International Journal of Computer & Information Technologies (IJOCIT)
www.ijocit.ir & www.ijocit.org
ISSN = 2345-3877

Improving Multi Task Running Time in Two Column Boundary Allocation Method in Mesh-based Chip Multiprocessors Using Combined Migration Mechanisms

Akram Reza[1*], Mahnaz Rafie

Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University[1],
Department of Computer Engineering, Ramhormoz Branch, Islamic Azad University[2],
Tehran[1], *Ramhormoz*[2]
Iran[1,2]
A.reza@qodsiau.ac.ir[1], mahnaz.rafie@gmail.com[2]

Abstract:- In this paper, a calculation algorithm, a processor allocation mechanism and a migration method for NoC-based multiprocessors is presented. Calculation algorithm is used for calculating the appropriate size of sub-mesh for input task to increase continuity in multiprocessors. Processor allocation aims to allocate the processing nodes to different tasks of an input application at run time. Indeed, we employ the idea of using migration to minimize fragmentation of the tasks. 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 to quantify all our proposed schemes and compare them against standard methods. Thus, we make clear recommendations on the choice of the strategies.

Keywords: Fragmentation, Processor Allocation, Processor Migration, Two Column Boundary (TCB).

1. Introduction

Multicomputer parallel computer systems are cost-effective alternatives of the traditional supercomputers [1]. The interconnection of multi-computers come in different styles called topologies. The two-dimensional (2D) mesh-based topology is probably the most common topology because it is simple, regular and scalable. Several recent commercial and experimental parallel computers have been built based this architecture such as the IBM Blue Gene/L and the Intel Paragon [2-4].

Manuscript received 30 March 2015; revised 24 May 2015; accepted 1 July 2015

Processor allocation in 2D-Mesh multicomputer is a major issue as it significantly affects the performance of any parallel system [2]. It is concerned with the way for allocation sub-mesh to a job request. Indeed, processor allocation strategies are divided into two categories: contiguous and non-contiguous. In contiguous allocation, jobs are allocated distinct contiguous processor sub-meshes for the duration of their execution. Contiguous allocation suffers from processor fragmentation [4-8]. It should be noted that processor fragmentation can be classified into internal and external fragmentation. Internal fragmentation occurs when more processors are allocated to a job than it requires [1, 3, 5, 8-9]. When a job is assigned more processors than it requires, the extra allocated processors are not used for actual computation, instead they are wasted. External fragmentation occurs when a sufficient number of processors are available to satisfy a request, but they cannot be allocated because they are not contiguous [9]. A lot of research has been carried out to solve the problem of processor fragmentation. For example, non-contiguous allocation has been considered [3, 7, 9-10]. In this allocation strategy, a job can be executed on multiple disjoint sub-meshes rather than waiting until a single sub-mesh of the requested size and shape becomes available [9, 11]. Studies show that non-contiguous allocation of requests may solve the drawbacks of contiguous allocation and eliminate fragmentation. However, since communication between processors running the same job can be indirect due to non-contiguity [12], communication latency is usually high.

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. In fact, this paper is organized in four sections. The second section includes previous studies related to the processor allocation algorithms in mesh networks. 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. The third section includes implementation of proposed algorithm. The fourth section concludes of this paper and discusses about future work.

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 [13]. A node in column c and row r has the coordinate of <c, r> where $\mathbf{0} \leq c < \boldsymbol{w}$ and $\mathbf{0} \leq r < \boldsymbol{h}$. Node <i , *j*> that

is not in borderlines of mesh approximates and connects directly with other four nodes: <$i \pm 1$>, j and i, $j \pm 1$ 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 $0 \leq c \leq w$ and $0 \leq r \leq h$. 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 $\beta 1$ = <0,0,1,3> , $\beta 2$ = <0,4,2,5>, $\beta 3$=< 2,0,3,1> 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 (3,4) in figure (1), we have: $\vartheta_{\beta 1,T}$ = <0,0,1,3>، $\vartheta_{\beta 2,T}$ = <0,0,2,5>، $\vartheta_{\beta 3,T}$ = <0,0,3,1> ،C_{ST}= {<0,0,1,3> , <0,0,2,5> , <0,0,3,1>}

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, $w \times h$ 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} = <0,3,5,5> and δ_{TV} = <4,0,5,5> in figure (1).

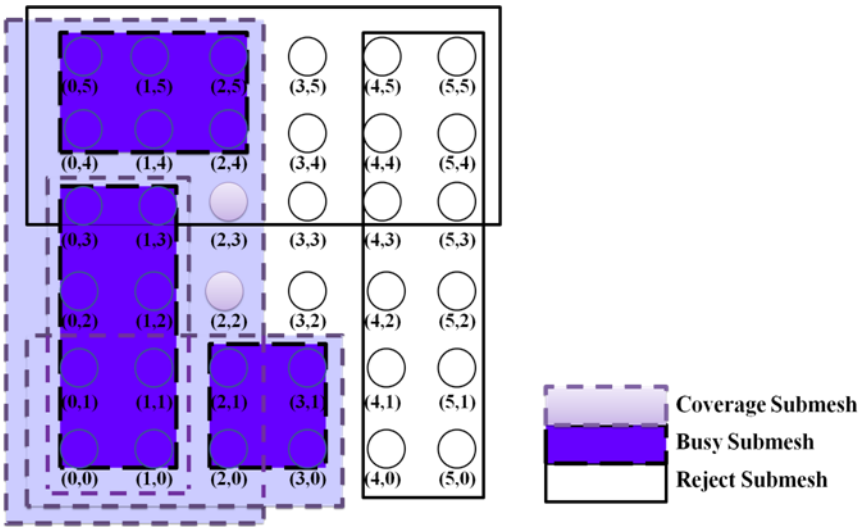

Figure 1: An Example of Allocation for T (3, 4)

2.2. Processor Allocation

Processor allocation in 2D-Mesh multicomputer is a major issue as it significantly affects the performance of any parallel system [1]. Also, contiguous allocation strategies attempt to locate a contiguous portion of the computing units for the execution of a parallel job. Indeed, most previous studies have been focused on reducing the negative effects of fragmentation of processors on the system efficiency due to the continuous allocation. Hereinafter, contiguous processor allocation schemes include a wide range of methods such as stack-based allocation [14, 15], adjacency allocation [15], adaptive scan allocation [13], and best/first fit allocation [16, 17]. In this article, improved stack based allocation algorithm was used to compare the speed of the proposed allocation method.

2.2.1. Improved Stack Based Allocation (ISBA)

Improvement of Stack Based Algorithm includes Rotation optimization and Task separation techniques. In fact, it 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 [18], 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 [19-21]. An important issue in task migration is minimizing the collision between migration traffic and the normal traffic generated by the applications [22]. In this section, a few task migration algorithms 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 [23]. In [24], a diagonal scheme is proposed. 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 [25], two strategies are presented. 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 method, 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. It should be noted that the ODC-FC scheme moves the tasks towards all four corners of the mesh so as to produce a larger contiguous space of free nodes in the centre of the mesh. This strategy is illustrated in figure 2.

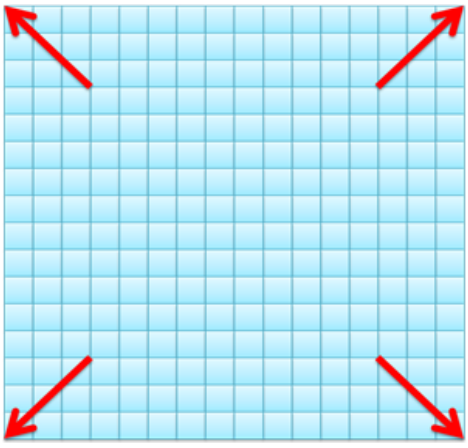

Figure 2: ODC-FC Migration Algorithm

2.4. Simulation Output Interpretation

There are three parameters used in our discussion. They are Mean Task Response Time (MTRT), Mean Task Execution Time (MTET), and Mean Task Waiting Time (MTWT). MTRT is the time from the submission of request until the first real response produced for tasks [17, 26]. MTET is the time from the allocation of the task's request until the moment the parallel task finishes execution [26]. MTWT 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 3.

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 3: MD Algorithm

3.2. Proposed Task Allocation Method

3.2.1. Two Column Boundary (TCB) 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 left and right 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 4. Thus, the problem of external fragmentation can be minimized.

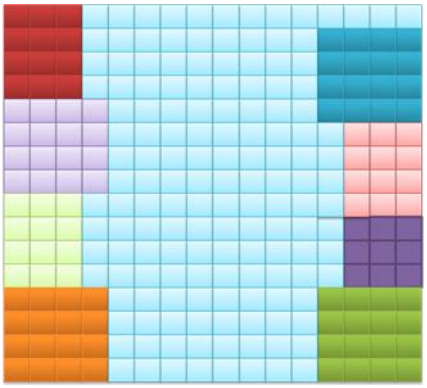

Figure 4: TCB Allocation Algorithm

The proposed TCB allocation algorithm is organized as figure 5.

TCB Allocation Algorithm

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

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

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

Allocate job on available base node

else

Select base node with min diameter and min F.C

end if

Figure 5: Pseudo Code of the TCB 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 [9]. To solve this problem the migration algorithms have been proposed.

3.3.1. Two Column Boundary (TCB) Migration Algorithm

The main goal of the migration algorithms is to determine the migration destination for sub-mesh. In this strategy, sub-meshes are displaced to the nearest boundary of left and right of the mesh. In this algorithm, at first, the distance from the nearest boundary of left and right of the mesh is calculated for all sub-meshes. Afterward, the minimum distance as the closest interval to the border for that sub-mesh is considered. Then, the sub mesh with minimum distance and non zero amount of free connections of the base or end node is selected as the final sub mesh for migration. Finally, the sub-mesh should be moved to the nearest border of the columns of the mesh so that the free nodes will be among the mesh which is shown in Figure 6.

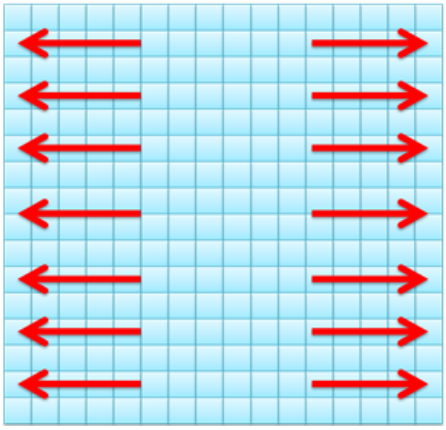

Figure 6: Two Column Boundary

3.3.2. Combinations of TCB and ODC-FC Migration Algorithms

This strategy has the benefits of the both TCB and ODC-FC migration algorithms. Really, in each iteration one of the algorithms is executed. For example if the number of migrations is equal to thirty, the TCB and ODC-FC algorithms will be run fifteen times.

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 online mapping. They are 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. Also, simulation configuration is shown in table 1. Indeed, the proposed algorithm has been compared with similar known methods. In the first phase, TCB strategy has been compared with the ISBA

allocation mechanism. In the second phase, TCB migration algorithm and combinations of the TCB and ODC-FC methods 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. It should be noted that the same traffic applied to all simulation conditions. As can be seen in Figure 7, MD / TCB / combinations of the TCB and ODC-FC (Dimensions of sub mesh / allocation / migration) method has relatively low average task 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

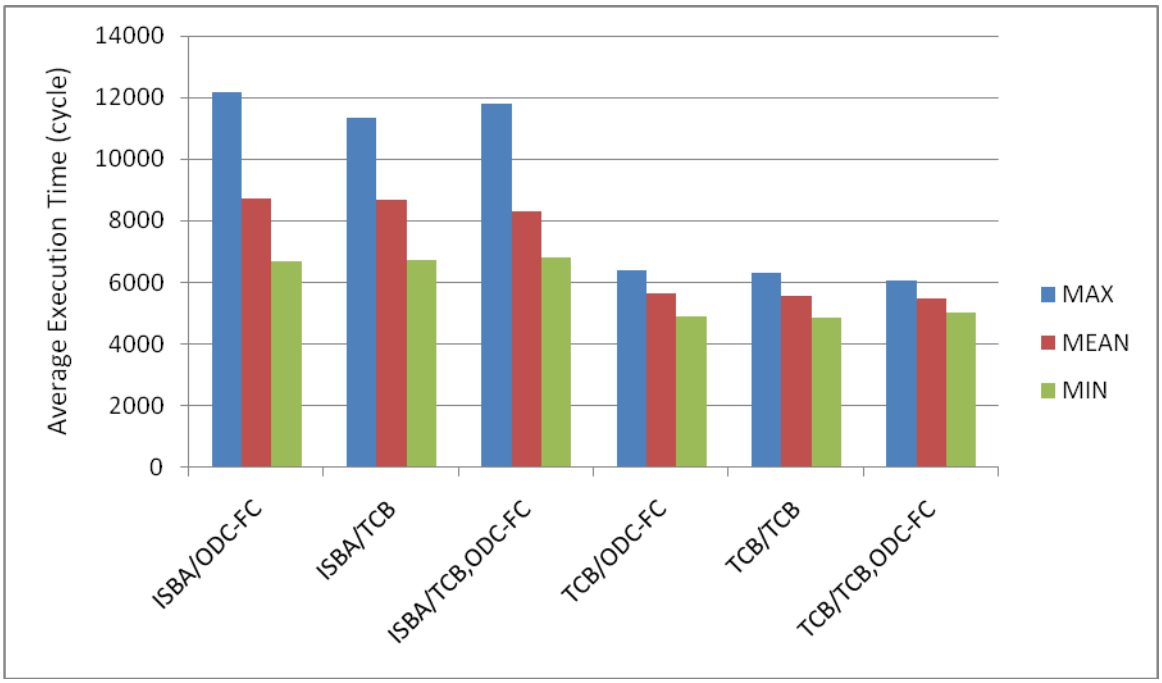

Figure 7: Average Execution Time for Tasks Considering Different Dimensions of Sub Mesh/Allocation/Migration Scheme

Average task response time for traffic patterns that mentioned above is displayed in figure 8. As can be seen in this graph, MD / TCB / combinations of the TCB and ODC-FC has the lowest average response time compared to other designs. Indeed, the minimum value of the average response time for TCB/complex algorithm is zero.

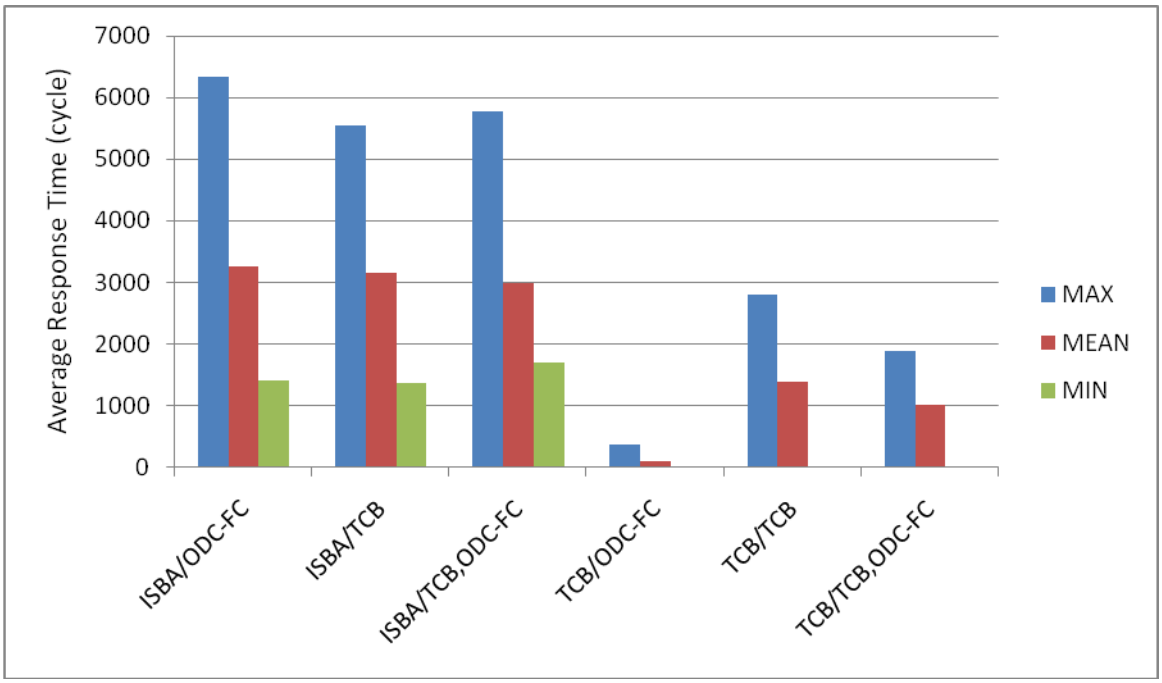

Figure 8: Average Response Time for Tasks Considering Different Dimensions of Sub Mesh/Allocation/Migration Scheme

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 9. As can be seen the MD / TCB / combinations of the TCB and ODC-FC has the least waiting time.

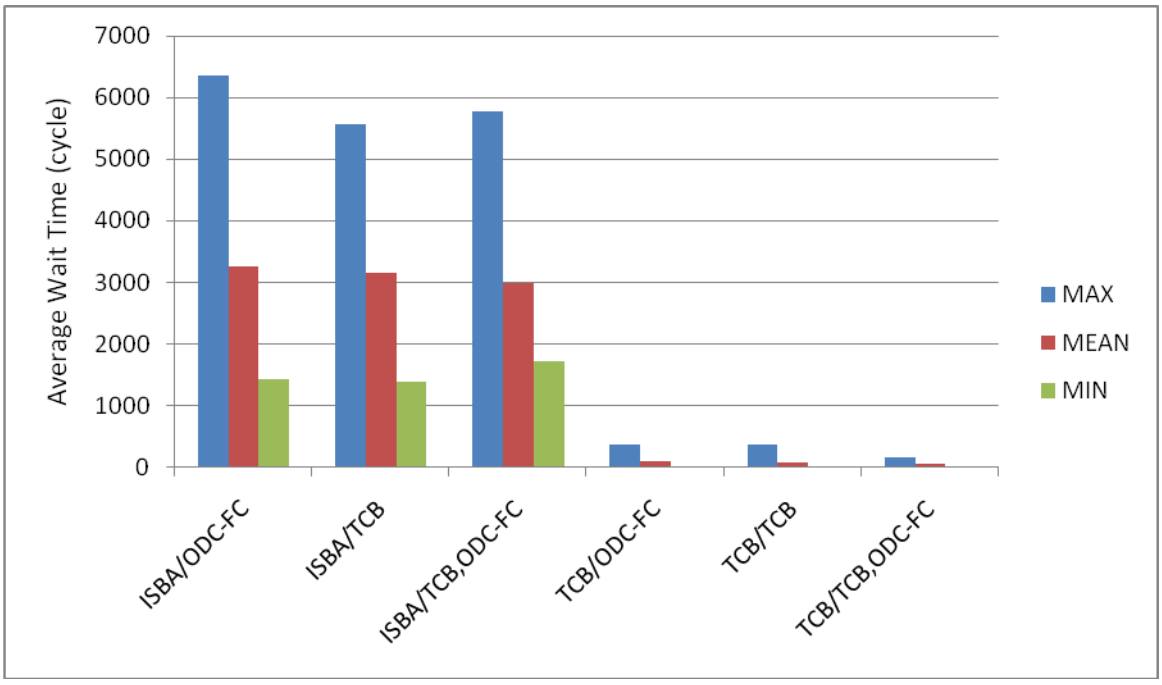

Figure 9: Average Wait Time for Tasks Considering Different Dimensions of Sub Mesh/Allocation/Migration Scheme

It should be noted that the minimum amount of the average wait time for TCB/ODC-FC, TCB/TCB and TCB/complex algorithms is zero which are not shown in figure 9.

4. Conclusion

The concepts of the dimensions of sub mesh, allocation, and migration are considered in the proposed algorithm. Dimensions of sub mesh are considered by MD methodology. The proposed allocation mechanism is TCB. And the proposed migration algorithm is combinations of the TCB and ODC-FC. 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 analyzed. They are average execution time, average response time and average waiting time. Results of simulation show that MD / TCB / combinations of the TCB and ODC-FC (Dimensions of sub mesh / allocation / migration) method with respect to these three parameters have better performance.

5. References

[1] C.-Y. Chang and P. Mohapatra, "Performance improvement of allocation schemes for mesh-connected computers", (1998), *In Proceedings of the Journal of Parallel and Distributed Computing*, Vol. 52, No. 1, pp. 40-68.

[2] I. Ababneh, "An efficient free-list submesh allocation scheme for two-dimensional mesh-connected multicomputers", (2006), *In Proceedings of the Journal of Systems and Software*, Vol. 79, No. 8, Elsevier Science Inc. , New York, NY, USA, August, pp. 1168-1179.

[3] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, "Non-contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers Based on Sub-meshes Available for Allocation", (2006), In Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06), Vol. 2, IEEE Computer Society Press, USA, pp. 41-48.

[4] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, "A Fast and Efficient Processor Allocation Strategy which Combines a Contiguous and Non-contiguous Processor Allocation Algorithms", (2007), Technical Report; TR-2007-229, DCS Technical Report Series, Department of Computing Science, University of Glasgow.

[5] K. Windisch, V. Lo, and B. Bose, "Contiguous and non-contiguous processor allocation algorithms for k-ary n-cubes", (1995), Technical Report, University of Oregon, Oregon, USA.

[6] ProcSimity V4.3 User's Manual, University of Oregon, (1997).

[7] S. Bani-Mohammad, Efficient Processor Allocation Strategies for Mesh-Connected Multicomputers, (2008), PhD Thesis, The Faculty of Information and Mathematical Sciences University of Glasgow, Glasgow, U.K.

[8] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Mackenzie, "Comparative evaluation of contiguous allocation strategies on 3D mesh multicomputers", (2009), *In*

Proceedings of the Journal of Systems and Software, Vol. 82, No. 2, pp. 307-318.

[9] K. Windisch, V. Lo, and B. Bose, "Non-contiguous processor allocation algorithms for mesh-connected multicomputers", (1997), *In Proceedings of the IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 7, pp. 712-726.

[10] D. Bunde, V. J. Leung and J. Mache, "Communication patterns and allocation strategies", (2004), Sandia Technical Report SAND2003-4522.

[11] V. Adve and M. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing", (1994), *In Proceedings of the IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 3, pp. 225-246.

[12] T. Srinivasan, J. Seshadri, A. Chandrasekhar, and J. Jonathan, "A Minimal Fragmentation Algorithm for Task Allocation in Mesh-Connected Multicomputers", (2004), *In Proceedings of the IEEE International Conference on Advances in Intelligent Systems – Theory and Applications – AISTA 2004 in conjunction with IEEE Computer Society*, ISBN 2-9599-7768-8, IEEE Press, Luxembourg, Western Europe.

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

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

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

[16] Y. Zhu, "Efficient processor allocation strategies for mesh-connected parallel computers", (1992), *in Proceedings of the Journal of Parallel and Distributed Computing*, Vol. 16, No. 4, pp. 328–337.

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

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

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

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

[21] S. Bertozzi, A. Acquaviva, D. Bertozzi, and A. Poggiali, "Supporting task migration in multi-processor systems-on-chip: a feasibility study", (2006), *in Proceedings of the Design,*

Automation and Test in Europe (DATE), Vol. 1, pp. 15–20.

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

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

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

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

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

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

Authors Profile

Akram Reza received her B.Sc. in Computer Engineering from Azad University of Ghazvin, Iran, MSc in Computer Architecture in Science and Research Branch of Islamic Azad University (SRBIAU), Tehran, Iran. She is currently a PhD candidate in Computer Architecture in SRBIAU. Her current research interests include network on chips, sensor networks, and computer architectures.

Mahnaz Rafie was born in Ahvaz, Iran. She received the B.Sc. degree in computer engineering from Allameh Mohaddese Noori Institute of Higher Education, Iran, in 2006, the M.Sc. degree in Computer Architecture from Azad University of Arak in 2011. She is currently a Ph.D. candidate in Computer Architecture at Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran. Since 2013, she has been with the Department of Computer Engineering, Islamic Azad University, Ramhormoz Branch. Indeed, she is a member of young researchers club since 2010 till now. Her current research interests include Network on Chips, Sensor Networks, Machine Learning and Computer Architectures.

