto a set of adjacent nodes which results in lower communication overhead. However, the main problem of this approach is the lower system utilization, since an application has to wait for a properly sized and shaped contiguous sub-mesh, while there may be sufficient number of free processors in non-contiguous regions. Scientists have extensively investigated contiguous allocation for two-dimensional (2D) mesh multicomputer [10–13]. In most previous studies, they have focused on reducing the effects of external fragmentation that are caused by the contiguous allocation strategies. Hereinafter, contiguous processor allocation schemes include a wide range of methods such as stack-based allocation [14, 15], adjacency allocation [15], adaptive scan allocation [12], and best/first fit allocation [13, 16].

2.2.1 Stack Based Allocation (SBA)

SBA has complete sub-mesh recognition ability by manipulating the orientation of each sub-mesh request. The key idea of this method is in the logic of finding a free sub-mesh for the request with a fixed orientation. Whereas, this logic is invoked twice at most, once for each orientation. It should be noted that implementing the candidate list as a stack for increasing the speed of search is the key idea of this algorithm. It has better speed and efficiency compared to the other methods. Time complexity of this algorithm is $O(N_a^2)$, where N_a is the number of busy sub-meshes [14].

2.2.2 Improved Stack Based Allocation (ISBA)

ISBA uses manipulating of job orientation to obtain complete sub-mesh recognition ability. However, when job *J(p,q)* has

both *p* and *q* sizes equal (*p=q)* there is no need to change job orientation. Using stack as a storage for candidate blocks, algorithm returns first found base block as a result. In [17], three allocation algorithms are compared: SBA, ISBA and Frame Sliding Algorithm (FS). Moreover, Simulation results show that ISBA is more efficient in most cases in comparison with the other algorithms.

2.3 Task Migration

Task migration problem has also been widely studied in the literature [18-20]. An important issue in task migration is minimizing the collision between migration traffic and the normal traffic generated by the applications [21]. In this section, a few task migration strategies that are used in mesh multi-computers will be described. Methods like General Task Migration Scheme (G-TMS) and Near-Optimal Task Migration Scheme (NOTMS) try to minimize the traffic collision between the migration packets and the normal application packets and also among different migration packets in a wormhole switched multi-computer by sending the migration data in a multi-phase procedure [22]. In [23], a diagonal scheme is presented. The contribution of this algorithm is finding separate ways to move a task from the source sub-mesh to the destination sub-mesh on the basis of the X-Y routing. In [24], two strategies are introduced. They are Online Dynamic Compaction-Single Corner (ODC-SC) and Online Dynamic Compaction-Four Corner (ODC-FC). The ODC-SC tries to find the destination to move a sub-mesh in such a way that a larger free fragment of processors are obtained. Indeed, ODC-FC is more optimized version of ODC-SC that gives a larger region of adjacent free nodes by more selectively moving the tasks. Also, these methods prevent external fragmentation in the system. By this algorithm, there will be a larger contiguous area of free nodes after migration as compared to the previous schemes. Really, experiments show that this strategy is particularly useful in yielding better performance.

2.4 Simulation Output Interpretation

Following is three parameters used in our discussion below:

2.4.1 Mean Task Response Time (MTRT):

The response time is the time from the submission of request until the first real response produced for tasks [25, 26].

2.4.2 Mean Task Execution Time (MTET)

The execution time of a parallel task is the time from the allocation of the task's request until the moment the parallel task finishes execution [25].

2.4.3 Mean Task Waiting Time (MTWT)

The waiting time is the time interval between the instant when a task arrives and when it is allocated [27].

3. OVERVIEW OF THE PROPOSED APPROACH

Three steps are considered in the proposed method as follows:

3.1 Calculation of the Appropriate Size of Sub Mesh for Input Task

The following algorithm can be considered to calculate the appropriate size of sub-mesh in continuous allocation:

3.1.1 Decrease Loss by Minimum Diameter (MD)

The minimum diameter is considered in this method. For example if core count is equal to nine, the sub mesh has three rows and three columns by this method. The algorithm is shown in figure 2.

Based on number of cores needed for job

Make array of right products of core count as row and column

Find one element of array with min row and column

Return (row, column);

Figure 2: MD Algorithm

3.2 Proposed Task Allocation Method

3.2.1 Two Row Boundary (TRB) Allocation Algorithm

In this strategy, selection of end node is different. It is calculated base on the base node and task size (p×q). It is shown by equation (4).

$$endnode.x_i = basenode.x_i + p$$
$$endnode.y_i = basenode.y_i + q \quad (4)$$

In this method, if there is more than one base node, a sub-mesh will be selected that its base node has minimum distance from the top and down points of mesh as well as minimum free connectivity. Equation 4, 5 are used to calculate the base and end nodes' distance of a sub-mesh from boundaries. The mesh size is considered m×n.

$$b.x_i = basenode.x_i - 0$$
$$e.x_i = m - endnode.x_i$$
$$\min.d_i = \min(b.x_i, e.x_i)$$
$$m.d = \min(all\ \min.d_i) \quad (5)$$

In this way the allocation of free nodes are kept in the middle of the mesh which is shown in figure 3. Thus, the problem of external fragmentation can be minimized.

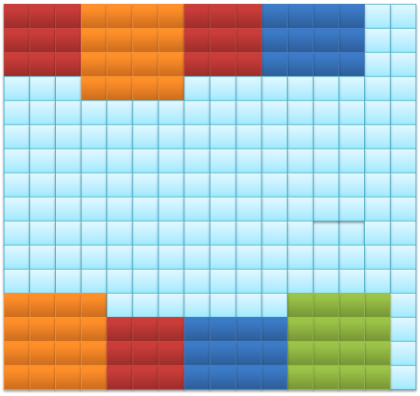

Figure 3: TRB allocation algorithm

The proposed TRB allocation algorithm is organized as follows:

TRB Allocation Algorithm

If *(number of free nodes less than needed nodes)* ***then***

Job must be waiting

else

Create CJ with respect to J (p,q) and J(q,p) based on busy list and create R_J *Create base node list based on (mesh –* $(C_J \cup R_J)$*)*

end if

If *(numbers of based node == 0)****then***

Job must be waiting// external fragmentation

else if *(numbers of based node == 1)****then***

Allocate job on available base node

else *///numbers of base node more than one*

Select base node with min diameter

Figure 4. Pseudo code of TRB allocation algorithm

3.3 Proposed Task Migration Method

In most previous studies, they have focused on reducing the effects of external fragmentation that are caused by the contiguous allocation strategies. In fact, external fragmentation occurs when the number of free nodes exceeds the number of nodes required for task, but no base node can be obtained for it**.** To solve this problem the migration algorithms have been proposed.

3.3.1 Limited Top-Down Compaction (LT-DC) Migration Algorithm

The proposed migration algorithm is derived from the ODC-FC method. For example, instead of a task to the top left corner or the top right corner of the mesh, only to be driven upward. Similarly, instead of a lower left corner or the lower right corner of the task, only to be driven down.

3.4 Task Mapping

After determining the mesh size for allocation, mapping algorithm and allocation algorithm can be run simultaneously. In this case, based on the output of the allocation function that is coordinates of the base node, and using the output of mapping function that is the coordinates of mapping nodes, the position of each nodes on the mesh can be achieved.

It is sufficient to sum the coordinate of each mapping function of output node with the coordinate of base node to gain real coordinate node of the mesh. After mapping task on the selected nodes of sub-mesh, the given task starts to run. The task will be put on a waiting list if the allocation function fails to allocate sub-mesh to input task, (low number of free nodes or external fragmentation problem). The output of the mapping algorithm is stored in the memory to use sub-mesh allocated to the given task. In this step, each mapping function can be used which in this paper random mapping function is used.

3.5 Simulation Results

For evaluation the proposed algorithm, we implement OM-simulator developed by C#, this simulator has three phase of on line mapping, allocation, migration (in non-preemptive allocation) and mapping. Simulator configuration is based on task parameter (task type, task size, task lifetime and task arrival time), network parameter (network size, communication rate), number of task and total time of the simulation.

The proposed algorithm has been compared with similar known algorithms. In the first phase, TRB allocation algorithm has been compared with the ISBA allocation algorithm. In the second phase, LT-DC migration algorithm has been compared with the ODC-FC migration algorithm. In addition, different random tasks of random size, time of arrival and the processing time is considered. Indeed, the proposed method with three different sub-mesh models has been implemented in the OM simulator. It should be noted that the same traffic applied to all simulation conditions. As can be seen in Figure 5, MD / TRB / ODC-FC (Dimensions of sub mesh / allocation / migration) method has the lowest average execution time.

Table 1: simulation configuration, OM simulator

Simulation Parameter	Value
NoC size	16×16
Communication rate	1 to 1000 bit/s
task type	Video & media
task size	Random between 9 to 32 core
task lifetime	Random between 100,000 and 1000,000
task arrival time	Random between 0 and 300,000
Total time of the simulation	20 million cycles
Number of tasks	Random between 50 and 200

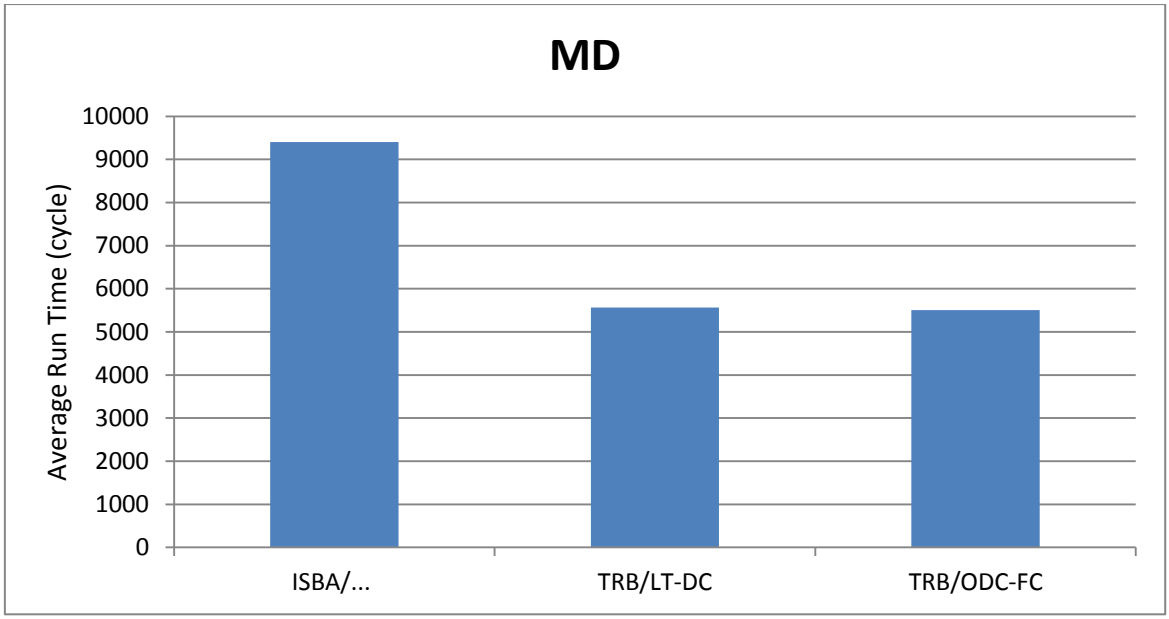

Figure 5: Average execution time for tasks considering different dimensions of sub mesh/allocation/migration schemes

Average task response time for traffic patterns that mentioned above is displayed in figure 6. As can be seen in this graph, MD / TRB / ODC-FC has the lowest average response time compared to other designs.

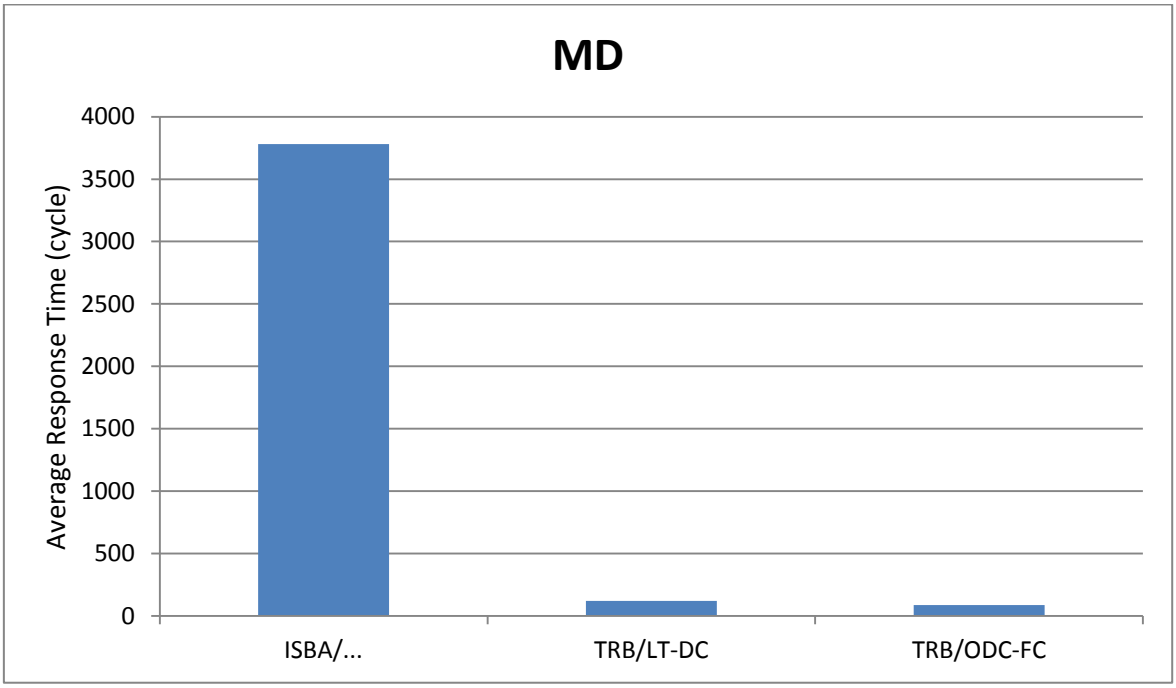

Figure 6: Average response time for tasks considering different dimensions of sub mesh/allocation/migration scheme

The average waiting time for all online mapping plans to input tasks on the mesh topology with network size of 16 × 16 is shown in figure 7. As can be seen the MD / TRB / ODC-FC has the least waiting time.

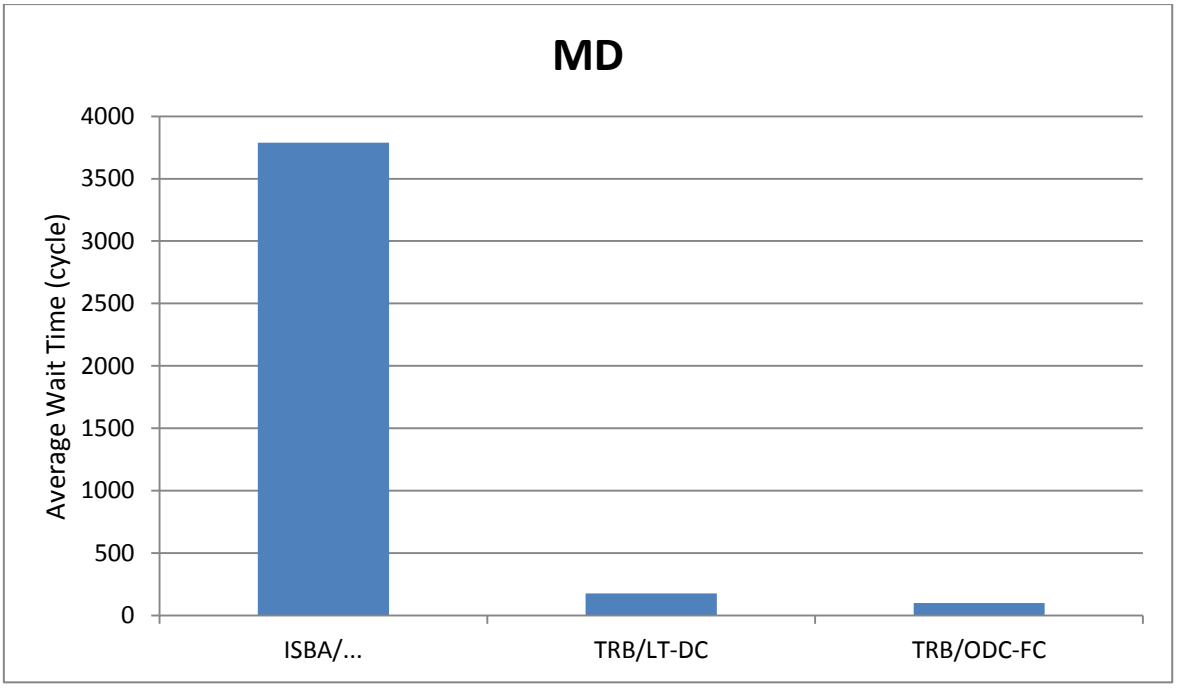

Figure 7: Average wait time for tasks considering different dimensions of sub mesh/allocation/migration scheme

4. CONCLUSION

Three concepts are considered in the proposed algorithm. They are the dimensions of sub mesh, allocation, and migration. Dimensions of sub mesh are considered by MD method. The proposed allocation mechanism is TRB strategy. And the proposed migration algorithm is LT-DC method. It should be noted that the proposed mechanism has been compared with similar known algorithms. They are ISBA allocation algorithm and ODC-FC migration method. Also, three parameters are considered. They are average execution time, average response time and average waiting time. Results of simulation show that MD / TRB / ODC-FC (Dimensions of sub mesh / allocation / migration) method with respect to these three parameters have better performance.

5. REFERENCES

[1] C. Celik, and C. F. Bazlamacci, "Effect of application mapping on network on chip performance", *in Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 465-472, 2012.

[2] P. K. Sahu, and S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design", *in Proceedings of the Journal of Systems Architecture*, Vol. 59, No. 1, pp. 60-76, 2013.

[3] Y. Zhu, "Efficient processor allocation strategies for mesh-connected parallel computers", *in Proceedings of the Journal of Parallel and Distributed Computing*, pp. 328-337, 1992.

[4] S. Bani-Mohammad, M. Ould-Khaoua, and I. Ababneh, "A new processor allocation strategy with a high degree of contiguity in mesh-connected multicomputers", Simulation Modelling Practice and Theory, pp. 465-480, 2007.

[5] V. Lo, K. Windisch, W. Liu, and B. Nitzberg, "Non-contiguous processor allocation algorithms for mesh-connected multicomputers", *in Proceedings of the IEEE Transactions on Parallel and Distributed Systems*, pp. 712-726, 1997.

[6] C. Peterson, J. Sutton, and P. Wiley, "iWARP: a 100-POS, LIW microprocessor for multicomputers", *in Proceedings of the IEEE Micro*, pp. 26-29, 1991.

[7] M. Levine, CRAY XT3 at the Pittsburgh Supercomputing Centre, *DEISA Symposium*, Bologna, 4-5 May 2006.

[8] W. Mao, J. Chen, and W. Watson, Efficient Subtorus Processor Allocation in a Multi-Dimensional Torus, *Proceedings of the 8th International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05)*, IEEE Computer society Press, pp. 53-60, 30 November - 3 December, 2005

[9] J. Ding, and L.N. Bhuyan, "An adaptive submesh allocation strategy for two dimensional mesh connected systems", *in Proceedings of the International Conference on Parallel Processing (ICPP)*, Vol. 2, pp. 193–200, 1993.

[10] K. Li, and K. Cheng, "A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system", *in Proceedings of the Journal of Parallel and Distributed Computing*, Vol. 12, No. 1, pp. 79–83, May 1991.

[11] P. Chuang, and N. Tzeng, "An efficient submesh allocation strategy for mesh computer systems", *in Proceedings of the 11th International Conference on Distributed Computing Systems*, pp. 256–263, 1991.

[12] J. Ding, and L.N. Bhuyan, "An adaptive submesh allocation strategy for two dimensional mesh connected systems", *in Proceedings of the International Conference on Parallel Processing (ICPP)*, Vol. 2, pp. 193–200, 1993.

[13] Y. Zhu, "Efficient processor allocation strategies for mesh-connected parallel computers", *in Proceedings of*

the Journal of Parallel and Distributed Computing, Vol. 16, No. 4, pp. 328–337, December 1992.

[14] B. S. Yoo, and C. R. Das, "a fast and efficient processor allocation scheme for mesh-connected multicomputers", *in Proceedings of the IEEE transactions on computers*, Vol. 51, No. 1, pp. 46-60, January, 2002.

[15] D.D. Sharma, and D.K. Pradhan, "A fast and efficient strategy for submesh allocation in mesh-connected parallel computers", *in Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing*, pp. 682–689, December 1993.

[16] Z.M. Al-Lami, "Communication Impact on Non-Contiguous Allocation Strategies for 2-D Mesh Multicomputer Systems", Master Thesis, Middle East University, Amman-Jordan, May 2011.

[17] G. Chmaj, D. Zydek, and L. Koszalka, Allocation Algorithms Problems in Mesh-Connected Systems, 2004.

[18] A. Kelly, and J. D. William, "Migration in single chip multiprocessors", *in Proceedings of the IEEE Computer Architecture Letters*, 1, 2002.

[19] M. Kandemir, and G. Chen, "Locality-aware process scheduling for embedded MPSoCs", *in Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, pp. 870–875, 2005.

[20] S. Bertozzi, A. Acquaviva, D. Bertozzi, and A. Poggiali, "Supporting task migration in multi-processor systems-on-chip: a feasibility study", *in Proceedings of the Design, Automation and Test in Europe (DATE)*, pp. 15–20, Vol. 1, March 2006.

[21] B. Goudarzi, and H. Sarbazi-Azad, "Task migration in mesh NoCs over virtual point to point connections", *in Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 463–469, 2011.

[22] N.C. Wang, and T.S. Chen, "Task migration in all-port wormhole-routed 2D mesh multicomputers", *in Proceedings of the Seventh International Symposium on Parallel Architectures, Algorithms, and Networks*, pp. 123–128, 2004.

[23] T. S. Chen, "Task migration in 2D wormhole-routed mesh multicomputers", *in Proceedings of the Journal of Information Processing Letters*, pp. 103–110, Vol. 73, No. 3-4, 2000.

[24] L. K. Goh, and B. Veeravalli, "Design and performance evaluation of combined first-fit task allocation and migration strategies in mesh multicomputer systems", *in Proceedings of the Journal of Parallel Computing*, pp. 508–520, Vol. 34, No. 9, September 2008.

[25] S. Bani-Ahmad, "On Improved Processor Allocation in 2D Mesh-based Multicomputers: Controlled Splitting of Parallel Requests", *in Proceedings of the 2011 International Conference on Communication*, Computing and Security (ICCCS'11), pp. 204-209, 2011.

[26] Z.M. Al-Lami, "Communication Impact on Non-Contiguous Allocation Strategies for 2-D Mesh Multicomputer Systems", Master Thesis, Middle East University, Amman-Jordan, May 2011.

[27] G. L. Kee, "Design and performance evaluation of migration-based submesh allocation strategies in mesh multicomputers", Master Thesis, National University of Singapore, 2005.