Efficientand Quick Algorithm for Processor Allocation in Mesh Multi-Computers Network

Rahmat Zolfaghari

***Abstract** -Until now, several continuous and discontinuous techniques have been given for processor allocation in mesh multi-computers networks. Continuous allocation methods always try to allocate a free continuous sub-mesh with the same requested dimensional structure to the parallel input job. For this reason, it produces the internal fragmentation in the processors network. Discontinuous allocation algorithms were produced with the aim of removing processors fragmentation. In discontinuous allocation algorithms, message interference between different jobs and struggle to get communication resources increases network communication overheads due to the increase in path length passed by the message. This communication overhead is highly dependent on to the manner of free sub-meshes allocation and the manner of recording by the algorithm. my paper, a discontinuous allocation algorithm called Quick Non-Contiguous Allocation (QNA) has been presented for a two-dimensional mesh network with C programming language. The efficiency of this algorithm and continuous and discontinuous allocation algorithms is determined and compared via simulator tool ProcSimity . Simulation results indicate improved performance parameters in the given algorithm.*

***Keywords:** multi-computers network, allocation processor, fragmentation, continuous and discontinuous algorithms*

I. INTRODUCTION

For optimal use of the computing power of a large multicomputer network, having a processor allocation algorithm and an efficient job schedule is very vital. Processor allocation is responsible for selecting a set of processors in order to run parallel work on them, while job schedule is responsible for determination of executing works. Job Schedule selects the next job for execution based on stated policy and then the processor allocation algorithm finds the free processors for the selected work. If input job cannot be executed upon the arrival due to lack of processor and or other jobs, it will be transferred to the waiting line. When some processors are allocated to a job, this job keeps the processors with itself until completion of work. After completion, job is gone out the system and the processors become free for other tasks. Most of the continuous and discontinuous allocation algorithms have been designed for two-dimensional mesh network. Mesh network has been the most favorite network among other networks for implementation of parallel computers with distributed memory due to simplicity, scalability, regularity and easy implementation and has been used in several machines such as: iWARP [9], IBMBlueGene / L [1,3] and DeltaTouchstone [6].

Manuscript published on 30 June 2013.
* Correspondence Author (s)
Rahmat Zolfaghari, Islamic Azad University, Hashtgerd Branch, Department of Computer Engineering, Tehran, Iran

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 job 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 job may be equal to the job execution time. Hence, development of strategies for minimizing search time (which is also called time allocation) 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 job. Allocation method (as shown in [10]) 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 [2, 7, 8, 13, and 14]. Discontinuous allocation is able to execute a job on several sub-meshes smaller than that the input job has requested and will not wait to release a continuous sub-mesh. Although a discontinuous allocation increases conflicts between messages in the system, it increases processors utilization in using the system processors and reduces the problem of fragmentation .Method of allocation operations has a direct impact on algorithm performance in discontinuous allocation algorithms. It should be noted that, processors fragmentation operation must be conducted in a way that the processors allocated to a job have necessary continuity because this continuity has a crucial role in decreasing communication overhead and maintains useful efficiency of system resources. For those discontinuous allocation algorithms presented for two-dimensional meshes, it should be mentioned that processor allocation operation is not conducted based on continuous free sub-meshes available in the network but it has been used predefined local models or mathematical that reduce the efficiency of these algorithms. A discontinuous allocation algorithm that is called quick non-contiguous allocation algorithm (QNA) has been proposed for a two-dimensional mesh network.

QNA algorithm combines the advantages of both continuous and discontinuous allocation methods. For example, the advantage of continuous allocation is to eliminate the communication overhead between processors assigned to a job that is also deeply considered in this algorithm. This algorithm has the capability of complete detection and reduction of allocation overhead. This quality is achieved by maintaining the maximum continuity between the processors assigned to a job. QNA algorithm is capable to be applied in both two- and three-dimensional mesh multi-computers networks.

In this paper, QNA algorithm performance has been compared using simulations with discontinuous allocation algorithms known as Paging (0) and MBS. These two algorithms have been selected because of the best performance among other algorithms [8]. QNA algorithm has been compared to FF continuous algorithm (that has been used in previous studies) in order to show superiority of discontinuous allocation to continuous allocation with respect to the problem of fragmentation in continuous allocation. At first, previous studies related to the processor allocation algorithms in mesh networks will be reviewed. In review of literature, studies conducted on improvement in efficiency of allocation algorithms will be investigated and the manner of these algorithms performances will be summarized. In part 3, QNA discontinuous allocation algorithm will be described and the manner of allocation this algorithm will be exemplified. In Section 4, QNA algorithm and implemented continuous and discontinuous allocation algorithms have been compared from the viewpoint of several important parameters in performance. And finally, results of the previous studies are discussed.

II. REVIEW OF LITERATURE

Definitions and methods of continuous and discontinuous allocation 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. A node in column and row b has the coordinate of $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$where $\mathbf{0} \leq \mathbf{a} < \boldsymbol{w}$ and$\mathbf{0} \leq \mathbf{b} < \boldsymbol{h}$. Node $\langle i, j\rangle$that is not in borderlines of mesh approximates and connects directly with other four nodes:$\langle i \pm 1, j\rangle$ and$\langle i, j \pm 1\rangle$ 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 (a, b) in the mesh M (w, h) is a subnet M (a, b) that $0 \leq a \leq w$ and $0 \leq b \leq h$. When a job requests a sub-mesh with dimensions$a \times b$, this job is expressed via T (a, b). Address for sub-mesh S is known by its end and base node that is a four-parameters variable as$\langle x, y, x', y'\rangle$ where,$< x, y >$ shows the lower left corner and $\langle x', y'\rangle$ shows the upper right corner of sub-mesh S. it is clear that$a = x' - x + 1$ and $b = y' - y + 1$ and base node of sub-mesh, is $\langle x, y\rangle$ and the sub-mesh area is the number of nodes inside it that is equal to a ×b.

*Definition 2-1-2:*Busy sub-mesh β is a sub-mesh that all its nodes are assigned to a job at that moment. A set of busy sub-meshes B is the set that set includes all the busy sub-meshes available in the network that is called busy list. For example, in figure (1), three busy sub-meshes exist in network M (6, 6); therefore, $B = \{\beta_1, \beta_2, \beta_3\}$ where$\beta_1 = \langle 0,0,1,2\rangle$, $\beta_3 = \langle 4,3,5,5\rangle$, $\beta_2 = \langle 2,0,3,1\rangle$are the members of this set.

*Definition2-1-3:***Co**verage sub-mesh for busy sub-meshβ 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 job T with respect to busy sub-mesh$\vartheta_{\beta,T}$. Coverage sub-mesh $\vartheta_{\beta,T}$ is equal to $\langle x_c, y_c, x', y'\rangle$for $\beta\langle x, y, x', y'\rangle$and the job $\beta$where, $y_c = \max(0, y-b+1)$ and $x_c = \max(0, x-a+1)$. A according to the input job T, coverage set C_T is a collection of coverage sub-meshes for the job T where, $C_T = \{\vartheta_{\beta,T} | \beta \in B\}$. For example, for the input job T (3, 2) in figure (1), we have: $\vartheta_{\beta 1,T} = \langle 0,0,1,2\rangle$ ، $\vartheta_{\beta 2,T} = \langle 0,0,3,1\rangle$ ،$\vartheta_{\beta 3,T} = \langle 2,2,5,5\rangle$،

$C_T = \{\langle 2,2,5,5\rangle, \langle 0,0,3,1\rangle, \langle 0,0,1,2\rangle\}$

Definition2-1-4:According to the input job T, rejectδ_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 job 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. $\delta_{TV} = \langle a', 0, w, h\rangle$ and $\delta_{TH} = \langle 0, b', w, h\rangle$ and$a' = w - a + 1$ and $b' = h - b + 1$ where, $w \times h$ is sub-mesh size. A set of reject sub-meshes Δ_T is calculated by adding$\delta_{TH}$and δ_{TV}. For example, $\delta_{TH} = \langle 0,5,5,5\rangle$and $\delta_{TV} = \langle 4,0,5,5\rangle$ in figure (1).

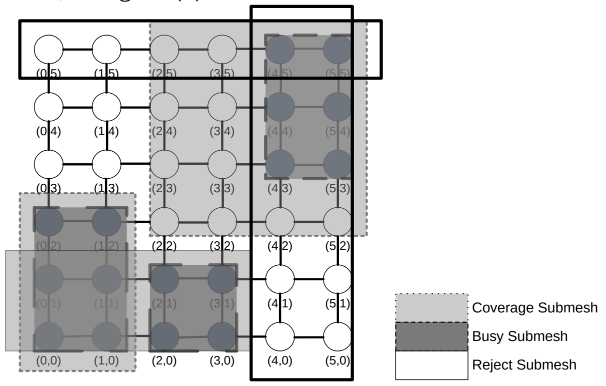

Figure 1 – An example of allocation for T (3, 2)

2-2 Continuous allocation methods

Continuous allocation has been proposed for mesh multi-computers networks. Most previous studies have been focused on reducing the negative effects of fragmentation of processors on the system efficiency due to the continuous allocation. Some known solutions will be described below.

First-Fit/ Best-Fit (FF / BF)

First-Fit/ Best-Fit algorithms [10] were proposed to improve the efficiency of the sliding frame. First-Fit algorithm is implementable on the sub-mesh with any size as sliding frame and can allocate a sub-mesh with the requested size correctly. This algorithm keeps bit map of the status of mesh free and allocated nodes in the array called busy array and according to the job given for allocation, look for busy array algorithm for creating an array called coverage array. Coverage array has been produced by scanning all busy arrays from left to right and top to bottom and returns the address of the first free node found in coverage array as a base node for allocation. Best-Fit method is similar to First-Fit but it returns a node as a job basis node where its sub-mesh has the most allocated neighbors. The simulation results show that First-Fit method is better than Best-Fit [10]. In First-Fit/ Best-Fit, whole mesh must be scanned to find the base node. Therefore, it is the time complexity of algorithm O (N) where N is the number of processors.

First-Fit method has not a complete diagnosis. Another drawback of this algorithm is high overhead due to the array manipulation that decreases the popularity of this algorithm, especially in the large meshes.

2.3Discontinuous allocation methods

With the developments in routing techniques such as wormhole switching, delayed communication had a fewer sensitivity to distances between nodes. These developments led to a more acceptable form of discontinuous allocation in networks with large diameters such as mesh. in a case of sufficient processors for allocation, discontinuous allocation does not seek for data execution and necessarily a continuous pattern. Some discontinuous allocation methods will be examined here .

Paging allocation

Paging allocation method [8] divides the whole mesh into pages that are sub-meshes with equal sizes and in length 2^{size_index} is larger than or is equal to zero. A page is regarded as an allocation unit. To determine the type of navigation, pages are identified by the same index. Page sizes are expressed bypaging(*size index*)**.** For example, Paging (2) means the pages that composed of sub-meshes with dimensions 4× 4. If a job asks for a sub-mesh with dimensions$a \times b$, the number of required pages is calculated by the formula$\left\lceil {(a \times b)}/{Psize} \right\rceil$ where, Psize is the page size. This algorithm maintains free pages in a list and in a case of request; it allocates from this list and returns it to the list when releasing the page. If in a case of *size index=0***,** there is no fragmentation but there is fragmentation with increase in ***size index***, time complexity of algorithm is $O(a \times b)$.

Multiple Buddy Systems (MBS)

Binary shape of this algorithm is a developed form of [8]. This method divides mesh network into a square and non-overlapped sub-meshes with dimensions of 2 spuare. If a job asks for a processor P, this request is converted to the request in base 4. In this way, $P = d_k \times \left(2^k \times 2^k\right) + \cdots + d_0 \times (2^0 \times 2^0)$ so that$d_0 \ldots d_k \in \{0,1,2,3\}$. Algorithm tries to allocate $d_i \times (2^i \times 2^i)$ according to the available resources. If some blocks do not exist, the algorithm breaks repeatedly the larger blocks and converts them to four smaller partners in order to achieve its intended size.

Four-partners blocks will be$(2^j \times 2^j)$ and four blocks will be$(2^{j-1} \times 2^{j-1})$. In a case of sufficient processors, algorithm is always successful because the smallest part that can be allocated is block1×1 . Consequently, there will be no fragmentation. Time complexity of this algorithm is O (N) where N is the number of processors in system.

III. QUICK NON-CONTINUOUS ALLOCATIONS

Method of quick algorithm

Suppose that the input job T (3, 2) has been given to the system and we can do the allocation by use of QNA algorithm. It is clear form figure (1) that busy sub-meshes include$\boldsymbol{\beta_2} = \langle \mathbf{2,0,3,1} \rangle$, $\boldsymbol{\beta_1} = \langle \mathbf{0,0,1,2} \rangle$and$\boldsymbol{\beta_3} = \langle \mathbf{4,3,5,5} \rangle$ that were allocated in mesh network M (6, 6).

According to the busy sub-meshes and the input job T, the coverage sub-meshes will be C_T= $\{\langle 2,2,5,5\rangle, \langle 0,0,3,1\rangle, \langle 0,0,1,2\rangle\}$ and the reject horizontal and vertical areas are

$\boldsymbol{\delta_{TH}} = \langle \mathbf{0,5,5,5} \rangle$، $\boldsymbol{\delta_{TV}} = \langle \mathbf{4,0,5,5} \rangle$

The main idea for QNA algorithm is to collect information from available rows in sub-mesh network via the coverage sub-meshes made of the busy sub-meshes. From this information, we can determine in the shortest time whether there is a node in a row for allocation to the input job T as the base node. This information is merely obtained by the comparison of the coverage sub-meshes and the rows and minimizes the comparisons in search spaces and finally allocation time and waiting time a great degree.

For algorithm performance, it is necessary to introduce a one-dimensional array called *last-covered*, which keeps the very right node covered in each (*x-coordinate*) row in the mesh network. In this article, a set of connected nodes in a row of mesh net is called a piece that begins from the very left node in the row (It is usually zero in definitions). If all the nodes in a piece belong to one of the coverage sub-meshesC_T, then that piece is called “coverage piece”. In array j of array *last covered[j]* where,$1 \le j \le b' - 1$, it keeps x-coordinate of the last node of coverage piece in the row j. At the beginning, algorithm calculates reject sub-meshes Δ_T after determination of the coverage sub-meshes according to the dimensions of the input job and eliminates it from whole search domain. Then, we arrange the coverage sub-meshes according to their coordinate *Xc* of base node parameters in an ascending form and then calculate the values of arrays *last-covered* by the *last-covered* function. If there is no coverage piece in the *j* row, the value of *last-covered[j]* will be zero. For example, the values of *last-covered[j]* for ∫*=0,1,2,3,4* will be *(3,3,5,0,0)* respectively.

Procedure Submesh Allocation

```
{
Step 1.flag←false. /* flag representing the orientation */
Step 2.Job_Size= $a \times b$
Step 3. Decide the orientation of T as follows, and determine the reject set.
if (flag = false)
        thenT ←T(w, h), a' ←a-w + 1, b'←b-h + 1
elseT←T(h, w), a'←a-h + 2, b←b-w + 1
Step 4. Based on current B and T, determine $\boldsymbol{\vartheta_{\beta,T}}$ and
Last_covered[j] $(1 \le j \le b' - 1)$← 0
      For each $\boldsymbol{\beta}\langle \mathbf{x,y,x',y'} \rangle$, determine
        $\boldsymbol{\vartheta_{\beta,T}}\langle \mathbf{x_c,y_c,x',y'} \rangle$
   Arrange $\boldsymbol{\vartheta_{\beta,T}}$ s in the increasing order of $\mathbf{x_c}$
   For each $\boldsymbol{\vartheta_{\beta,T}}$ (starting from one whose $\mathbf{x_c}$ is
smallest)
        If ( $\mathbf{y_c}$<b')
          For each row j $(y_c \le j \le \min(y', b' - 1))$
   If$(x_c \le last\_covered[j] + 1 \le x')$then
last_covered[j]← x'
Step 5.
j←1
while (j <b' AND last_covered[j] + 1 ≥a') /* no freesubmesh is found in the j th row */
        j←j+ 1
```

```
if (j = b') /* no free submesh found in that orientation */
        if (flag = false)
        then  flag←true and go back to Step 3
elsei← (last_covered[j] + 1) and go to Step 6. /*a
freesubmesh is found */
Step 6.
        If (flag = false)
thenS ←<i, j, i + w-1, j + h-1>
        elseS ←<i, j, i + h-1, j + w-1>
        Allocate S to T and add StoB.
        Return success
```

For example, in figure (1) $\langle 0,0,1,2\rangle \beta_2 = \langle 2,0,3,1\rangle$, $\beta_1 =$ and $\beta_3 = \langle 4,3,5,5\rangle$ and the input job T=(3 , 2) and b'-1=4, then the *last-covered* is calculated as follows:*last_covered[j] (j=0,1, 2, 3,4)= 0*

By using three coverage sub-meshes $\boldsymbol{\beta_3}$,$\boldsymbol{\beta_2}, \boldsymbol{\beta_1}$we get three sub-meshes $\boldsymbol{\vartheta_{\beta 1,T}} = \langle \mathbf{0,0,1,2}\rangle$,$\boldsymbol{\vartheta_{\beta 2,T}} = \langle \mathbf{0,0,3,1}\rangle$and$\boldsymbol{\vartheta_{\beta 3,T}} = \langle \mathbf{2,2,5,5}\rangle$ then we arrange them as, $\boldsymbol{\vartheta_{\beta 1,T}}$,$\boldsymbol{\vartheta_{\beta 2,T}}and\boldsymbol{\vartheta_{\beta 3,T}}$.

According to the$\boldsymbol{\vartheta_{\beta 1,T}}$, the values *last-covered[j]* for *j=0, 1, 2* equals 2. For j=0,1 the values of *last-covered[j]* for $\boldsymbol{\vartheta_{\beta 3,T}}$ equals to 3 and with the quantity of *last-covered [2]* is changed and equals to 5.

Final value for last_covered for 5 element left to right is in order 0*,0,5,3,3* .

As it can be seen, if a node belong to sub-meshβ, it belongs certainly to sub-meshC_T. Therefore, for determining the dependency of a node, we need to examine coverage sub-meshes. And *last-covered* has the necessary information in this regard. By examining the values of this array, we can determine whether a node exists to allocation to a job. Now, we have b=5 and a=4 for allocating a node to the job according to figure (1) and because$last_covered[j](1 \le j \le 3) + 1 \ge a'$, the result of value j is equal to 4. Then, because$last_covered[4] + 1 < a'$, node$\langle 1,4\rangle$ can be allocated to the job T as a base node. Note that QNA algorithm is more time-saving in compared to other methods.

```
ProcedureQNA_Allocate (a,b):
{
Total_Allocated=0
Job_Size= $a \times b$
 Step1. if (number of free processors<Job_Size)
                Return failure.
 Step2. if (there is a free S(x,y) suitable for S(a,b))
        {
Allocateit using Submesh Allocation contiguous
allocation algorithm.
        return success.
        }
Step3. $\alpha = a and \beta = b$
Step4. Subtract 1 from max $(\alpha, \beta)$ if max >1
Step5. if(Total_allocated + $(a \times b)$ >Job_Size go to step4
Step6. if there is a free S (x,y) suitable for $S(a \times b)$
        {
        allocate it using Submesh Allocation.
        Total_allocated = Total_allocated+ $(a \times b)$.
        }
Step7. if (Total_allocated = Job_Size)
                return success.
        else
                go to Step4.
}end procedure
```

In QNA algorithm, when a parallel job is chosen for the processor allocation, the algorithm begins to search for a mesh in order to find a suitable sub-mesh for the input job. If the requested sub-mesh is found, it will be allocated to the job and the allocation process will be ended. Otherwise, the largest free sub-mesh which can be placed in S (a, b) will be allocated to it. Then the algorithm will search for the largest sub-mesh whose dimensions do not exceed the previous allocated sub-mesh provided that the number of the allocated processors does not exceed the quantity $\boldsymbol{a} \times \boldsymbol{b}$ The last phase is repeated until $\boldsymbol{a} \times \boldsymbol{b}$processors are allocated. For example, take into account the mesh situation M (6, 6) which is shown in figure (1) and then suppose that the input job has asked for a sub-mesh with the dimensions 62. As we see in the figure, there are no free $\mathbf{6 \times 2}$ sub-meshes. Therefore, QNA algorithm of the free$\langle \mathbf{0,3,3,4}\rangle$ and$\langle \mathbf{4,0,5,1}\rangle$sub-meshes are allocated to it as we will explain. First, the algorithm subtracts one unit from the largest angle of the requested sub-mesh; and the result will be sub-mesh $\mathbf{5 \times 2}$ which does not exist again. The process of subtraction goes on until the sub-mesh $\mathbf{4 \times 2}$ is obtained which does exist. Then, the algorithm while expressing that the quantity of the processors should not exceed$\mathbf{6 \times 2}$, will try to choose the sub-mesh whose dimensions does not exceed the previous allocated sub-mesh $(\mathbf{4 \times 2})$. In this example, $[(\mathbf{4 \times 2}) + (\mathbf{4 \times 2})] > (\mathbf{6 \times 2})$ consequently, the algorithm subtracts one unit from the largest angle of the sub-mesh $(\mathbf{4 \times 2})$and the result of the sub-mesh will be$(\mathbf{3 \times 2})$. But again,$[(\mathbf{4 \times 2}) + (\mathbf{3 \times 2})] > (\mathbf{6 \times 2})$, the subtraction goes on until the summation of the angles of the sub-mesh is less than the dimensions of the allocated submesh or equals the intended processors $(\mathbf{6 \times 2})$. In this example, $(\mathbf{2 \times 2})$sub-mesh is obtained which is available in the system. Then, the sub-mesh$\langle 4,0,5,1\rangle$is allocated to the job and the process is finished.

IV. THE RESULTS OF SIMULATION

Here, we represent the results of the assimilation of some contiguous and non-contiguous allocation methods such as Paging (0)**,** MBSand First-Fit (FF). We perform the algorithm of the allocation and release of these methods with the C language, and assimilate it by the assimilation software ProcSimity which is a tool for assimilating processor allocation and priority given to the job in multi-computers systems [9]. The mesh model which is used in assimilation is a square mesh with the length of L. The way of producing and entering of jobs are supposed to be of powered distribution and are serviced in the form of FCFS. The time of doing is supposed as the form of powered distribution with the average amount of a time unit. Two kinds of distributions are used for the way of producing the length and the width of the job. The first one is the monotonous distribution on [1, L] in which the length and width of the job are produced separately.

The second one is the powered distribution in which the length and the width of the job are produced in the powered form and with the average of half of the entire mesh. These distributions are the ones which are used in most assimilations [2, 9, 10]. Each assimilator is based on a perfect implementation of 1000 jobs. The results of assimilation on a sufficient number of implementation are averaged. Thus, their reliability is %90 and the error is less than %5. The inter-communication network uses a crawling procedure and an XY routing. Sending fleet data between two adjacent nodes takes a time unit and ***t1*** time unit is spent for finding the route of the fleet between two nodes. The message length is shown as$[\mathbf{1},\mathbf{L}]$. The allocated processors use one of the current communication models. The first model is the *all to one* model. In this model, a processor which is randomly chosen from a job sends the data packs to all of the processors of that job. As it has been said in [8, 11, 12], the number of the messages produced by a given job has a powered distribution of an average quantity of *num-mes*. The second communication model is called the *all to all* model in which each allocated processor to a job sends the data packs to all the processors of that job. This communication model creates much message communication involvement in the network and this is the weakness of non-contiguous allocation algorithms. In both models, the processors allocated to a job in a linear array are recorded and are numbered by a network row scanning in the array. The processor assimilator chooses the starting point and the destination from this array and then determines the starting point and the destination coordinates by a record. The system on which the assimilation is done is a (16×16)mesh in which the$\boldsymbol{t_s} = \mathbf{3}$time unit and the fleet is $\boldsymbol{P_{len}} = \mathbf{8}$and the *num-mes=5*

The parameters chosen for comparison are: the *average turnaround time of jobs, mean waiting time and mean system utilization*. The average turnaround time of job is the time which a job spends from the entering to the exit time. The average finishing time of all jobs is the time which is spent for doing all the entering jobs. The average optimum use of the system is the percentage of using system processors during the implementation; and it is estimated as follows:

$$\mathrm{SystemUtilization} = \sum_{i=1}^{t} \frac{w \times h - n_i}{(w \times h) \times t} \qquad (1)$$

In this formula $\boldsymbol{n_i}$is the number of free processors of the system in time $\boldsymbol{i}$ and$\boldsymbol{t}$ is the total spent time, and $\boldsymbol{w} \times \boldsymbol{h}$is the number of the system processors. System loading is an independent parameter in the system which has an invert relation with the *mean inter-arrival of jobs* and is estimated as follows:

$$\lambda = \frac{N \times T_e}{\mathrm{SystemLoad} \times P} \qquad (2)$$

In this formula $\boldsymbol{F}$ is the total number of the processors and the jobs are entered into the system by the potation distribution and the rate of the $\boldsymbol{\lambda}$ in the time unit. $\boldsymbol{N}$ is the average number of the wanted processors by each job, and $\boldsymbol{T_e}$ is the average powered distribution of the implementation time.

The Completion Time of Each Job

In figures (2) and (3) the average completion time of each job in relation to the system's load for the communicating model "one to all" is represented. The results have been shown that QNA has a better performance than the other contiguous and non-contiguous allocation algorithms with both distribution models of job size (considered in this article).We should notice that QNA has a better performance than FF contiguous allocation for both models of job size distribution. For example, in figure (2) the algorithm QNA in mean inter-arrival time of jobs 0,0205 jobs/Time unit has been shown %65 more efficient in comparison to the FF , and %36 more efficient in comparison to Paging (U) and %30 more efficient than the MBS. Using messages longer than (16, 32 and 64 fleets) shows the same results from the efficiency aspect. The results also have been shown that by extending the length of packs, discrepancy of the parameters of QNA efficiency in comparison to other contiguous and non-contiguous algorithms has been shown more improvment.

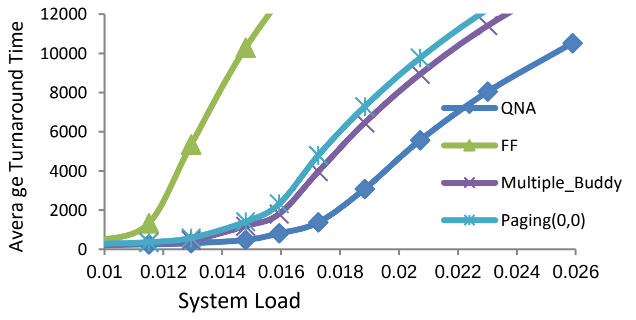

Figure (2) – The average completion time of a job according to the system of loading in the *one to all* communicating model with a monotonous distribution of jobdimensions

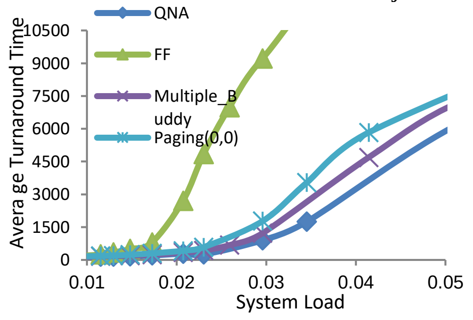

Figure (3)- The average completion time of a job according to the system of loading in the *one to all* communicating model with the powered distribution of job dimensions

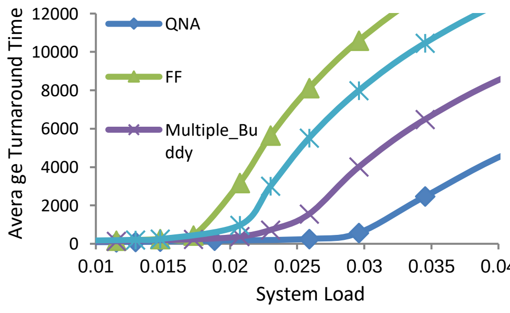

Figure (4)-The average completion time of a job according to the system of loading in the *all to all* communicating model with the monotonous distribution of job dimensions

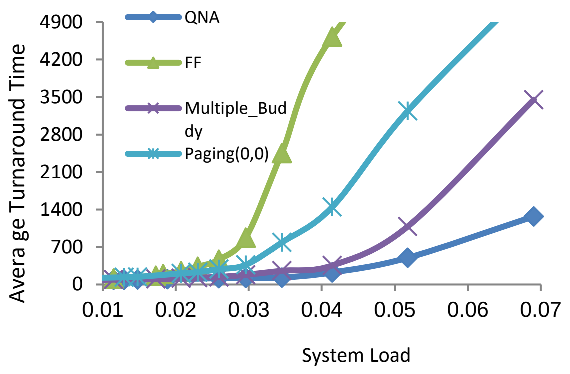

Figure (5)- The average completion time of a job according to the system of loading in the *all to all* communicating model with the powered distribution of job dimensions

In figures (4) and (5) the average completion of each job is measured in relation to the system load for the "all to all" communicating model. Again, QNA has been shown a better performance than the other allocation algorithms for both models of job size distribution. For example, in figure (4) when the mean inter-arrival time of jobs is 0, 0305 jobs/unit, the average completion time of the algorithm QNA is %20, %24 and %38 of the average completion time of FF, Paging (U) and MBS methods respectively.

The assimilation also has been shown shows that using messages longer than 16, 32 and 64 fleets have also the same results.

Utilization

Figures (6) and (7) have shown the average productivity of system resources in the allocation algorithms QNA, MBS, Paging (U) and FF for both communicating models and job size distribution. The assimilation results in these figures are obtained in the system's heavy load. The heavy load, i.e. the waiting line of the system is rapidly filled and causes the allocation algorithm to reach the highest level of using the system's resources. For both job size distributions of non-contiguous allocation algorithms they found an average productivity quantity of %71 to %76, but the contiguous method FF could not go beyond %50, and this was because doing QNA operation by other allocation algorithms for both of job size distributions showed a better performance. For example, in figure (4) when the mean inter-arrival time of jobs is 0,0305 jobs/unit, the average time of algorithm completions are %20, %24 and %38 of the average completion time of FF, Paging (U) and MBS methods respectively.

Also, it has been shown that the allocation is contiguously done and after that fragmentation occurred that prevents a good allocation. The average productivity of system resources for non-contiguous algorithms for both job size distributions is almost equal and this is because both of these algorithms have the same power in reducing the fragmentation. When the numbers of free processors of the system were equal or more than to the requested processors, these algorithms always do the job allocation successfully.

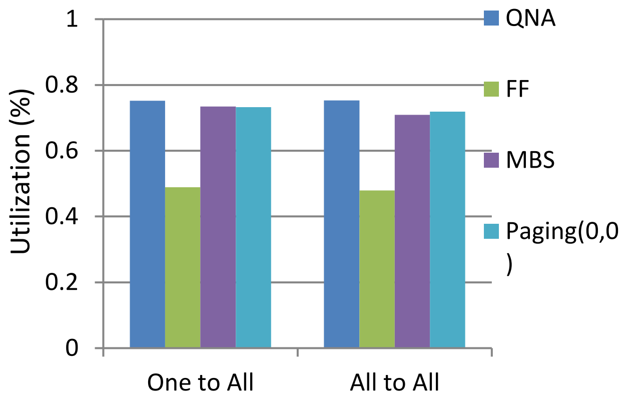

Figure (6)- The optimum use of system resources in contiguous and non-contiguous methods for both communicating models with monotonous job dimensions distribution in 16 * 16 sub mesh

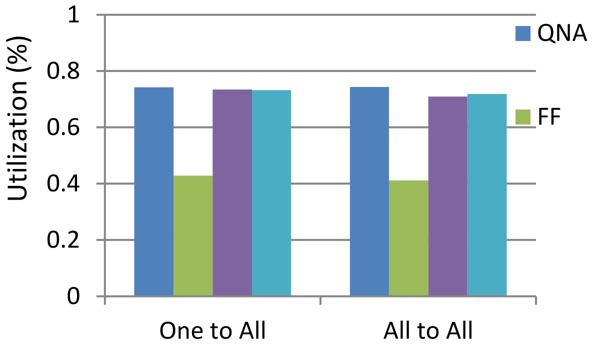

Figure (7)- The optimum use of system resources in contiguous and non-contiguous methods for both communicating models with powered distribution of job dimensions in 16 * 16 sub mesh

Waiting Time

In figures (8) and (9) the mean waiting time for each job in regard of system's load is shown for the *all to one* communicating model. The results show that QNA has a better performance than the other contiguous and non-contiguous allocation algorithms with both job size distribution models (which are considered in this article) and the reason for this is that allocation processors in QNA are more contiguous than the previous non-contiguous allocation methods which decreases the passed distance by the related messages to a job. After the decreased distance passed by a message, we will see a decreasing in the overload of allocation; and this shows that the processor allocation in QNA is performed better than the other methods and the logical result is that the waiting time decreases for a job. QNA performance in comparison to the FF contiguous allocation has a considerable improvement for both job size distribution models as well. For example, figure (8) represents that the mean waiting time for QNA in the mean inter-arrival time of jobs with 0,0205 jobs/unit are respectively is equal to %35, %64 and %70 of the mean waiting time in FF, Paging(U) and MBS methods.

In figures (10) and (11) according to the system's load for the *all to all* communicating model, mean waiting time for each job is given. Again QNA has a better performance than the other allocation algorithms (for both job size distribution models). For example, in figure (11) when mean inter-arrival time of jobs is 0,05 jobs/unit, the mean waiting time of QNA algorithm will be %19, %27 and %50 in FF, Paging(U) and MBS methods respectively.

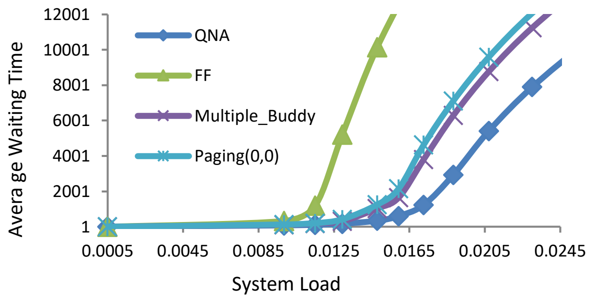

Figure (8)- The mean waiting time according to the system's loading in *all to one* communicating model with monotonous job dimensions distribution

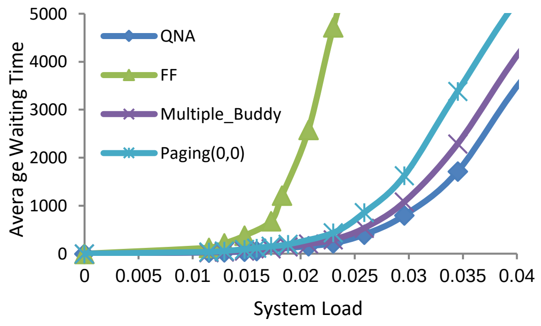

Figure (9) The mean waiting time according to the system's loading in *all to one* communicating model with powered job dimensions distribution

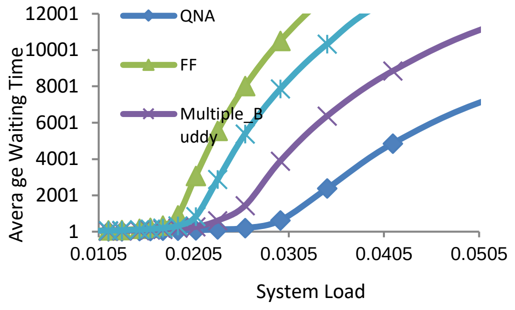

Figure (10)- The mean waiting time according to the system's loading in *all to all* communicating model with monotonous job dimensions distribution

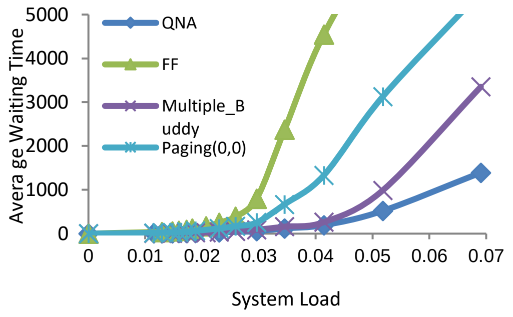

Figure (10)- The mean waiting time according to the system's loading in *all to all* communicating model with powered job dimensions distribution

CONCLUSION

The efficiency of QNA was compared with the efficiency of contiguous and non contiguous algorithms. The results of assimilations shown that QNA in spite of the available communicating in the net, it has been resulted of interference of different jobs messages with each other; it increasing the efficiency to a great extent. QNA also efficiently takes advantage of the system's resources while keeping maximum consistency and preventing internal and external fragmentation.
Also, the results considerably shown that QNA with respect to job completion time which is an important parameter of efficiency has superior to known allocation methods such as MBS and Paging (U). Furthermore; the experiences prove that QNA also has a better performance in comparison to the previous contiguous and non-contiguous allocation techniques when the packs are longer and the sub meshes systems have larger dimensions. It is expected that this procedure practically keeps its efficiency because when the sub mesh dimension get larger, it increasing the needs of the programs such as the number of the required processors as well.

REFERENCES

[1] Aridor Y., Domany T., Goldshmidt O., Kliteynik Y., Moreira J., and Shmueli E.; "Open Job Management Architecture for the Blue gene/L Supercomputer," Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing, pp. 91-107, 2005.
[2] Bani-Mohammad S., Ould-Khaoua M., and Ababneh I., “A new processor allocation strategy with a high degree of contiguity in mesh-connected multicomputers,” Simulation Modelling Practice and Theory, pp. 465-480, 2007.
[3] Blumrich M., Chen D., Coteus P., Gara A., Giampapa M., Heidelberger P., Singh S., Steinmacher-Burow B., Takken T., Vranas P.; "Design and Analysis of the BlueGene/L Torus Interconnection Network," IBM Research Report RC23025, pp. 231-235, 2003.
[4] Chang C. Y., Mohapatra P.; "Performance improvement of allocation schemes for mesh-connected Computers," Journal of Parallel and Distributed Computing, pp. 40-68, 1998.
[5] Chuang P. J., Tzeng N. F.; "Allocating precise submeshes in mesh connected systems," IEEE Transactions on Parallel and Distributed Systems, pp. 211-217, 1994.
[6] Kim G., Yoon H.; "On submesh allocation for mesh-connected multicomputers: a best-fit allocation and a virtual submesh allocation for faulty meshes," IEEE Transactions on Parallel and Distributed Systems, pp. 175-185, 1998.
[7] Lo V., Windisch K., Liu W., and Nitzberg B.; "Non-contiguous processor allocation algorithms for mesh-connected multicomputers," IEEE Transactions on Parallel and Distributed Systems, pp. 712-726, 1997.
[8] Peterson C., Sutton J., and Wiley P.; "iWARP: a 100-POS, LIW microprocessor for multicomputers," IEEE Micro, pp. 26-29, 1991.
[9] Windisch K., Miller J.V, Lo V.; "ProcSimity: an experimental tool for processor allocation and scheduling in highly parallel systems," in: Proc. 5th Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, pp. 414-421, 1995.

[10] Zhu Y.; "Efficient processor allocation strategies for mesh-connected parallel computers," Journal of Parallel and Distributed Computing, pp. 328-337, 1992.
[11] D. G. Feitelson, Workload Modeling for Computer SystemsPerformanceEvaluations, 2007*http://www.cs.huji.ac.il/~feit/wlmod/wlmod.pdf*
[12] L. He, S. Jarvis, D. Spooner, H. Jiang, D. Dillenberger, and G. Nudd, Allocating Non-Real-Time and Soft Real-Time Jobs in Multiclusters, *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 2, pp. 99-112, 2006.
[13] M. Levine, CRAY XT3 at the Pittsburgh Supercomputing Centre, *DEISA Symposium*, Bologna, 4-5 May 2006.
[14] 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

MR Rahmat Zolfaghari is presently working as faculty in Islamic Azad University Hashtgerd Branch, Department of Computer Engineering, Tehran Iran, He is having 12 years experience both in industry and academia, He received his Software Engineering Bachelor (BS) of Shahid Beheshti University in Iran and Software Engineering Master (MS) of Sharif University of technology in Iran, His research interests are Database, Software Design , Modelling and E_Commerce