Non continuous allocation Algorithm for Increasing the Utilization Processor in Multi-Computers Network

Rahmat Zolfaghari

Department of Computer Engineering, Has.C., Islamic Azad University, Hashtgerd, Iran
Dr.Zolfaghari@iau.ac.ir

Abstract: *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.* Two strategies are used for the allocation of jobs to processors connected by mesh topologies: contiguous allocation and non-contiguous allocation. 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. In non-contiguous allocation, a job request can be split into smaller parts that are allocated possibly non-adjacent free sub-meshes rather than always waiting until a single sub mesh of the requested size and shape is available. Lifting the contiguity condition is expected to reduce processor fragmentation and increase Utilization. However, the distances traversed by messages can be long, and as a result the communication overhead, especially contention, is increased. The extra communication overhead depends on how the allocation request is partitioned and assigned to free sub- meshes In large multi-computers, using an allocation algorithm in particular and an efficient scheduled algorithm is very crucial to have the maximum computing power. *Mesh is a widely used architecture in parallel computing systems. Research on efficient allocation of processors to incoming tasks on mesh architecture is very important in achieving the desired high performance. communication overhead is highly dependent on to the manner of free sub-meshes allocation and the manner of recording by the algorithm. In this Research, a fast and efficient noncontinuous allocation algorithm(FENA) with the ability to maintain maximum continuity multiprocessors with mesh topology has been presented for a two-dimensional mesh network .The performance of this algorithm and continuous and noncontinuous allocation algorithms are determined and compared via simulator, Simulation results indicate the improved performance parameters in the FENA(fast and efficient non-continuous allocation)algorithm.*

[Rahmat Zolfaghari. *Non continuous allocation Algorithm for Increasing the Utilization Processor in Multi-Computers Network.* ***J Am Sci 2026;22(7):17-26]. ISSN 1545-1003 (print); ISSN 2375-7264 (online).*** ***http://www.jofamericanscience.org. 02. doi:10.7537/marsjas220726.02***

Keywords: Utilization; multi-computers network; allocation processor; fragmentation; continuous and noncontinuous allocation algorithms.

1- INTODUCTION

In large multi-computers and parallel computing systems, efficient allocation of processors to incoming tasks is necessary to achieve the desired high performance. An efficient allocation algorithm should run fast, have low memory cost and result in short average task (waiting and Completion) time, and high system resource utilization. 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. The processor allocation is responsible for selecting a set of processors in order to run parallel work on them, while job schedule is responsible for the 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 an input job cannot be executed upon the arrival due to the 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 the completion of work. After completion, job is gone out of 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 the implementation of parallel computers with distributed memory due to simplicity, scalability, regularity and easy implementation and has also been used in several machines such as: iWARP [1], IBMBlueGene L [2,3] and DeltaTouchstone [4].

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 the system

size, time for finding sub-meshes for the allocation to input job may be equal to the job execution time. Hence, the development of strategies for minimizing search time (which is also called time allocation) is very important. The 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 [5]) results in high fragmentation. Excessive fragmentation degrades the performance parameters of the system. In order to resolve the fragmentation that occurred in the continuous allocation, discontinuous allocation methods were proposed [6, 7, 8 and 9].

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 the processors utilization in using the system processors and reduces the problem of fragmentation The 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 used predefined local models or mathematical that reduce the efficiency of these algorithms. A discontinuous allocation algorithm that is called fast and efficient non-contiguous allocation algorithm (FENA) has been proposed for a two-dimensional mesh network.

FENA 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.

FENA algorithm is capable to be applied in both two- and three-dimensional mesh multi-computers networks. In this paper, FENA 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 [10].

FENA algorithm has been compared to FF continuous algorithm (that has been used in previous studies) in order to show the 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, FENA discontinuous allocation algorithm will be described and the manner allocation of this algorithm will be exemplified. In Section 4, FENA algorithm and implemented continuous and discontinuous allocation algorithms have been compared from the view point of several important parameters (CompletionTime of Each Job, Waiting Time, Utilization) in performance.

2- BACKGROUND

Definitions and methods of continuous and discontinuous allocation used for multi-computers mesh networks have been reviewed in this section.

A. DEFINITIONS

A two-dimensional mesh M (w, h) is a rectangle of nodes with dimensions of w × h where w is the 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 $\langle a, b\rangle$ where $0 \le$ a $<$ w and $0 \le$ b $<$ h. Node $\langle i, j\rangle$ that is not in the 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 A.1- Two-dimensional sub-mesh S (a, b) in the mesh M (w, h) is a subnet M (a, b) that $0 \le a \le w$ and $0 \le b \le h$. 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-parameter 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 $a \times$ b.

DEFINITION A.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.

DEFINITION A.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 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 β 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: $\vartheta_{\beta 1,T} = \langle 0,0,1,2 \rangle$ ، $\vartheta_{\beta 2,T} = \langle 0,0,3,1 \rangle$ ،$\vartheta_{\beta 3,T} = \langle 2,2,5,5 \rangle$ ، $C_T = \{\langle 2,2,5,5 \rangle, \langle 0,0,3,1 \rangle, \langle 0,0,1,2 \rangle\}$

DEFINITION A.4- According to the input job 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 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. $\delta_{TV} = \langle a', 0, w, h \rangle$, $\delta_{TH} = \langle 0, b', w, h \rangle$ and $a' = w - a + 1 \; and \; b' = h - b + 1$ where, $w \times h$ is the sub-mesh size. A set of reject sub-meshes Δ_T is calculated by adding δ_{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).

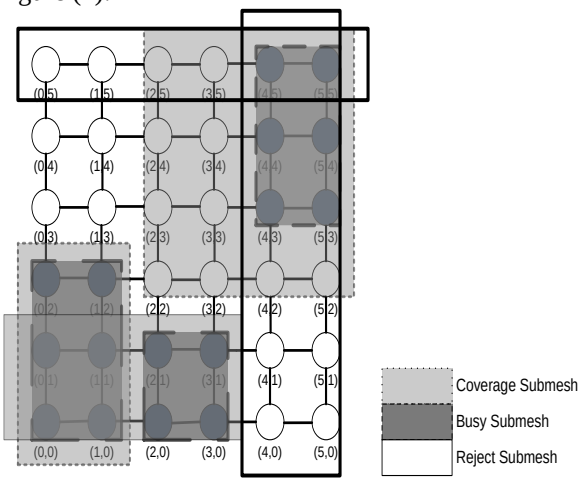

Figure 1 : An example of allocation for T (3, 2).

B. 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.

B.1 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 [11].

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.

C. DISCONTINUOUS ALLOCATION METHODS

With the developments in routing techniques such as wormhole switching, delayed communication had a fewer sensitivity to the 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 .

C.1- PAGING ALLOCATION METHOD

Paging allocation method [12] 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 by paging(*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 $O(a \times b)$.

C.2- MULTIPLE BUDDY SYSTEM(MBS) METHOD

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 square. If a job asks for a processor P, this request is converted to the request in base 4. In this way, $P = d_k \times (2^k \times 2^k) + \cdots + d_0 \times (2^0 \times 2^0)$ 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-partner 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 block 1×1 . Consequently, there will be no fragmentation. Time complexity of this algorithm is O (N) where N is the number of processors in system.

3- FAST AND EFFICIENT NON-CONINUOUS ALLOCATION METHOD(FENA)

Suppose that the input job T (3, 2) has been given to the system and we can do the allocation by use of FENA algorithm.

It is clear in figure 1 that busy sub-meshes include $\beta_2 = \langle 2,0,3,1\rangle$, $\beta_1 = \langle 0,0,1,2\rangle$ and $\beta_3 = \langle 4,3,5,5\rangle$ 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 $\delta_{TH} = \langle 0,5,5,5\rangle$،$\delta_{TV} = \langle 4,0,5,5\rangle$ The main idea for FENA algorithm is to collect information from available rows in sub-mesh network via the coverage sub-meshes made of the busy sub-meshes. Mentioning 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 the 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.

TABLE 1
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)
then*T ←T(w, h), a' ←a-w + 1, b'←b-h + 1*
else*T←T(h, w), a'←a-h + 2, b←b-w + 1*
Step 4. *Based on current B and T, determine* $\vartheta_{\beta,T}$ *and Last_covered[j]* $(1 \le j \le b' - 1) \leftarrow 0$
For *each* $\beta\langle x, y, x', y'\rangle$*, determine* $\vartheta_{\beta,T}\langle x_c, y_c, x', y'\rangle$
Arrange $\vartheta_{\beta,T}$ *s in the increasing order of* x_c
For *each* $\vartheta_{\beta,T}$ *(starting from one whose* x_c *is smallest)*
If $(y_c < b')$
For *each row* $j\ (y_c \le j \le \min(y', b' - 1))$
If $(x_c \le \text{last\_covered}[j] + 1 \le x')$***then*** *last_covered[j]* $\leftarrow x'$
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***

else *i← (last_covered[j] + 1) and* ***go to Step 6****. /*a free submesh is found */*

Step 6.

If *(flag = false)*

Then *S ←<i, j, i + w-1, j + h-1>*

Else *S ←<i, j, i + h-1, j + w-1>*

Allocate S to T and add S to B.

Return success

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

By using three coverage sub-meshes β_3 ,β_2, β_1 we get three sub-meshes $\vartheta_{\beta 1,T} = \langle 0,0,1,2\rangle$, $\vartheta_{\beta 2,T} = \langle 0,0,3,1\rangle$ and $\vartheta_{\beta 3,T} = \langle 2,2,5,5\rangle$ then we arrange them as, $\vartheta_{\beta 1,T}$, $\vartheta_{\beta 2,T}$ and $\vartheta_{\beta 3,T}$.According to the $\vartheta_{\beta 1,T}$, the values *last-covered[j]* for *j=0, 1, 2* equals to 2. For j=0,1 the values of *last-covered[j]* for $\vartheta_{\beta 3,T}$ equals to 3 and with the quantity of *last-covered [2]* is changed and equals to 5. The final value for the last_covered for 5 elements from left to right is in order 0,*0,5,3,3* . As it can be seen, if a node belongs to sub-meshβ, it belongs certainly to sub-mesh C_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 allocate 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, since $last\_covered[4] + 1 < a'$, node $\langle 1,4\rangle$ can be allocated to the job T as a base node. Note that FENA algorithm is more time-saving compared to the other methods.

TABLE 2

PROCEDURE *FENA_ ALGORITHM (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 (α, β) 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 FENA 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 $\boldsymbol{a \times b}$ the last phase is repeated until $\boldsymbol{a \times b}$ 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 $\mathbf{6 \times 2}$. As we see in the figure, there are no free $\mathbf{6 \times 2}$ sub-meshes. Therefore, FENA 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 don,t 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 to 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.

4) SIMULATION

A. VARIABLES OF RESEARCH

The criteria for evaluation of the proposed algorithms are evaluated by comparing them to the standards that have been used in most studies criteria such as:

1. The average Completion(turnaround) Time of Each Job.
2. Average waiting times for each task.
3. Percent of optimal use of system resources(utilization).

These parameters can be well compared with the performance of the proposed algorithm compared to the existing portfolio.

B. LIMITATIONS OF RESEARCH

In this study, we aimed to compare the efficacy of the assumptions we used in most previous studies have been widely used.

1. The distance between the entry actions are independent and have an exponential distribution.
2. By default, the task scheduler log on First-Come-First-Served(FCFS) and, unless the situation is clear.
3. Duringsub-mesh requested by tasks as independent production and distribution is possible. Two different size distributions have been considered in this study. First, a uniform distribution on the interval to the size of the screen, and the power distribution is the exponential distribution with mean value sub-mesh work requested by the size of the mesh.
4. Messages in the network by switching creep are sent using XYrouting.
5. Messages of fixed length (a fixed number of fleets). Respectively, The number of messages generated by the work is exponential distribution.

C. THE RESULTS OF SIMULATION

Here, we represent the results of the assimilation of some contiguous and non-contiguous allocation methods such as Paging (0)**,** MBS and 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 [13].

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 to be in 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 [14,15]. 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 $[1, L]$. 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 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 $t_s = 3$ time unit and the fleet is $P_{len} = 8$ and the *num-mes*=5

The parameters chosen for comparison are: the average turnaround(Completion)time of each job, 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:

$$\text{System Utilization} = \sum_{i=1}^{t} \frac{w \times h - n_i}{(w \times h) \times t} \qquad (1)$$

In this formula n_i is the number of free processors of the system in time ***i*** and ***t*** is the total spent time, and $w \times h$ 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 estimated as follows:

$$\lambda = \frac{N \times T_e}{\text{System Load} \times P} \quad (2)$$

In this formula ***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 λ in the time unit. ***N*** is the average number of the wanted processors by each job, and T_e is the average powered distribution of the implementation time.

C.1- THE COMPLETION(TURNAROUND) 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 FENA 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 FENA has a better performance than FF contiguous allocation for both models of job size distribution. For example, in figure (2) the algorithm FENA 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 have also been shown that by extending the length of packs,the discrepancy of the parameters of FENA efficiency in comparison to the other contiguous and non-contiguous algorithms has shown more improvement.

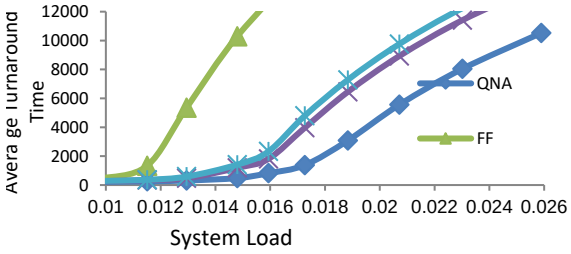

Figure 2: The average completion time of a job according to the system of loading in the *one to all* communicating model with a monotonous distribution of jobdimensions.

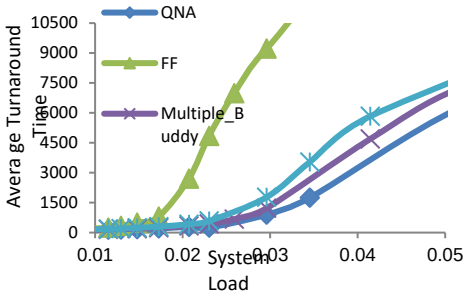

Figure 3: The average completion time of a job according to the system of loading in the *one to all* communicating model with the powered distribution of job dimensions.

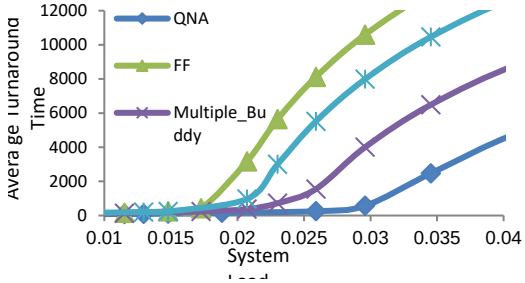

Figure 4:The average completion time of a job according to the system of loading in the *all to all* communicating model with the monotonous distribution of job dimensions.

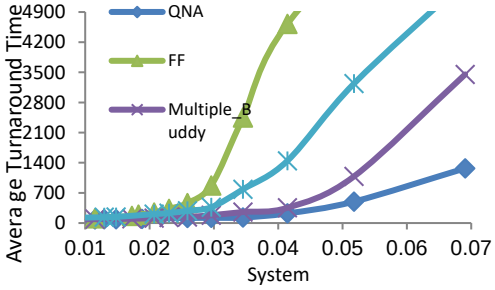

Figure 5: **The average completion time of a job according to the system of loading in the *all to all* communicating model with the powered distribution of job dimensions.**

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, FENA has 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 FENA is %20, %24 and %38 of the average completion time of FF, Paging (U) and MBS methods respectively.

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

C.2- Utilization

Figures 6 and 7 have shown the average productivity of system resources in the allocation algorithms FENA, 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 queue 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 FENA 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.

Likewise, it has been shown that the allocation is contiguously done and after that fragmentation occurred that prevents a good allocation. The average productivity of the 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.

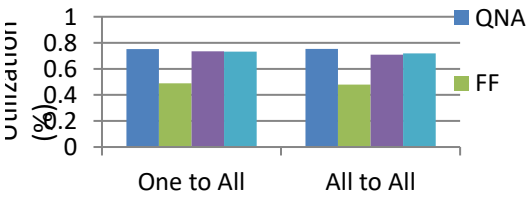

Figure 6: The optimum use of system resources in contiguous and non-contiguous methods for both communicating models with monotonous job dimensions distribution in 16 * 16 sub mesh.

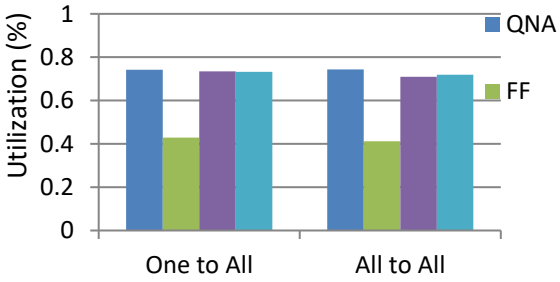

Figure 7: The optimum use of system resources in contiguous and non-contiguous methods for both communicating models with powered distribution of job dimensions in 16 * 16 sub mesh.

C.3- Waiting Time

In figures 8 and 9 the mean waiting time for each job in regard of the system's load is shown for the *all to one* communicating model. The results show that FENA 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 FENA 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 decrease in the overload of allocation; and this shows that the processor allocation in FENA is performed better than the other methods and the logical result is that the waiting time decreases for a job.

FENA 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 FENA in the mean inter-arrival time of jobs with 0,0205 jobs/unit are respectively 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, FENA has a better performance than the other allocation algorithms (for both job size distribution models). For example in figure 11, when the mean inter-arrival time of jobs is 0,05 jobs/unit, the mean waiting time of FENA algorithm will be %19, %27 and %50 in FF, Paging(U) and MBS methods, respectively.

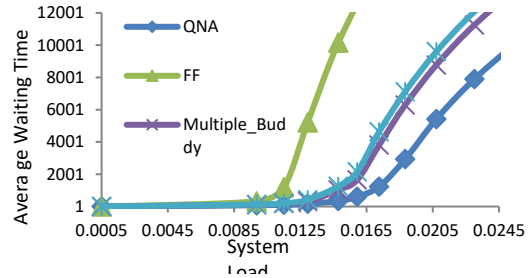

Figure 8: The mean waiting time according to the system's loading in *all to one*communicating model with monotonous job dimensions distribution

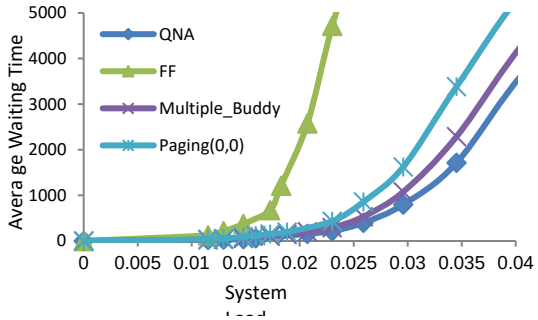

Figure 9: The mean waiting time according to the system's loading in *all to one* communicating model with powered job dimensions distribution

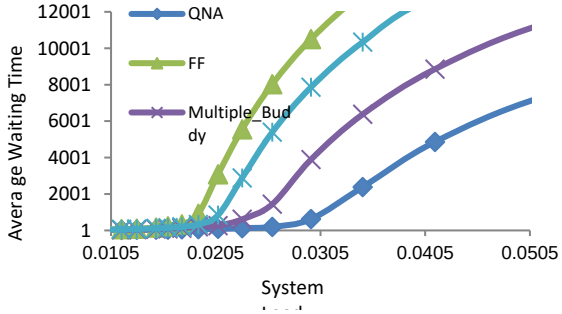

Figure 10:The mean waiting time according to the system's loading in *all to all*communicating model with monotonous job dimensions distribution

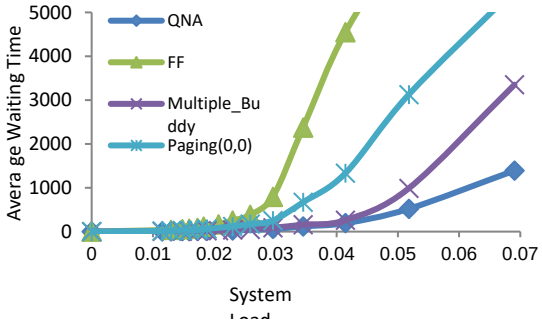

Figure 11: The mean waiting time according to the system's loading in *all to all* communicating model with powered job dimensions distribution.

5- Conclusion and Future Directions

The efficiency of FENA was compared with the efficiency of contiguous and non contiguous algorithms. The results of assimilations have shown that FENA in spite of the available communicating in the net, it has been resulted from the interference of different jobs messages with each other; it increases the efficiency to a great extent. FENA also efficiently takes advantage of the system's resources while keeping the maximum consistency and preventing internal and external fragmentation.

Likewise, the results have considerably shown that FENA 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 FENA has also 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 gets larger, it increases the needs of the programs such as the number of the required processors as well. As a continuation of this research in the future, it would be interesting to assess the suggested allocation strategy in other common multicomputer networks, such as torus networks.

Another possible line for future research is to implement our strategy based on real workload traces from different parallel machines and compare it with our results obtained by means of simulations. Since our country is studying on the production of a national supercomputer, the results of this study can be an effective step to develop this objective.

6- REFERENCES

[1] 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.

[2] 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.

[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] 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.

[5] Zhu Y.; "Efficient processor allocation strategies for mesh-connected parallel computers," Journal of Parallel and Distributed Computing, pp. 328-337,1992.

[6] Zhu Y.; "Efficient processor allocation strategies for mesh-connected parallel computers," Journal of Parallel and Distributed Computing, pp. 328-337,1992.

[7] Chuang P. J., Tzeng N. F.; "Allocating precise submeshes in mesh connected systems," IEEE Transactions on Parallel and Distributed Systems, pp. 211-217,1994.

[8] 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.

[9] Peterson C., Sutton J., and Wiley P.; "iWARP: a 100-POS, LIW microprocessor for multicomputers," IEEE Micro, pp. 26-29, 1991.

[10] Peterson C., Sutton J., and Wiley P.; "iWARP: a 100-POS, LIW microprocessor for multicomputers," IEEE Micro, pp. 26-29, 1991.

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

[12] Peterson C., Sutton J., and Wiley P.; "iWARP: a 100-POS, LIW microprocessor for multicomputers," IEEE Micro, pp. 26-29, 1991.D. G. Feitelson, Workload Modeling for Computer SystemsPerformanceEvaluations, 2007*http://www.cs.huji.ac.il/~feit/wlmod/wlmod.pdf*.

[13] Zhu Y.; "Efficient processor allocation strategies for mesh-connected parallel computers," Journal of Parallel and Distributed Computing, pp. 328-337, 1992.

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

[15] 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.