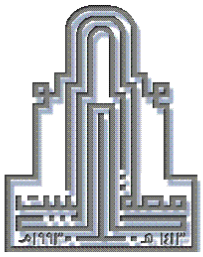

Al al-Bayt University

Computer Science Department

Prince Hussein Bin Abdullah College for Information Technology

The effect of Heavy-Tailed Distribution on the Performance of Non-contiguous Allocation and Job Scheduling Strategies for 2D Mesh-Connected Multicomputers

By

Batool Hmedan Ali Zyoud

Supervisor

Saad Bani-Mohammad

Co-supervisor

Prof. Ismail M. Ababneh

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Science in computer science

Deanship of Graduate Studies
Al al-Bayt University

2016

Dedication

This work is dedicated to my parents, who have always helped me and prayed for me and whose good examples have taught me to work hard for the things that I want to achieve.

This work is also dedicated to my brothers and my sister, who have been a constant source of support and encouragement during my studies.

Acknowledgments

First of all, I owe my gratitude to Allah for his graces and guidance and without his guidance this thesis would not have been possible. I would like to thank my thesis advisors, Dr. Saad Bani-Mohammad & Prof. Ismail M. Ababneh, for their continuous support of my study and research, for their patience, enthusiasm, encouragement, and massive knowledge. Their guidance helped me in all the time of research and writing of this thesis, and I am gratefully indebted to my advisors for their very valuable comments on this thesis. I am also thankful to my family who has been a constant source of love, concern, support and strength at my studies period.

Table of Contents

Table of Figures

Figure.....**Page number**

List of Abbreviations

Abbreviation	Meaning
FCFS	First Come First Served scheduling strategy
GABL	Greedy Available Busy List allocation strategy
MBS	Multiple Buddy Strategy
OO	Out of Order scheduling strategy

The Effect of Heavy-Tailed Distribution on the Performance of Non-contiguous and Job Scheduling Strategies for 2D Mesh-Connected Multicomputers

By
Batool Hmedan Ali Alzyoud
Supervisor
Dr. Saad Bani-Mohammad

Abstract

Various earlier studies have used the uniform distribution for producing job sizes when estimating the performance of allocation strategies. So, the uniform distribution have been used to produce the job sizes for the non-contiguous allocation strategies for different scheduling strategies considered in this thesis. However, different measurement studies have demonstrate that the job size of particular computational jobs can be generated by heavy-tailed distribution because it is more realistic than uniform distribution.

In this thesis, we have investigated the effect of the bounded pareto job size distribution on the performance of the well-known non-contiguous allocation strategies (Random, GABL, MBS, and Paging (0)) using different communications patterns (Near Neighbour, One to all, and Random) and various scheduling strategies (First-Come-First-Served (FCFS), Out-of-Order (OO) and window-based) in a 2D mesh connected multicomputers. Moreover, we have conducted extensive simulation experiments to compare the performance of the bounded pareto job size distribution with the performance of the uniform job size distribution in terms of turnaround time and system utilization by using ProcSimity simulator. The results show that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto. Also, the results for the two job size distributions considered (i.e., uniform distribution, bounded pareto distribution) show that the GABL, MBS and Paging(0)

allocation strategies outperform the Random allocation strategy. The results also reveal that the scheduling strategies OO and window-based improve the system performance over the FCFS scheduling and that the near neighbour communication has the best performance among the communication patterns considered in this research work.

Chapter 1

Introduction

Parallel computers are used in many real-life applications, especially in the fields of science and engineering (Bani-Mohammad, 2008); this is due to their properties like computational power, reliability, and concurrency (i.e., doing multiple things at the same time). Parallel computers can be defined as a group of processors that participate with each other to solve a computational problem (Foster, 1995).

Parallel computers are categorized according to their memory organization into shared memory and distributed memory (Grama, et al., 2003). In shared memory computers, also called multiprocessors, the processors communicate through shared memory (Grama, et al., 2003). In distributed memory computers also called multicomputers, the processors communicate by exchanging messages via interconnection networks (Grama, et al., 2003). There are two types of interconnection networks: static networks, and dynamic networks (Grama, et al., 2003). Static networks, also called direct networks, have direct (point-to-point) communication links between processing nodes. Examples of direct network include the mesh network, the k-aryn-cube, and the hypercube. Dynamic networks, also called indirect networks, consist of communication links and switches that form paths between processing nodes and memory banks. Examples of indirect networks include the bus network and the crossbar switch network (Grama, et al., 2003).

Mesh network is the most recent multicomputer architecture used due to many features such as ease of implementation, simple orderly connection, and high scalability (Grama, et al., 2003). There are many experimental and commercial multi-computers that used 2D mesh network as an interconnection network such as Delta Touchstonea and iWARP (Yoo and Das, 2002).

1.1Processor Allocation and Job Scheduling

Efficient processor allocation and job scheduling are crucial to exploiting the full computing power of a multicomputer (Bani-Mohammad and Ababneh, 2013; Bani-Mohammad, et al., 2007a,b). Processor allocation is a process that selects a set of processors that will execute an incoming parallel job (Bani-Mohammad, et al., 2011), while job scheduling is a process that determines the order in which jobs are executed (Bani-Mohammad, et al., 2006).

There are many processor allocation strategies that have been generated for mesh-connected multi-computers. These strategies can be divided into two main categories: contiguous and non-contiguous. In contiguous allocation the processors allocated to a parallel job are physically adjacent and have the same topology as the underlying multicomputer network, this type of allocation suffers from high processor fragmentation (Bani-Mohammad, et al., 2015). So, the non-contiguous allocation has been proposed as a solution to the fragmentation problem. In non-contiguous allocation, a job can be allocated to separate sub-meshes without need to wait until a single sub-mesh with the requested size and shape becomes available (Bani-Mohammad, et al., 2006). Random, Multiple Buddy, Paging (Lo, et al., 1997) and Greedy Available Busy List (Bani-Mohammad, et al., 2007b) are examples of non-contiguous allocation strategies. Examples of contiguous allocation strategy include Frame Sliding (Chuang and Tzeng, 1994), First Fit (Zhue, 1992) and 2D Buddy (Li and Cheng, 1991). Contiguous allocation and non-contiguous allocation are presented in Figure 1.1.

Fragmentation causes degradation in system performance in terms of job turnaround time and system utilization, where system utilization is the percentage of processors that are utilized over time, while job turnaround time is the time that is spent by job in the system from job arrival to job departure (Ahmed-Chandio, et al., 2013; Bani-Mohammad, et al., 2015). There are two

types of fragmentation: internal and external (Ababneh, 2008; Yoo and Das, 2002). Internal fragmentation occurs when more processors are allocated to a job than it requests, where the external fragmentation occurs when there are sufficient number of free processors to fulfill another job request but they are not allocated to it because they are not contiguous (Lo, et al., 1997).

Figure 1.2 shows an example of the internal and external fragmentation. Figure 1.2(a) represents a job that requested 6 processors and was allocated 16 processors, and as a result there is an internal fragmentation of 0.625. Figure1.2(b) represents an external fragmentation; when the job requested 4 processors, the sub-mesh cannot be allocated because the available processors are not contiguous.

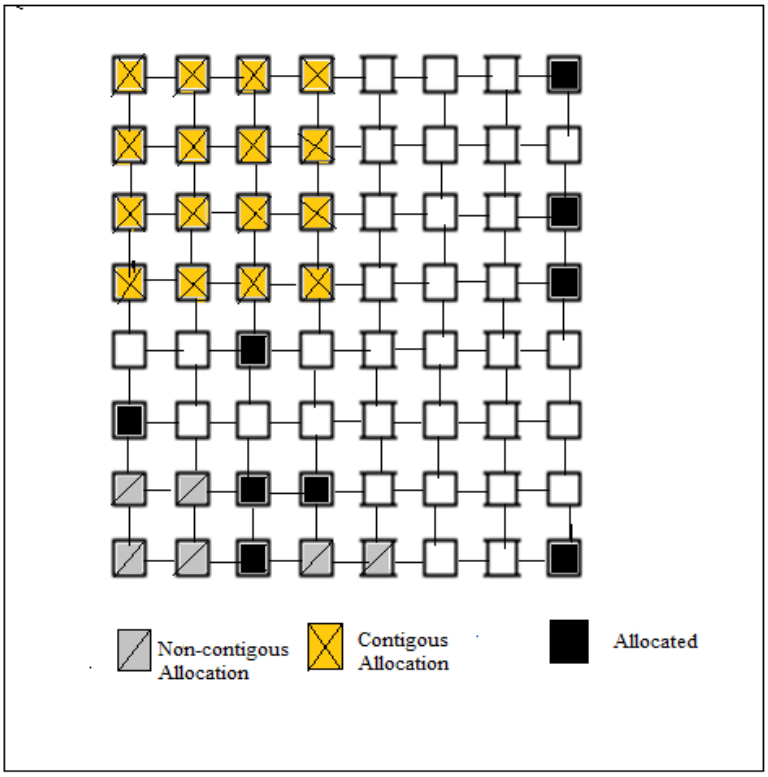

Figure 1.1: Contiguous and non-contiguous allocation.

Job scheduling can also be used to improve system performance in terms of job turnaround time and system utilization (Bani-Mohammad and Ababneh, 2011). The job scheduling algorithm determines the order in which jobs are executed (Bani-Mohammad, et al., 2010). Examples of job scheduling strategies include first-come-first-served (FCFS) (Chiu and Chen, 1999;Yoo, et al., 1997), aggressive out-of-order (OO) (Bani-Mohmmad, et al., 2010), and window-based (Bani-Mohammad and Ababneh, 2011).

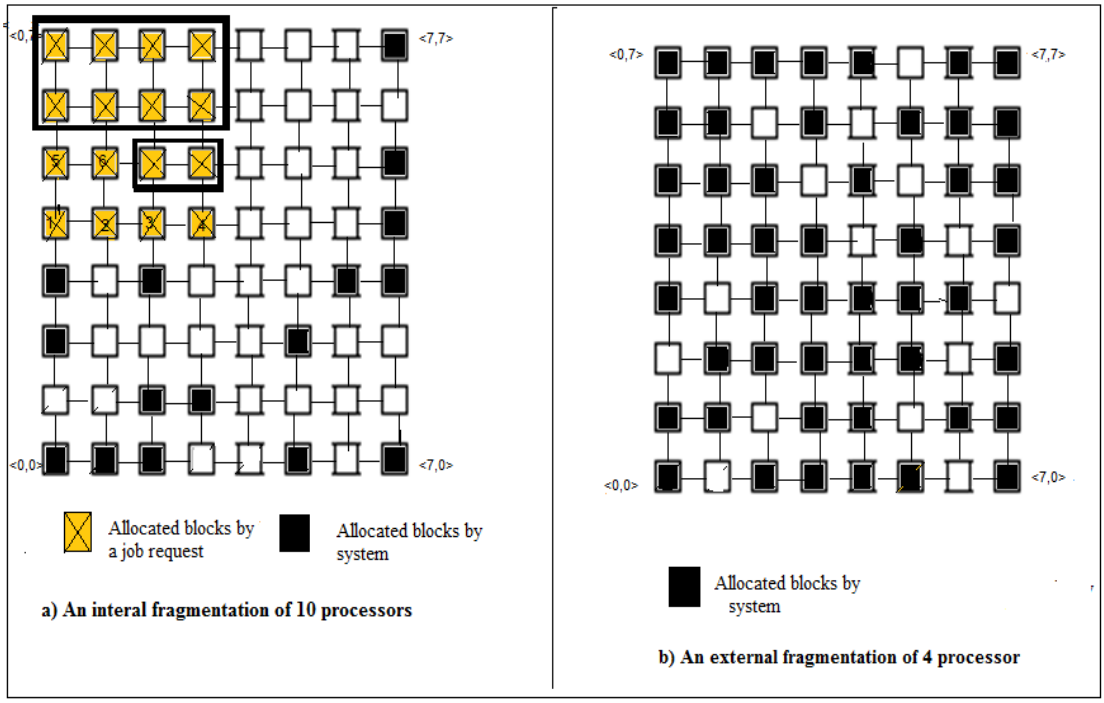

Figure 1.2: Internal and external fragmentation.

1.2 Motivation and Contributions

Many previous studies (Bani-Mohammad, et al., 2007a,b; Bani-Mohammad and Ababneh, 2011) have used the uniform distribution for generating job sizes when evaluating the performance of allocation strategies. Therefore, the uniform distribution will be used to generate the job sizes for the non-contiguous allocation strategies considered in this thesis when evaluating their

performance properties for different scheduling strategies. However, many measurement studies (Crovella and Lipsky, 1997; Harchol-Balter, 1999) have conclusively proved that the job size of particular computational jobs can be represented by heavy-tailed distributions; which mean that many jobs are short and fewer are long. Heavy-tailed distributions can capture this variability and behave quite differently from the uniform distribution (Bani-Mohammad, 2009; Crovella and Lipsky, 1997; Harchol-Balter, 1999).

Performance is measured in terms of average job turnaround time and the mean system utilization (Ahmed-Chandio, et al., 2013).The communication pattern that is used by applications can have a great impact on the performance of the non-contiguous processor allocation in multi-computers (Bani-Mohammad and Ababneh, 2013).

The authors in (Bani-Mohammad and Ababneh, 2013) have studied the effect of uniform job size on the performance of well-known non-contiguous processor allocation algorithms using simulation considering several communication patterns and the results have shown that the Greedy Available Busy List strategy (GABL) has the best performance in terms of job turnaround time for the common communication patterns (i.e., Near Neighbour, Ring, and Random).

To the best of our knowledge, there is no study that considers the impact of heavy-tailed job size distribution on the performance of the non-contiguous allocation strategies. In this thesis, we have implemented different simulation experiments to evaluate the performance of the existing well-known non-contiguous allocation strategies in the context of jobs with sizes that follow both heavy-tailed distributions (i.e., distributions whose tail dropped like a power law) and uniform distribution considering different communication patterns (Near Neighbour, One to all,

and Random) using different scheduling strategies (First-Come-First-Served (FCFS), Out-of-Order (OO) and window-based). Our results show that the performance of the non-contiguous allocation strategies is improved when the distribution of job size is heavy-tailed, and also the scheduling strategies OO and window-based improve system performance over the FCFS scheduling.

1.3 Structure of the Thesis

- Chapter 2 presents a background and preliminaries; then it provides an explanation of the existing non-contiguous allocation strategies that are considered in this thesis, and it describes the method of the study used in this research.
- Chapter 3 introduces the system model and the proposed method; it describes in details heavy tailed distribution (i.e., bounded pareto distribution) and its parameters along with the simulation parameters.
- Chapter 4 reports the results of a comprehensive performance study of the existing non-contiguous allocation strategies.
- Chapter 5 includes conclusions of this research and the possible directions for future research.

The main aim of this chapter is to provide a description of well-known non-contiguous allocation strategies that have been proposed for 2D mesh, and the simulation tool that is used in this study.

This chapter is organized as follows. Section 2.1 describes the non-contiguous allocation strategies considered in this thesis. Section 2.2 provides a description of the scheduling strategies considered in this thesis .Section 2.3 provides a description of the switching methods. Section 2.4 provides a description of the routing algorithms. Section 2.5 provides a brief description of the simulation tool used. Section 2.6 provides the justification of the method of the study.

2.1 Non-Contiguous Allocation Strategies

In this section we present some general allocation definitions for the allocation strategies that have been adopted in previous studies (Bani-Mohammad, 2008; Hamdan, 2010).

The target system is a $W \times H$ 2D mesh-connected multicomputer, where W is the width of the mesh and H is the height of the mesh. Every processor in the mesh is represented by the coordinates (x, y), where $0 \leq x < W$ and $0 \leq y < H$.

Definition2.1: A sub-mesh S is represented by the coordinates$(\mathrm{x}, \mathrm{y}, \mathrm{x}', \mathrm{y}')$, where (x, y) represents the lower left corner of the sub-mesh, and it is called the base node of the sub-mesh, while $(\mathrm{x}', \mathrm{y}')$ represents the upper right corner of the sub-mesh, which is called the end node of the sub-mesh. The size of $\mathrm{S}(\mathrm{w}, \mathrm{h})$ is $\mathrm{w} \times \mathrm{h}$ processors.

Definition2.2: An allocated sub-mesh is one whose processors are allocated to a parallel job.

Definition2.3: A free sub-mesh is one whose processors are unallocated to a parallel job.

Definition 2.4: A busy list can be defined as a list that contains all sub-meshes that are presently allocated in the mesh, while a free list is one that contains all free sub-meshes in the mesh.

The non-contiguous allocation strategies that are considered in this study are as follows:

2.1.1 Random Allocation Strategy:

Random allocation strategy (Lo, et al., 1997), is the simplest non-contiguous strategy where the request for a specific number of processors is satisfied by the same number of randomly chosen processors. Random strategy gets rid of the fragmentation since all jobs are assigned the requested number of processors, if available. However, it suffers from high communication interference between jobs because there is no contiguity imposed under this strategy.

2.1.2 Multiple Buddy Strategy(MBS):

Multiple buddy strategy is a developed and improved form of the 2D buddy strategy that was proposed by (Lo, et al., 1997). MBS eliminates fragmentation by allowing a various contiguous blocks to be allocated to a job non-contiguously. In MBS, the mesh network is divided into non-overlapped square sub-meshes with the side lengths equal to the power of 2. In MBS, a request for K processors is represented as abase 4 number of the form:$k = d_m \times 2^m \times 2^m + d_{m-1} \times 2^{m-1} \times 2^{m-1} + \ldots\ldots\ldots + d_0 \times 2^0 \times 2^0$.

This strategy tries to satisfy the request with blocks of size $2^m \times 2^m$, $2^{m-1} \times 2^{m-1}$, and $2^0 \times 2^0$. If a required block is not available, MBS recursively searches for larger block and iteratively breaks it down into smaller blocks. But if this process fails, MBS breaks the request block into 4 smaller requests and the searching process is repeated. In this strategy, external fragmentation can be eliminated because the large requests can always be broken down into 1×1 blocks. MBS consists of the following five parts: system initializing that splits the 2D mesh network into initial blocks at system startup. Request factoring algorithm which is responsible for converting and factorizing a request of size n into a number of needed buddies. Buddy generation algorithm

where a large block is divided into many smaller blocks. Allocation algorithm, and de-allocation algorithm, which responsible for allocation and deallocation for the requested processors.

Figure 2.1 shows an example of 2D mesh of size16×16; a white square represents a free processor, while a black square represents an allocated processor. Assume a request of 120 processors has been received to the system. Because a block of size 16×16 is not available in the mesh system, MBS breaks the request for a block of size16×16 into four requests for blocks of sizes 8×8, then a request for a block of size 8×8 can be broken into four requests for blocks of size 4×4, also a request for a block of size 4×4 can be broken into four requests for blocks of sizes 2×2. MBS needs one block of size 8×8, three blocks of size 4×4 and two blocks of size 2×2 to allocate the job request of size 10×12.

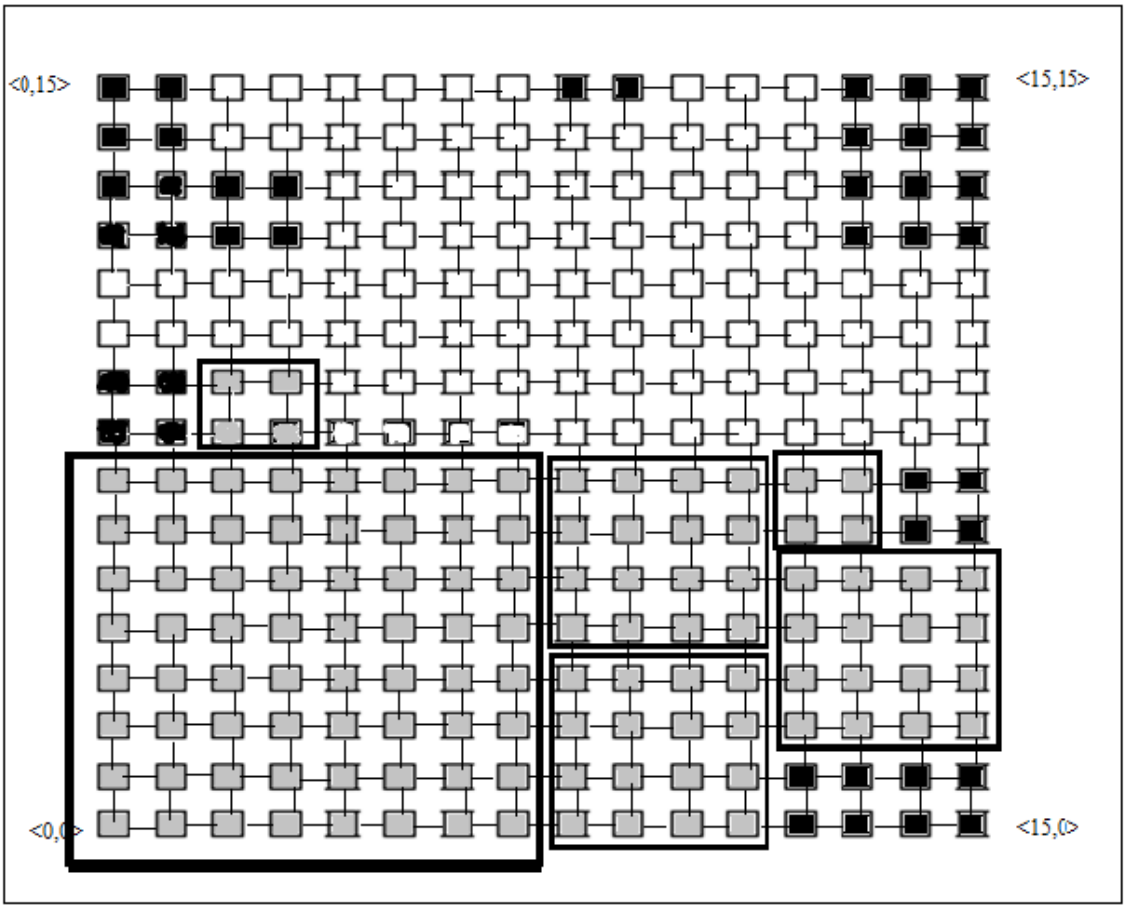

Figure 2.1: MBS accommodating a10 × 12 allocation request in a 16 × 16 mesh.

Next, suppose that the mesh system receives a request for 60 processors(5×12). A block of size 8×8 is not available in the mesh system. So, MBS searches for a free block of size 16×16 and breaks it down into buddies of smaller blocks size. Because MBS failed to find the larger block, it breaks the request for 60 processors(5×12) into four requests for blocks of size 4×4. A request for a block of size 4×4 can also be broken into four requests for blocks of sizes 2×2. MBS requires three blocks of size 4×4 and three blocks of size 2×2 to allocate the job request 5×12. MBS succeeded in allocation and the three blocks of size 4×4 are: (4, 8, 7, 11),

(8, 8, 11, 11), and (12, 8, 15, 11) respectively, while the three blocks of size 2×2 are: (0, 10, 1, 11), (2, 10, 3, 11), and (4, 12, 5, 13). The 6 blocks are assigned to the job as illustrated in Figure 2.2.

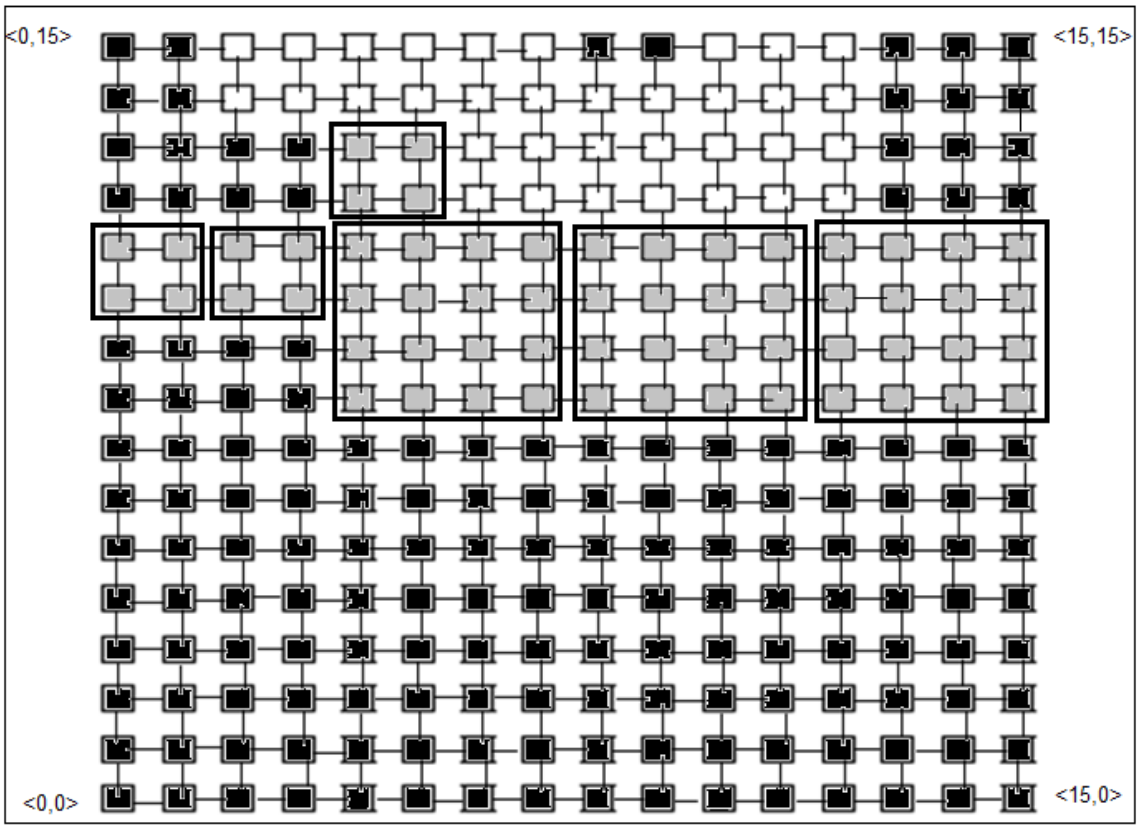

Figure 2.2: MBS accommodating a 5×12 allocation request in a 16×16 mesh.

Figure 2.3 shows how MBS strategy can deal with the job request of $\mathbf{10 \times 2}$ processors when it is arrives to the mesh system. To satisfy this job request, MBS allocates five blocks of size $\mathbf{2 \times 2}$ to the job request.

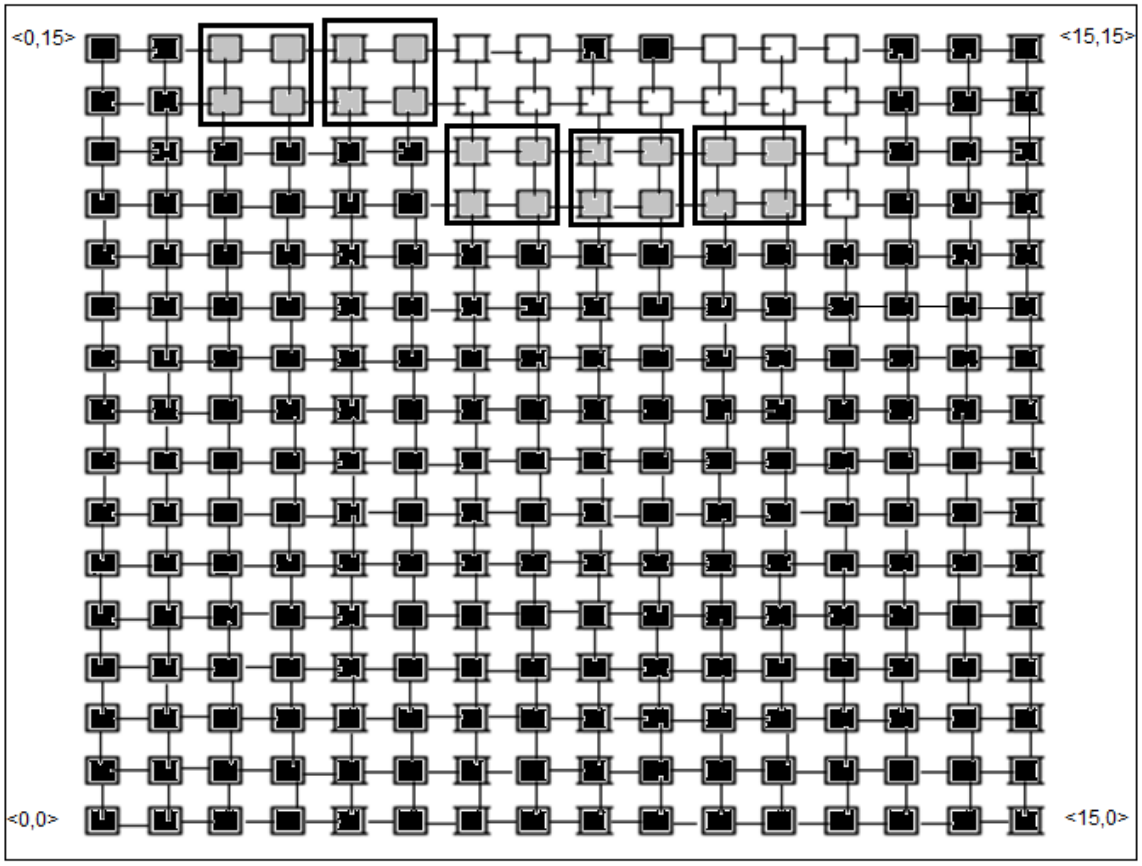

Figure 2.3: MBS accommodating 10 × 2 allocation request in 16 × 16 mesh.

2.1.3 Paging Allocation Strategy:

Paging allocation has been proposed in (Lo, et al., 1997), it initially splits the entire mesh into square pages that are sub-meshes with equal side's length of $2^{sizeindex}$, where $sizeindex$is a positive integer. A request for k processors can be achieved by allocating free pages until the number of requested processors is allocated. The number of pages that a job request is calculated by this formula: $\lceil (a \times b)/psize \rceil$,where $psize$is the size of the page, and a and b are the side lengths of the requested job. An indexing scheme (i.e., row-major, shuffled row-major, snake-like, and shuffled snake-like indexing) determines the order of free pages to be used in searching. The applied indexing scheme keeps some degree of contiguity among allocated pages. To keep track of unallocated pages, paging strategy uses an ordered list. Each entry of this list contains

the page's row and column indices with a unique order index. A paging strategy is marked as paging ($sizeindex$), for example, paging(2) means that the pages are sub-meshes of size 4×4. An internal fragmentation happens when the $sizeindex$ is greater than zero. In this thesis, we adopt row-major indexing scheme because the other indexing scheme have little effect on the performance of Paging. Furthermore, we deem a $sizeindex$=0 (i.e., Paging(0)) because this value of $sizeindex$ eliminates internal processor fragmentation.

To explain the allocation process in Paging strategy, consider the system state shown in Figure 2.4. Assume a job request of size 10×12 arrives to the mesh system; Paging strategy initially scans the entire mesh, searches for free pages and allocates them to the requested job. Paging (0) divides the mesh system into pages each of size 1×1, thus to allocate the job request, 120 free pages are requested. Paging strategy searches for these pages in row major scheme, it succeeds to allocate the incoming job request as shown in Figure 2.4.

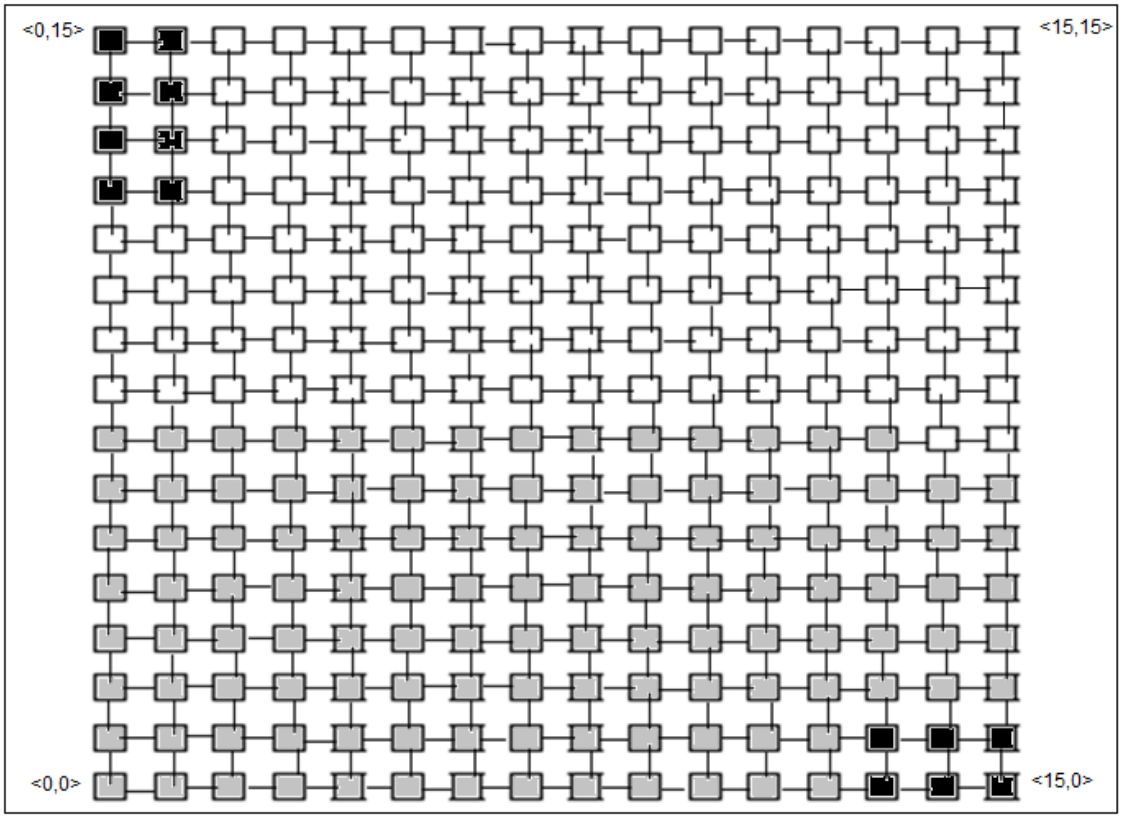

Figure 2.4: Paging(0) accommodating a 10×12 allocation request in a 16×16 mesh.

Suppose a job request of size $\mathbf{5 \times 12}$ arrives to the system. Paging strategy tries to find and allocate 60 pages. Searching begins from left to right and from bottom to top, searching for free pages; Paging strategy succeeds to allocate the job request $\mathbf{5 \times 12}$ as shown in Figure 2.5.

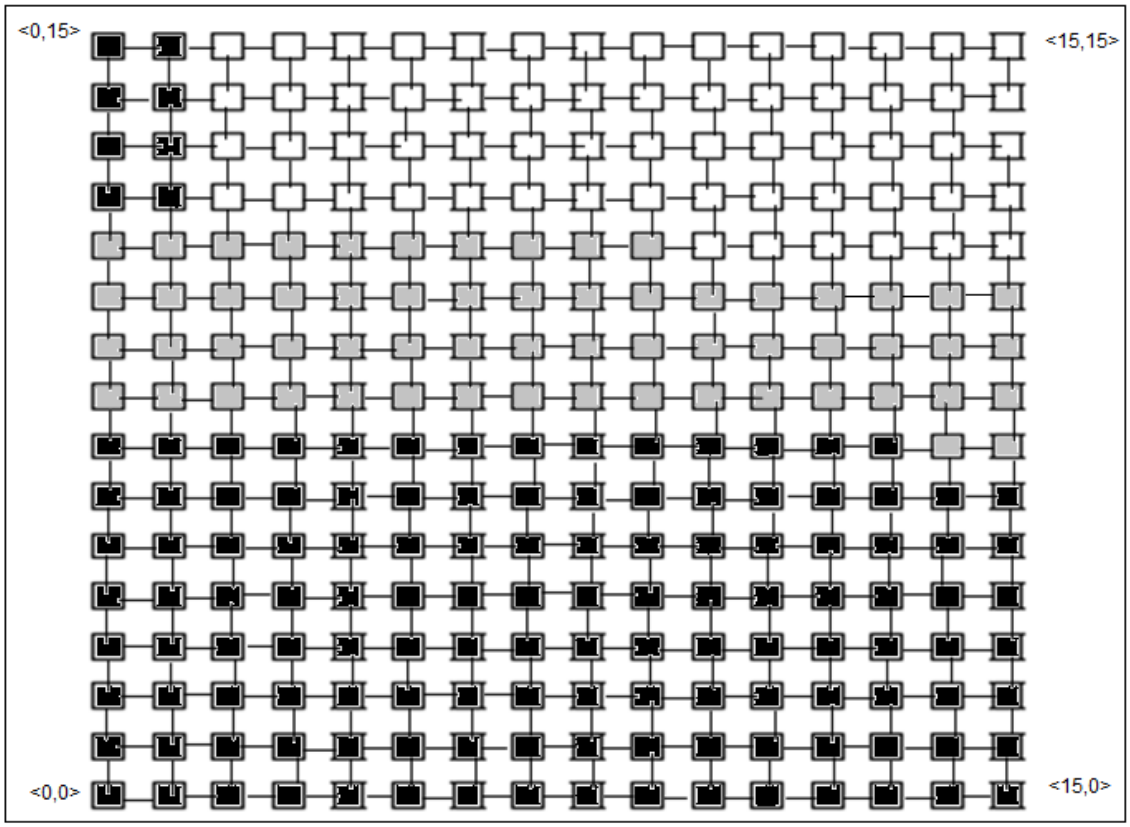

Figure 2.5: Paging(0) accommodating a 5 × 12 allocation request in a 16 × 16 mesh.

2.1.4 Greedy Available Busy List(GABL):

GABL (Bani-Mohammad, et al., 2007b) is the best non-contiguous allocation strategy among the various non-contiguous allocation strategies considered in this thesis. This is due to its ability in decomposing the allocation request in a way that differs from other allocation strategies; it decomposes the allocation request based on the sub-meshes available for allocation so as to achieve a high degree of contiguity among processors allocated to a job. This decreases the number of sub-meshes allocated to a job, and minimizes the distance traversed by messages, which reduces the communication overhead.

GABL tries to allocate a job request $J(\alpha, \beta)$contiguously, where α and β are the dimensions of the job request. If the allocation process fails, the job request is rotated and the allocation process is repeated again. If the allocation fails for the original job request and its rotation form, the largest free sub-mesh $S(w, h)$ in the system that can fit inside the job request $J(\alpha, \beta)$ is allocated. Then GABL attempts to allocate the next appropriate largest sub-mesh whose side lengths do not surpass the corresponding side lengths of the previous allocated sub-mesh, this step is repeated until the requested number of processors is allocated to the job. The allocated sub-meshes are stored in a busy list, that is updated after each allocation and de-allocation operation.

To explain the allocation process in the GABL strategy, consider the system state shown in Figure 2.6, and suppose a job request of size 10×12 arrives to the mesh system. GABL strategy always tries to allocate any job request contiguously. It checks the mesh system and searches for a free sub-mesh of size 10×12, the sub-mesh $(0,0,9,11)$ is found and GABL allocates the job request of size(10×12) contiguously.

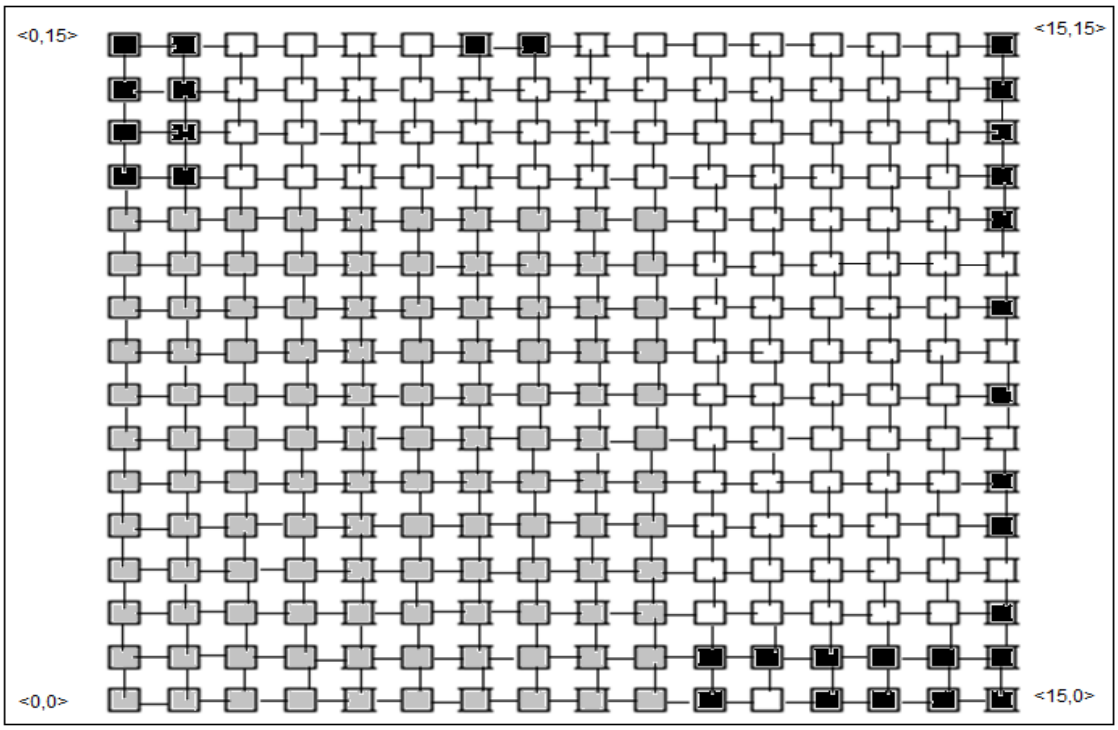

Figure 2.6: An example showing how GABL deals with a job request of size 10×12.

Another example is represented in Figure 2.7, assuming that the mesh system receives a job request of size 5×12. The sub-mesh (10, 2, 14, 13) is found and GABL allocates the job of size 5×12 contiguously.

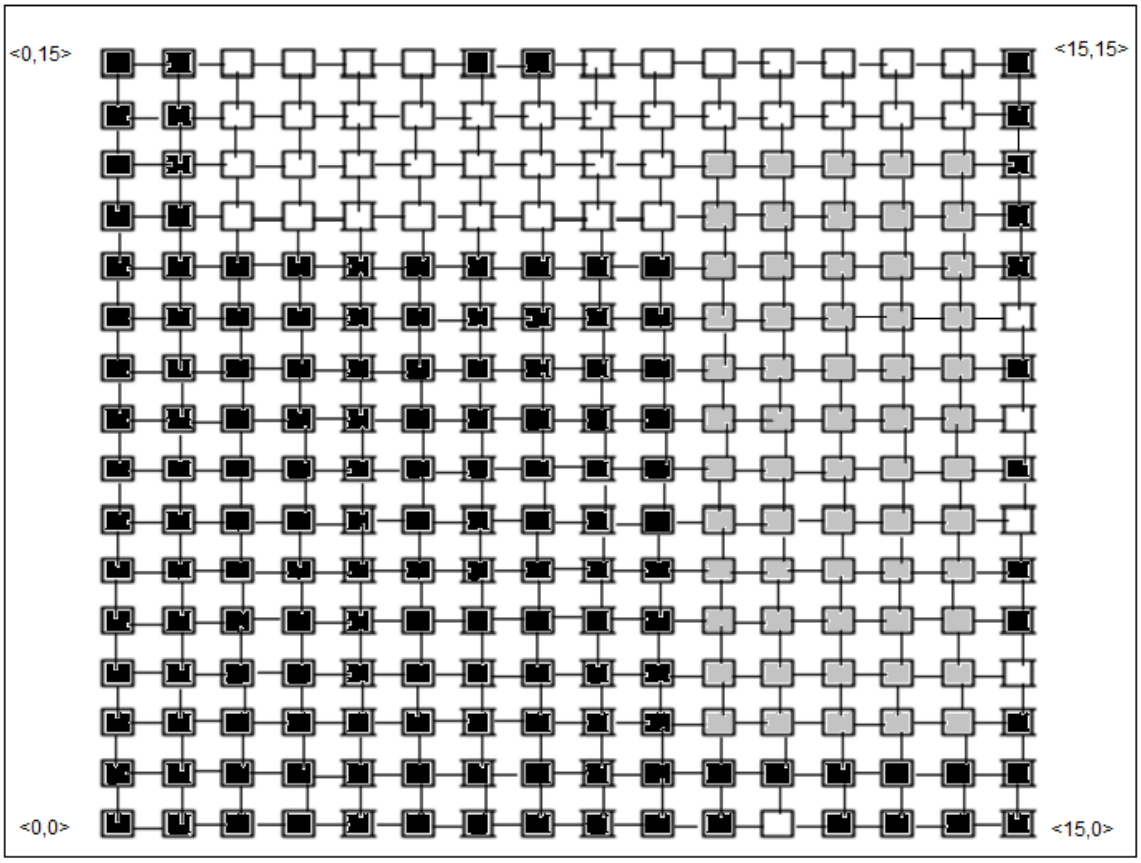

Figure 2.7: An example showing how GABL accommodates a job request of size 5×12.

In Figure 2.8, if a job request of size 10×2 is received by the mesh system, then the GABL strategy tries to allocate the job request contiguously; so it checks the mesh system and searches for a free contiguous sub-mesh of size 10×2. The sub-mesh 10×2 is not found, then GABL subtracts one from the maximum length of the side lengths of the job request until a free sub-mesh is available, GABL finds a sub-mesh of size 8×2, then the subtraction process is repeated until the job request is satisfied. So the 4×1 submesh is allocated.

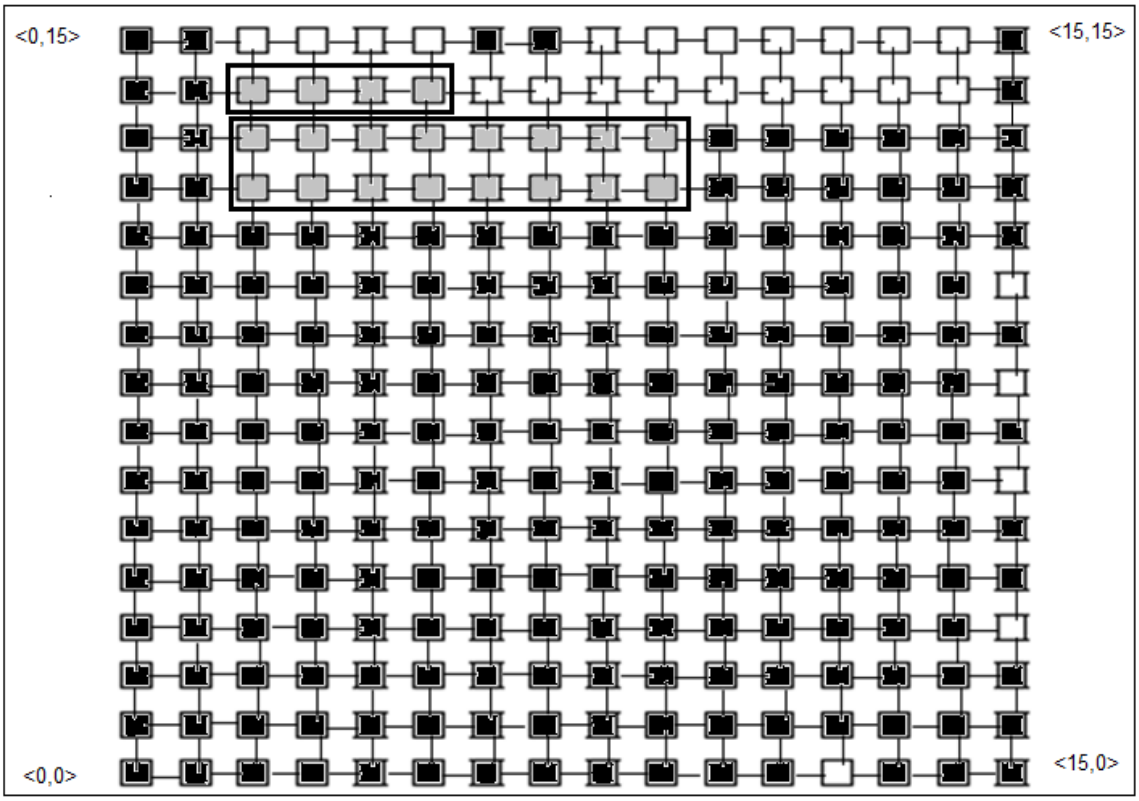

Figure 2.8: An example showing how GABL accommodates a job request of size 10×2.

2.2 Scheduling Strategies

There are many job scheduling strategies that are considered in this research such as first-come-first-served(FSFS) (Yoo, et al., 1997), aggressive out-of-order (OO) (Bani-Mohmmad, et al., 2010), window-based(Bani-Mohammad and Ababneh, 2011).

In FCFS, the jobs stay in a FIFO queue in their incoming order and the first job in the queue is served before any other job. One advantage of this strategy is its fairness. However, a job may cause a delay to the subsequent jobs if the free processors do not satisfy the job's request(Yoo, et al., 1997).

In OO scheduling, the jobs are considered for allocation as they arrive and in their arrival sequence without having to wait for the head of the waiting queue to be served. A large job at the

head of the waiting queue may suffer extreme delays if small allocation requests continue arriving (Bani-Mohmmad, et al., 2010).

The Window based job scheduling scheme is proposed to solve the problem that the OO suffers from. A window-based job scheduling scheme uses a window of sequential jobs that starts with the existing oldest waiting job. The width of the window corresponds to the look ahead of the scheduler, and within this window the jobs will be considered for allocation and execution (Bani-Mohammad and Ababneh, 2011).

2.3 Switching Methods

Switching methods are responsible for determining the way messages are handled as they travel through intermediate nodes. Switching takes place in the router and consists of the receipt of a message, determining the appropriate output channel, and then sending the message through this channel (Bani-Mohammad, 2008; Lo, et al., 1997). There are many switching techniques that have been used in multicomputer networks like store-and-forward, virtual cut through, and wormhole switching (Hamdan, 2010; Lo, et al., 1997). In this section, we presented a summary for these techniques.

2.3.1 Store-and-forward Switching:

In store-and-forward switching, a message is partitioned into fixed-length packets that are routed from source to destination. Each packet includes a header that contains the data needed for routing the packet. An entire packet is completely stored in each intermediate node before it is transmitted to the next node. Store-and-forward switching is useful when messages are short and more frequent, so that the full utilization of the communication link can be achieved. However, store-and-forward switching technique has two major defects: memory consuming due to large

buffer spaces that are required to store entire packets and time consuming due to the time that is required to transmit a message which is proportional to the distance between the source and destination nodes (Bani-Mohammad, 2008; Hamdan, 2010).

2.3.2 Virtual cut-through Switching:

Virtual cut-through is an improved form of store-and-forward switching to reduce the amount of time spent in transmitting data and full buffer requirement. In virtual cut-through switching, a message header (i.e., the part of the message that contains routing information) is checked upon arrival at an intermediate node. If the next required channel is busy, the message is entirely stored at the intermediate node. Otherwise, it is forwarded to the next node without needing to buffer it. In this switching method, the network latency is reduced. However the nodes must provide sufficient buffer spaces for all blocked messages passing through it and multiple messages may become blocked simultaneously, so a very large buffer space is required at each node (Bani-Mohammad, 2008; Hamdan, 2010).

2.3.3 Wormhole Switching:

Wormhole switching is an improved switching form of virtual cut-through switching. Wormhole switching works by forwarding the header of the packet directly from the source node to the next node rather than buffering a packet completely in a node and then transmitting it to the next node. A packet is partitioned into a sequence of fixed-size parts, called flits (flow control digits). A flit is the smallest unit of data transmission. The header flit controls the route. When a node checks the header flit of a message, it chooses the next node on the route and starts forwarding flits to that node. The remaining message flits follow the header over the same path to the destination in a pipelined fashion. Wormhole switching has motivated the use of non-contiguous

allocation in multi-computers that have long communication distances, such as meshes (Bani-Mohammad, 2008; Hamdan, 2010) because the message latency (i.e., the time taken by a message to transfer from source to destination) is not sensitive to the distance between the source and destination (Bani-Mohammad, 2008; Hamdan, 2010).

2.4 Routing Algorithms

Routing algorithms are necessary to the efficient interconnection networks process because they are responsible for determining the path that is used by each message in the network (Bani-Mohammad, 2008; Hamdan, 2010). A perfect routing algorithm minimizes the number of hops that are required for packets to reach their destination to minimize the latency of the network; furthermore, it should be able to exhaust deadlock situations (Bani-Mohammad, 2008; Hamdan, 2010).

Routing algorithms can be classified into two types: deterministic or adaptive, upon their ability to alter routing paths depending on the dynamic network conditions (Bani- Mohammad, 2008; Hamdan, 2010). In deterministic routing, a message always uses the same path between the source and destination; intermediate nodes are unable to redirect messages to any alternative paths. However, adaptive routers select the route that the packet will transfer through it depending on the current dynamic conditions of the network such as the presence of congestion or failures (Bani-Mohammad, 2008; Hamdan, 2010). Thus it provides multiple paths from the source to destination. Dimension-ordered routing is a well-known example of deterministic routing where messages cross network dimensions in a pre-defined order. For mesh networks, dimension-ordered routing gets rid of deadlock problem. Dimension-ordered routing in 2D mesh is indicated as XY routing (Bani-Mohammad, 2008; Hamdan, 2010). The two dimensions of a

mesh network are titled as X and Y. A message firstly is transmitted in the X dimension and then in the Y dimension. Figure 2.10 represents an example of XY routing among source node and destination node in 8×8 mesh network. Dimension-ordered routing is used in this research when examining the performance of the non-contiguous allocation algorithms(Bani-Mohammad, 2008; Hamdan, 2010).

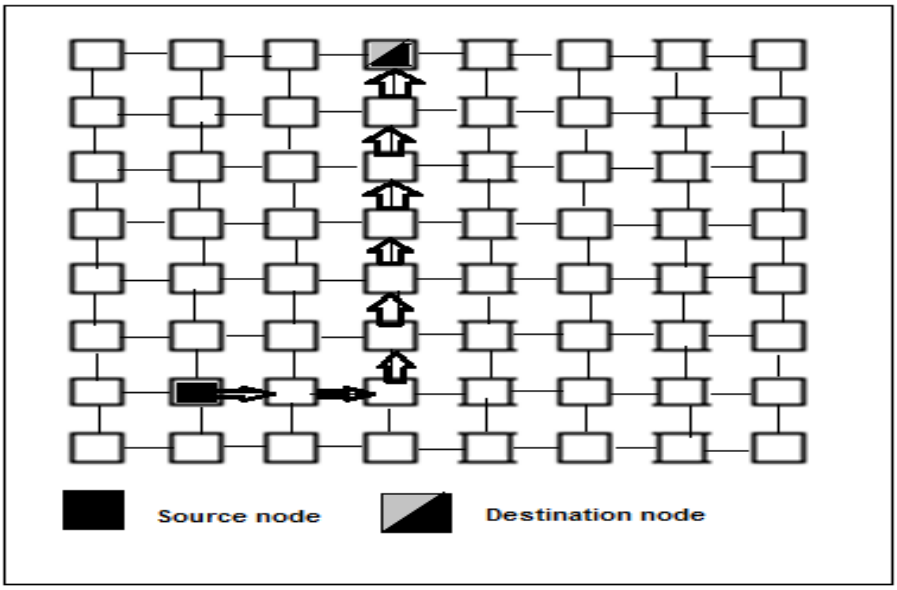

Figure 2.9: Dimension-ordered(XY)routing in8× 8 mesh network.

2.5 Communication Patterns

Processors allocated to a parallel job communicate with each other based on some communication patterns (Bani-Mohammad, 2008; Hamdan, 2010). Three communication patterns have been used in this research, to estimate the performance of the non-contiguous allocation strategies.

In one to all communication pattern, a randomly chosen processor sends a message to all other processors allocated to the same job. This communication pattern causes an enormous message congestion bottleneck at the sending processor, which causes degradation in network utilization.

In the random communication pattern, message sources and destinations are a random pair of processors allocated to the same job. This communication pattern shows the case where irregular communications are used in applications, such as the conjugate gradient solver and the Euler solver (Bani-Mohammad, Ababneh, 2013).

Near Neighbor Communication (NN) has two phases, mapping and sending. In this communication, the processors allocated to a job are mapped to a virtual two-dimensional array of a size that is equal to the job's request. All these processors communicate with their virtual neighbors. NearNeighbor communication has been used in this research because it is a popular communication pattern, particularly for physical phenomena simulations such as heat and wave propagation(Bani-Mohammad, Ababneh, 2013).

2.6 Simulation Tool (ProcSimity Simulator)

This section provides a description of the common simulation tool, which is called ProcSimity (Windisch, et al., 1995). ProcSimity is a software tool that is applied for processor allocation and job scheduling in distributed memory multi-computers (Bani-Mohammad, Ababneh, 2013; Grama, et al., 2003). ProcSimity was written in the C programming language, it is an open-source and includes detailed simulation of important communication patterns and operations for chosen multi-computer networks (Bani-Mohammad, 2008; Bani-Mohammad, Ababneh, 2013; Lo, et al., 1997).

The main aim of the ProcSimity is to give an appropriate environment for performance analysis of processor allocation and job scheduling algorithms. The ProcSimity simulator specifies the target machine environment that includes the network topology, routing, and flow control mechanism, and it involves the chosen of a scheduling and an allocation algorithm from a set of

given algorithms (Bani-Mohammad, 2008; Windisch, et al., 1995). Also, allocation strategies, scheduling strategies, and communication patterns can be added to the simulation tool.

When ProcSimity simulates a mesh-connected multicomputer, various user jobs that reach to the system, request free sub-meshes. If the number of free processors in the mesh system is not sufficient to fulfill the job request, or there are other waiting jobs in the queue, the job is transferred to the waiting queue. Scheduling strategy selects which job to be executed from the waiting queue, and then the processor allocation algorithm selects and allocates the set of processors on which the job will execute. The allocated processors may be contiguous or non-contiguous according to the allocation strategy that is used (Bani-Mohammad, 2008; Windisch, et al., 1995).

2.7 Justification of the Method of Study

In this study, many simulation experiments have been conducted to examine the effect of heavy-tailed job size distribution on the performance of the non-contiguous allocation strategies. This section justifies concisely the reasons to choose the simulation as the method of study in this research.

In this research, simulation has been chosen as the method of study because it can provide a good level of reliability in order to mimic the behavior of the real system (Bani- Mohammad, 2008; Hamdan, 2010). There are three main approaches for examining the performance of any system: measurement, analysis, and simulation (Bani-Mohammad, 2008; Hamdan, 2010). In measurement approach, the system is fully implemented and its performance is measured directly. While the analysis approach uses mathematical analysis from first principles to evaluate the system. While in simulation approach, a model for the real system is designed and tests are

performed either for understanding the behavior of the system or for examining several strategies. Simulation is mostly more appropriate because it includes fewer approximations than conventional approaches (i.e. measurement, analysis); it is often used because it is the only applicable alternative.

Analysis may be too complicated, or may need too many simplifying assumptions that limit their applicability to a restricted number of scenarios. Measurement approach is sometimes impossible because either the system does not exist or it would consume too much time. In other situations it is irrelevant because we cannot change the configuration as required (Bani-Mohammad, 2008; Hamdan, 2010).

We have used the ProcSimity simulator in this research. ProcSimity simulator has already been developed and mostly validated (Windisch, et al., 1995). Extensive simulation experiments were executed so as to compare the performance of the non-contiguous allocation strategies that are considered in this research work.

3.1 Introduction

In this research, extensive simulation experiments have been executed for several system loads to study the impact of heavy-tailed job size on the performance of the non-contiguous allocation strategies. This chapter describes in details heavy tailed distribution (i.e., Bounded Pareto distribution) and its parameters along with the simulation parameters.

3.2 Heavy Tailed Distributions

Heavy tailed distributions have a quite different behavior from other distributions (i.e., normal distribution and exponential distribution). The exponential distribution and heavy tailed distribution have the same "ski-slope" shape when drawn on non-logarithmic scales, however the tails for heavy tailed distributions decline relatively slowly so the probability of very large observations happening when sampling random variables that follow heavy tailed distributions is non-negligible. While the exponential distribution drops off at a constant rate (Crovella and Lipsky, 1997; Harchol-Balter,1999).

Many earlier analytic works in computer system design has supposed that the job sizes (services demands) are exponentially distributed. Several policies, algorithms, and general rules of thumb which are currently used in computer systems originated from analysis which assumed an exponentially-distributed workloads (Crovella and Lipsky, 1997; Harchol-Balter,1999). For heavy tailed distributions features, the computing systems designers are increasingly concerned in employing heavy-tailed distributions for generating workloads used in simulation experiments (Crovella and Lipsky, 1997; Harchol-Balter,1999). This is because the values generated based on heavy-tailed distributions are more realistic than those generated based on exponential distributions.

We assume that job sizes show some maximum values. As an outcome, we model job sizes using a distribution that follows a power law and it has an upper bound. This distribution is called Bounded Pareto distribution (i.e., the simplest heavy-tailed distribution) as follow:

$f(x) = \frac{\alpha k^{a}}{1-(k/q)^{\alpha}} x^{-\alpha-1} (k \leq x \leq q).$

Where k is the lower limit of the job size and q is the upper limit of the job size, and α is a parameterthat reflects the variability of job sizes. In the experiments, these parameters are set to k =1, q =16, and α =0.2. A Bounded Pareto distribution shows very high variability when $k \ll q$ and α ≈0.2 as suggested in (Dillenberger, et al., 2006). So, the values of$k$, q, and a have been selected as previously to show this variability. However, when α increases the probability of large values decreases.

3.3 System Model

In this section, we present the simulation parameters that were used. The target mesh modeled in the simulation experiments is square with side lengths L. The jobs are served on First-Come-First-Served (FCFS), Out-of-Order (OO) and window-based scheduling strategies. These strategies have been selected because they have been used in several similar studies and they are common (Bani-Mohammad, 2008; Bani-Mohammad and Ababneh, 2013; Hamdan, 2010).

Several communication patterns have been used in this study. These are the One to All, the Random, and the Near Neighbor (NN) communication patterns. The details of these communications patterns will be provided in Chapter4. Two job size distributions were used, first is the uniform job size distribution, over the interval $[1, L]$, where each job side length is generated independently. The second distribution used is theBounded Pareto job size distribution, over the interval $[1, L]$.

The allocation strategies considered are Multiple Buddy Strategy (MBS) (Lo, et al., 1997),Greedy Available Busy List(GABL) (Bani-Mohammad, et al., 2007b), Paging(0) (Lo, et al., 1997), and Random(Lo, et al., 1997). The execution times of jobs depend on many factors: the time needed for flits to be routed through the node, packet sizes, the number of messages sent, message contention and distances messages traverse. The time values are simulation times, not real times. An important independent input variable in the simulations is the inter-arrival time of jobs, where the inter-arrival time is the delay between the arrivals of two consecutive jobs to the system. The inverse of the mean inter-arrival time is the job arrival rate, which is referred to as the system load. The range of load values, from low to heavy loads, has been determined through experimentation with the simulator, permitting each allocation strategy to reach its upper limits of utilization; where the utilization is the percentage of processors that are utilized over time (Ababneh, 2008; Bani-Mohammad and Ababneh, 2011). The flow control mechanism assumed is wormhole switching. In wormhole switching, the message latency does not sensitive to the distances. The dimension order XY routing is used. The packet size is 8 flits. The routing delay is 3 time units. The mean time between sends is (0.0). The values of these parameters have been used in the previous studies (Bani-Mohammad, et al., 2007a; Bani-Mohammad, 2008; Lo, et al., 1997) and have been recommended in (Windisch, et al., 1995). The number of messages per job is the number of messages generated per iteration of the communication pattern. The number of jobs per run is 1000. Runs are repeated many times to ensure that the relative errors do not exceed 5% with a confidence level 95%.

4.1 Introduction

In this chapter, we study the effect of bounded pareto distribution on the performance of the allocation strategies considered in this research and compare its performance with the uniform distribution performance when these distributions have been used for job sizes. To achieve this goal, extensive simulation experiments were conducted for different communication patterns with different scheduling strategies under both the uniform and bounded pareto job size distributions with various system loads. The main performance parameters observed are the average turnaround time of jobs and the mean system utilization. In the figures that are presented below, the x-axis represents the system load while the y-axis represents the values of the performance metric of interest.

4.2 Average Turnaround Time Results

In Figures 4.1 and 4.2, the average turnaround time of jobs is plotted against the system load for the one-to-all communication pattern and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under FCFS scheduling. It can be observed from these figures that Random strategy performs worse than all other non-contiguous allocation strategies for both job size distributions considered in this research work. This is due to the lack of contiguity among the allocated processors in the Random allocation strategy. However, GABL, MBS and Paging(0) have comparable performance. This is because these strategies maintain a higher degree of contiguity among the allocated processors than that of the Random strategy.

Also, the results reveal that the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. This is because in bounded pareto distribution most jobs have small sizes and fewer jobs have large sizes, while in uniform

distribution many jobs have a large size, which means that the bounded pareto is more realistic than the uniform distribution. So, the average turnaround time of non-contiguous allocation under FCFS in bounded pareto is decreased and consequently leads to an improvement in system performance.

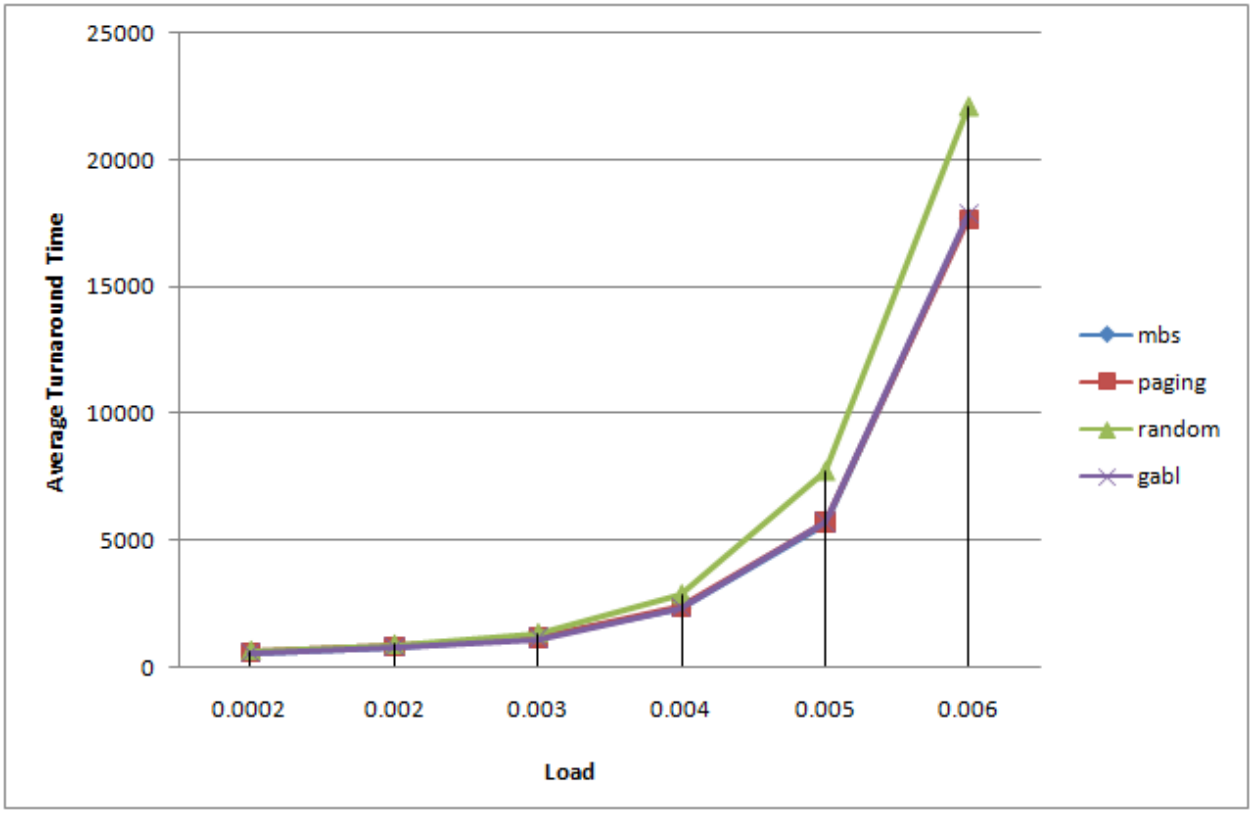

Figure 4.1: Average turnaround time of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the FCFS strategy and uniform job size distribution in a 16 × 16 mesh.

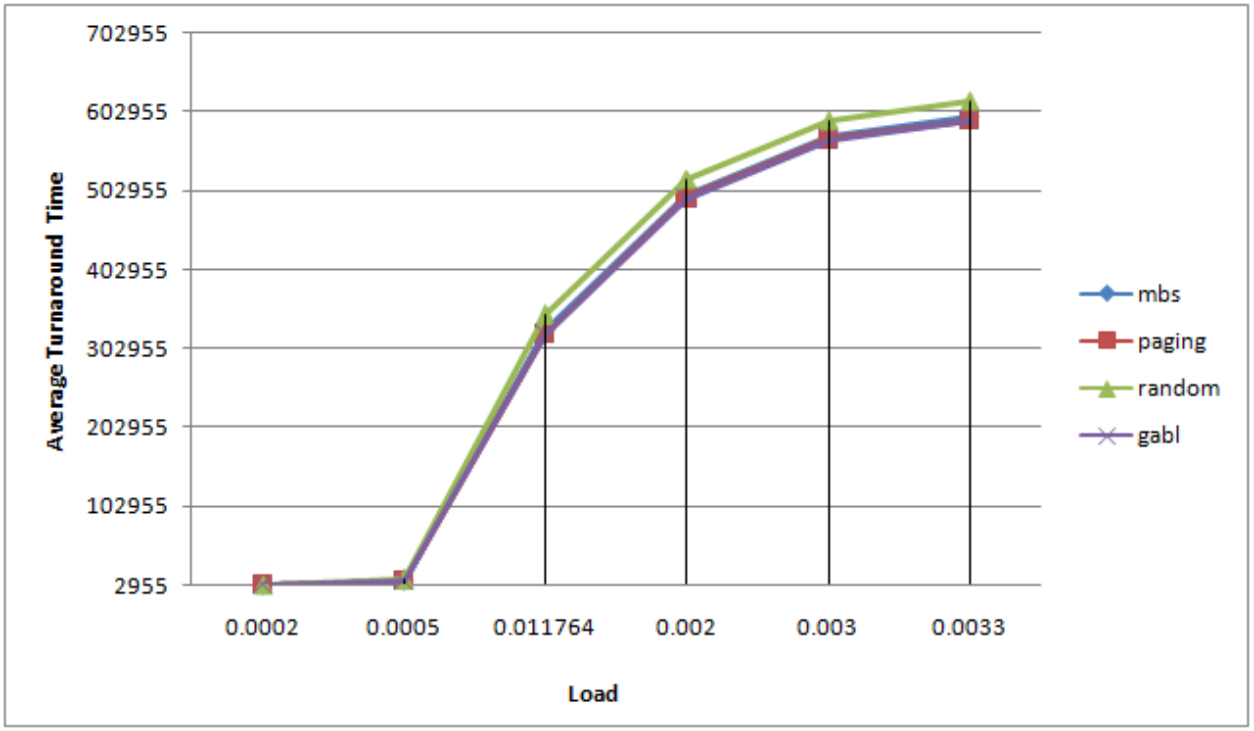

Figure 4.2: Average turnaround time of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the FCFS strategy and bounded pareto job size distribution in a 16 × 16 mesh.

Figures 4.3 and 4.4 display the average turnaround time of jobs that is plotted against the system load for the one-to-all communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under OO scheduling. The results for both job size distributions considered in this research reveal that GABL, MBS and Paging(0) outperform Random allocation. This is because GABL, MBS and Paging(0) maintain a higher degree of contiguity among the allocated processors than that of the Random strategy. In OO scheme, the jobs are allocated based on their arrival sequence instead of waiting at the head of the waiting queue. The results also reveal that the average turnaround time of non-contiguous allocation under OO is lower than that of non-contiguous allocation under FCFS; which leads to improvement in system performance under the OO scheduling strategy over the FCFS scheduling strategy.

As indicated previously in Figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

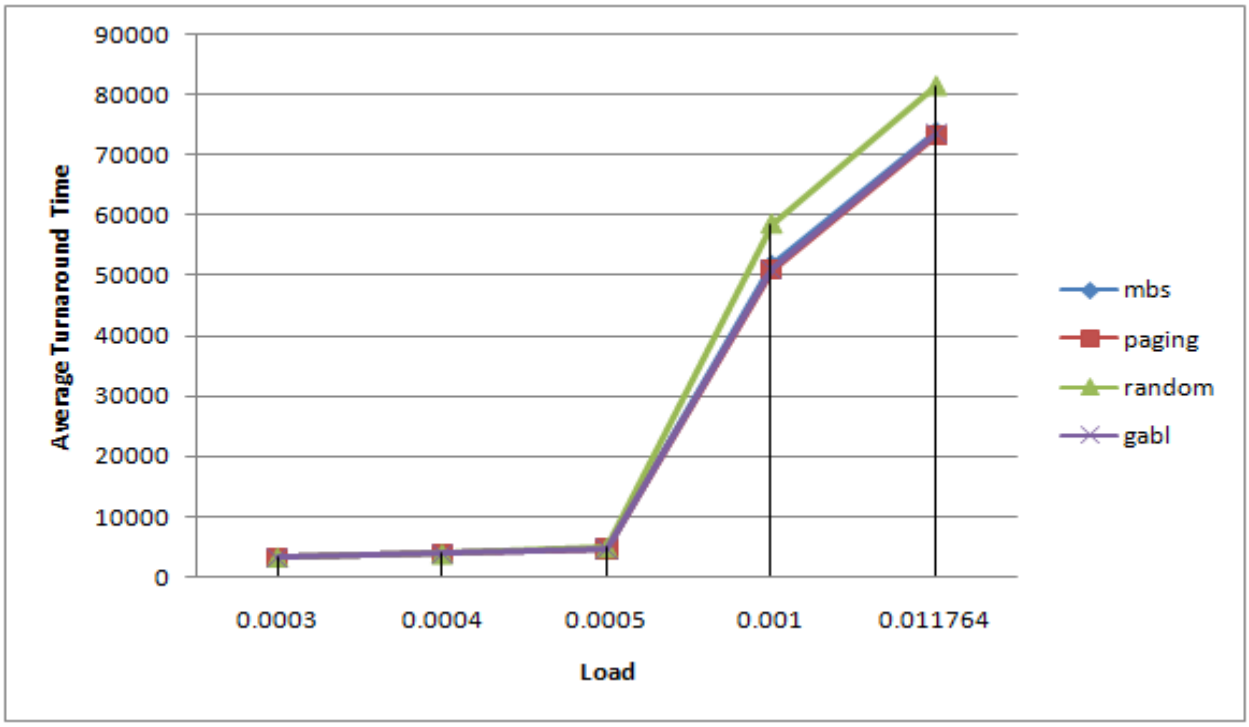

Figure 4.3: Average turnaround time of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the OO strategy and uniform job size distribution in a 16×16 mesh.

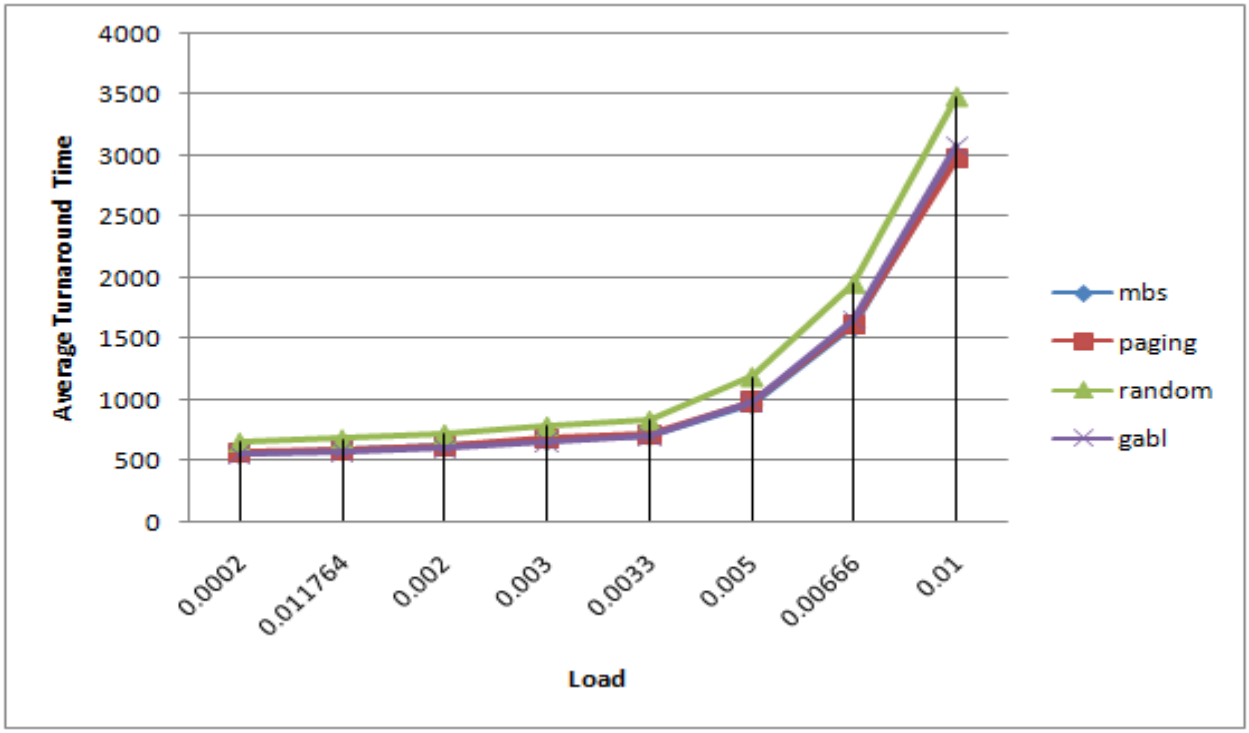

Figure 4.4: Average turnaround time of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the OO strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.5 and 4.6, the average turnaround time of jobs is plotted against the system load for the one-to-all communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under window-based scheduling.

The results for both job size distributions show that MBS, GABL and Paging(0) are better than the Random allocation strategy. This is due to the lack of contiguity among the allocated processors in Random allocation. A window-based scheme uses a window of sequential jobs that starts with the oldest waiting job and this job can be passed by a subsequent job is within the window of k consecutive jobs that starts with the oldest waiting job, where$k$ is 240 as recommended in (Bani Mohammad and Ababneh, 2011), and within this window the jobs will be selected for allocation and execution. The results also reveal that the average turnaround time of

non-contiguous allocation that results from window-based is lower than that of non-contiguous allocation under FCFS; which leads to improvement in system performance under window-based scheduling over the FCFS scheduling. As stated previously in Figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

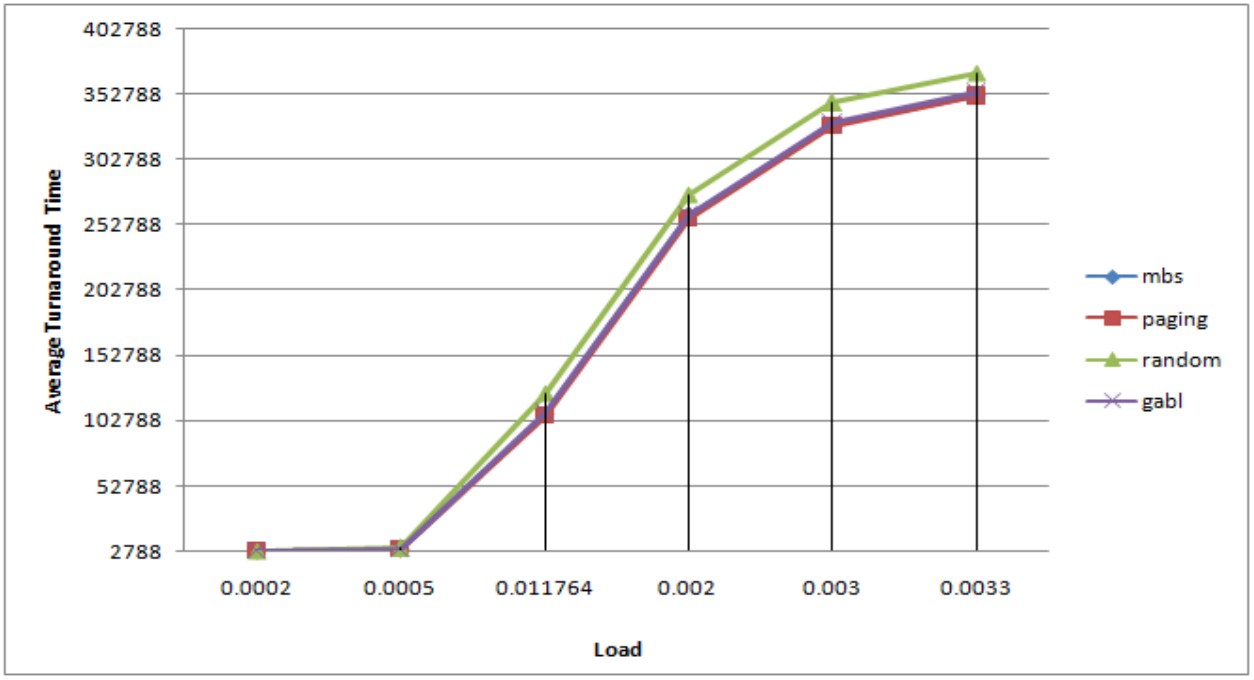

Figure 4.5: Average turnaround time of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the window-based strategy and uniform job size distribution in a 16 × 16 mesh.

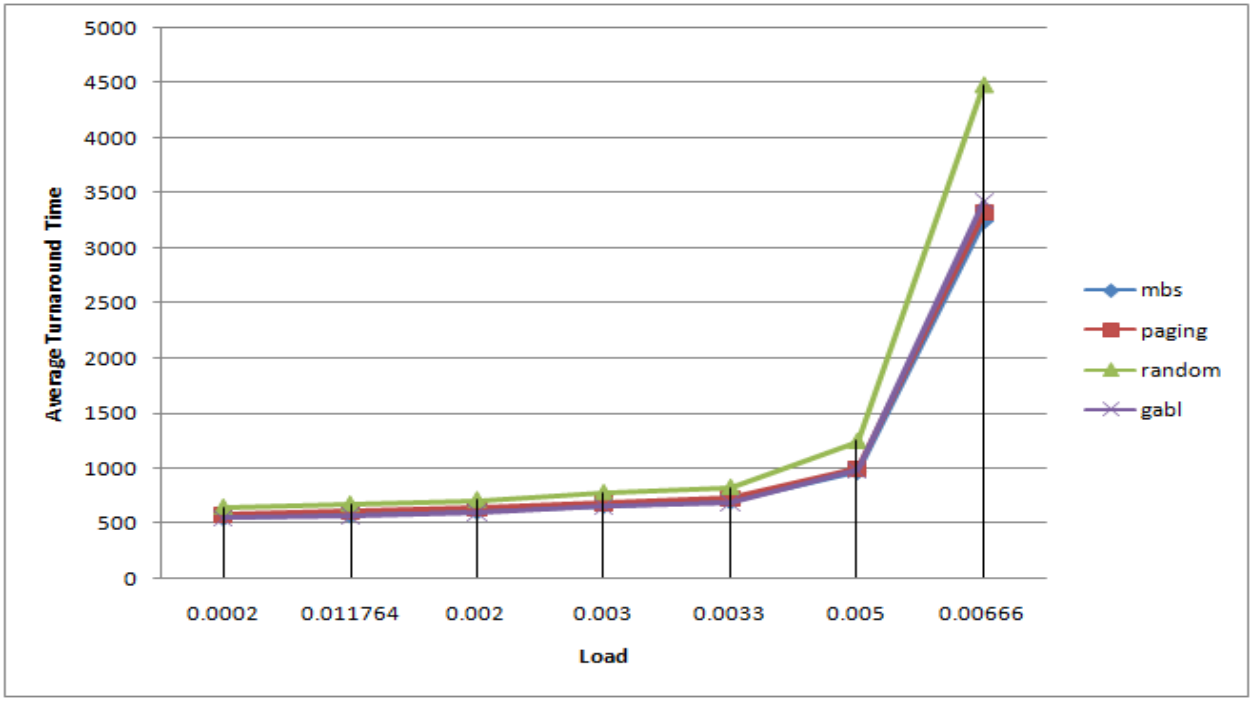

Figure 4.6: Average turnaround time of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the window-based strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.7 and 4.8, the average turnaround time of jobs is plotted against the system load for the random communication pattern and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under FCFS scheduling. It can be noticed from these figures that Random strategy has the worst performance for both job size distributions considered in this research. Again, this is because the lack of contiguity among the allocated processors in the Random allocation strategy. Although, GABL, MBS and Paging(0) have worthy performance. This is because these strategies provide a higher degree of contiguity between the allocated processors than that of the Random strategy. As shown early in Figures 4.1and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. In the random communication pattern, message contention is smaller than that for the one-to-all communication pattern. This is because destinations are chosen randomly and paths are less likely to overlap.

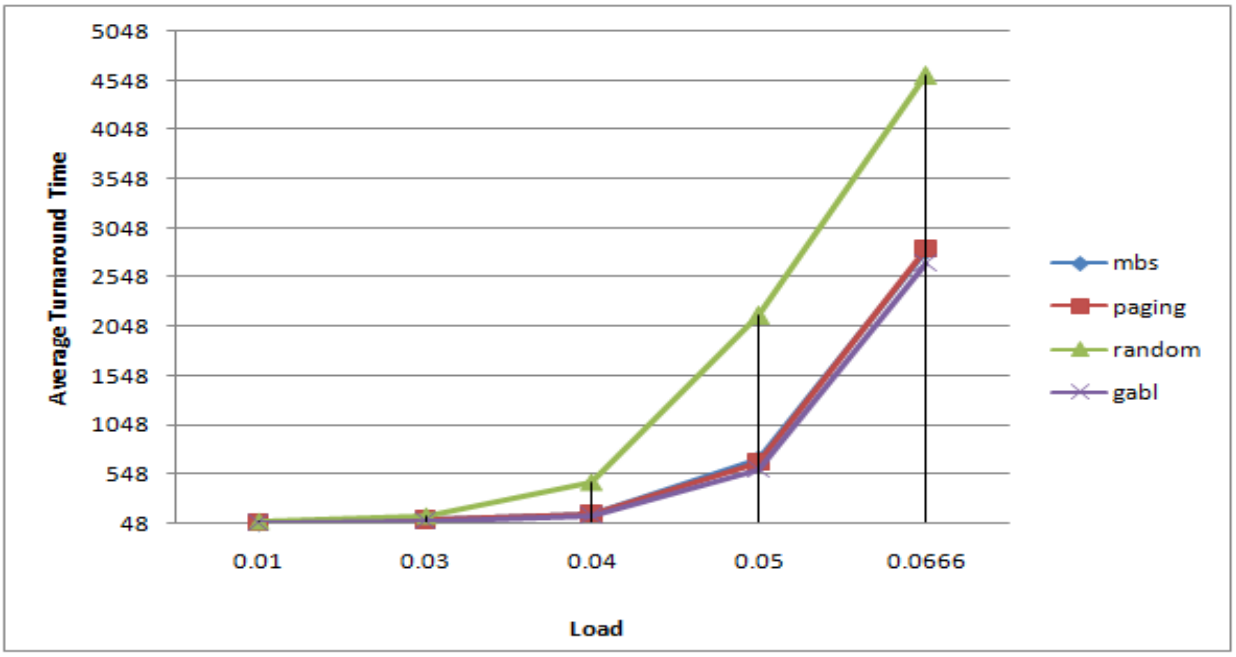

Figure 4.7: Average turnaround time of the non-contiguous allocation strategies vs. system load for the random communication pattern under the FCFS strategy and uniform job size distribution in a 16 × 16 mesh.

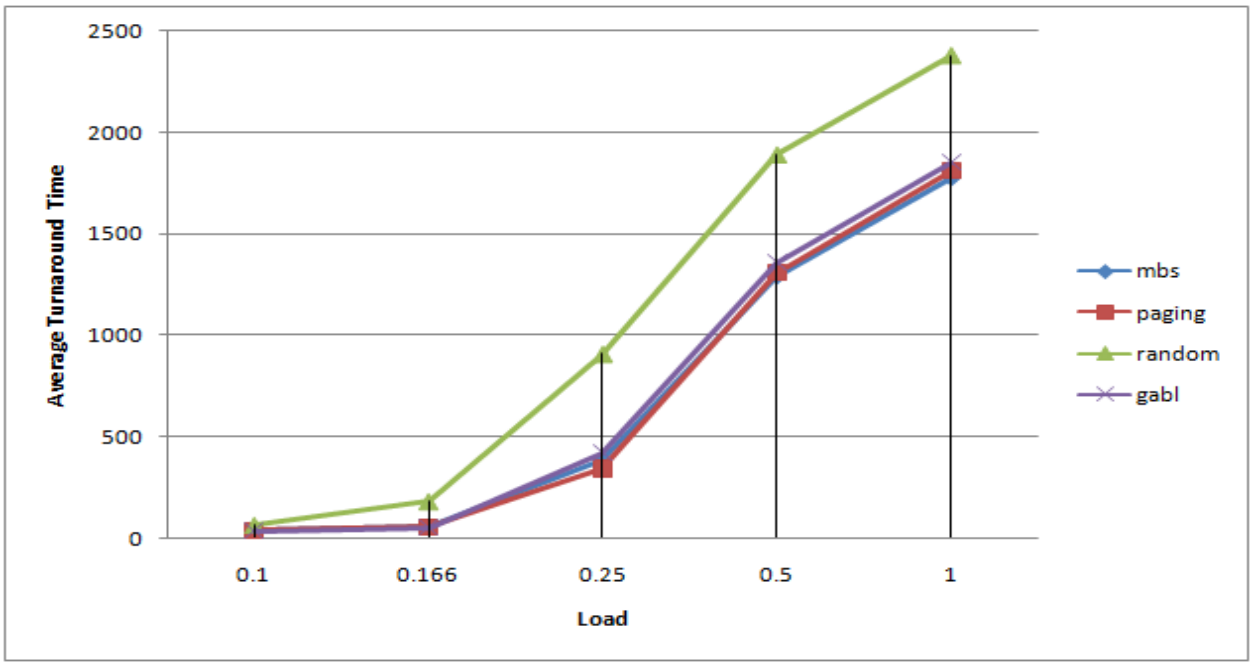

Figure 4.8: Average turnaround time of the non-contiguous allocation strategies vs. system load for the random communication pattern under the FCFS strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.9 and 4.10, the average turnaround time of jobs is plotted against the system load for the random communication and two job size distributions(i.e., uniform distribution and bounded pareto distribution) under OO scheduling. The results for both job size distributions considered in this research reveal that GABL, MBS and Paging(0) surpass Random allocation. This is because GABL, MBS and Paging(0) have a higher degree of contiguity among the allocated processors than that of the Random strategy. It can also be noticed in the figures that OO scheduling is much better than FCFS scheduling. As reported previously in Figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

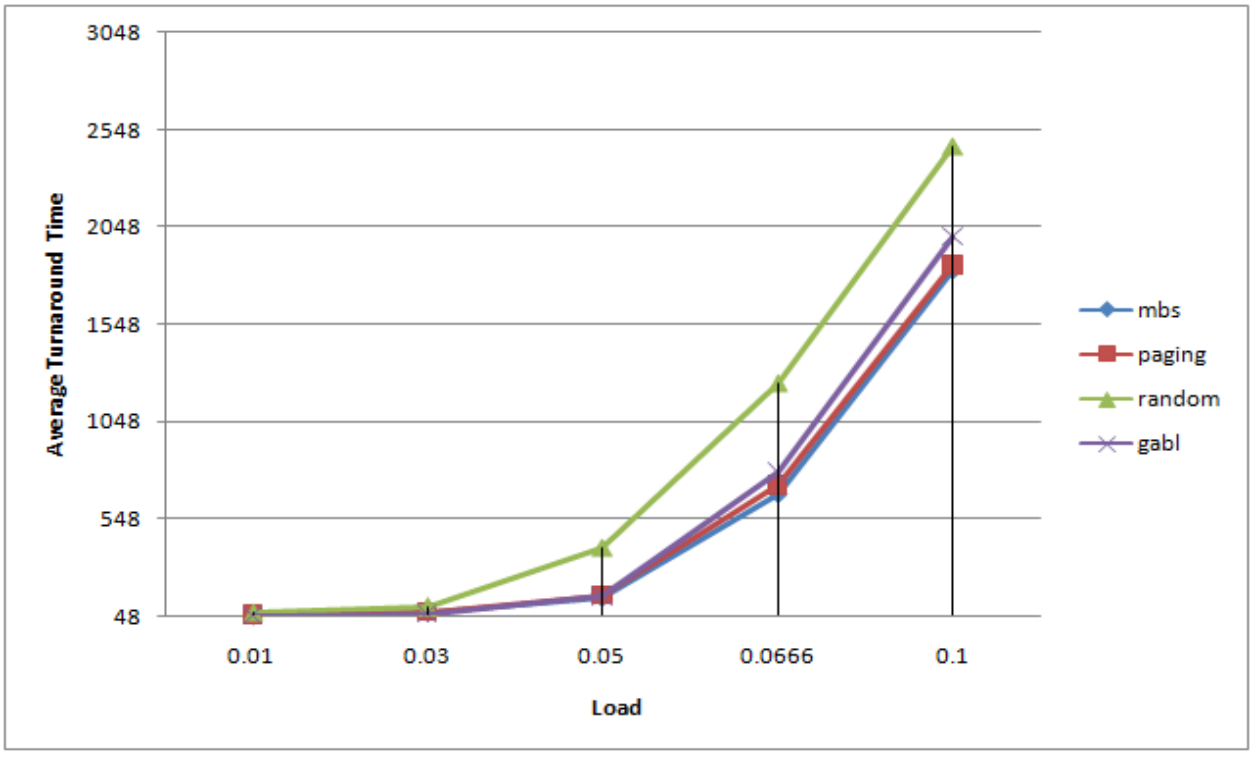

Figure 4.9: Average turnaround time of the non-contiguous allocation strategies vs. system load for the random communication pattern under the OO strategy and uniform job size distribution in a 16 × 16 mesh.

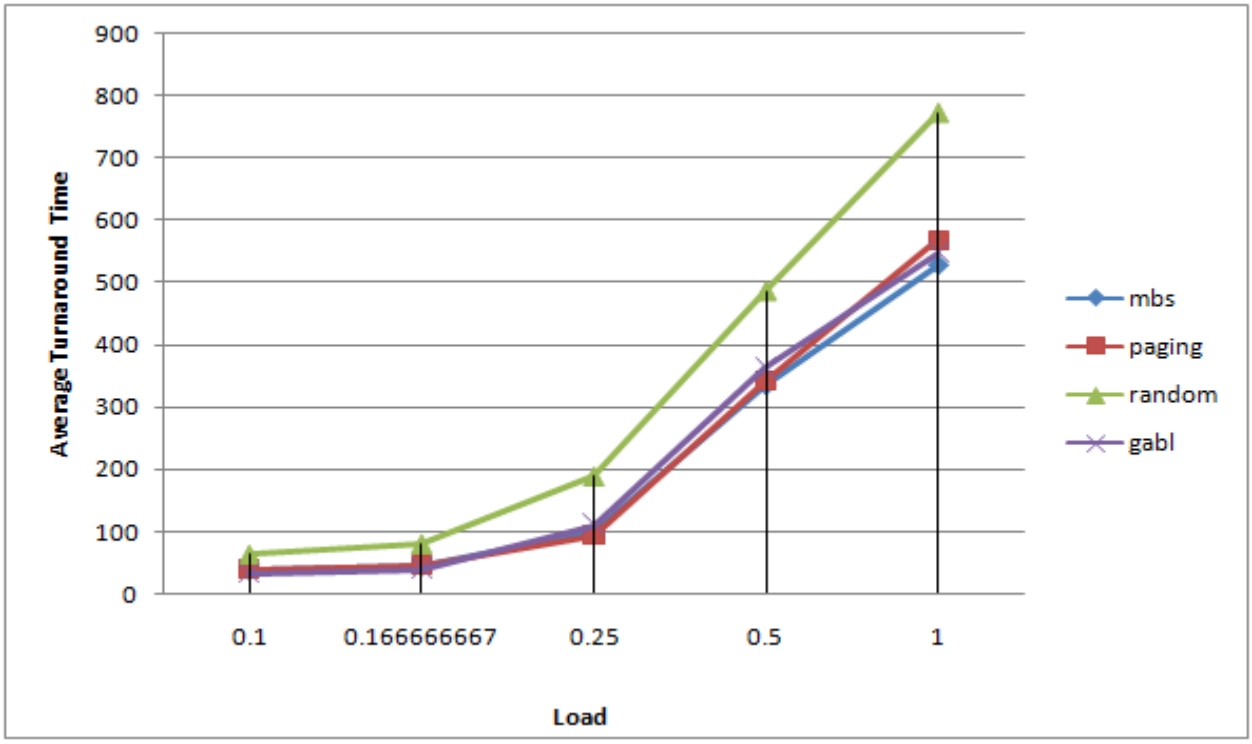

Figure 4.10: Average turnaround time of the non-contiguous allocation strategies vs. system load for the random communication pattern under the OO strategy and bounded pareto job size distribution in a 16 × 16mesh.

In Figures 4.11 and 4.12, the average turnaround time of jobs is plotted against the system load for the random communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under window-based scheduling. The results for both job size distributions considered in this research show that MBS, GABL and Paging(0) have a good performance than that of the Random allocation strategy. This is due to the absence of contiguity between the allocated processors in Random allocation. It can be observed from the figures that window-based scheduling is much better than FCFS scheduling. As mentioned previously in figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

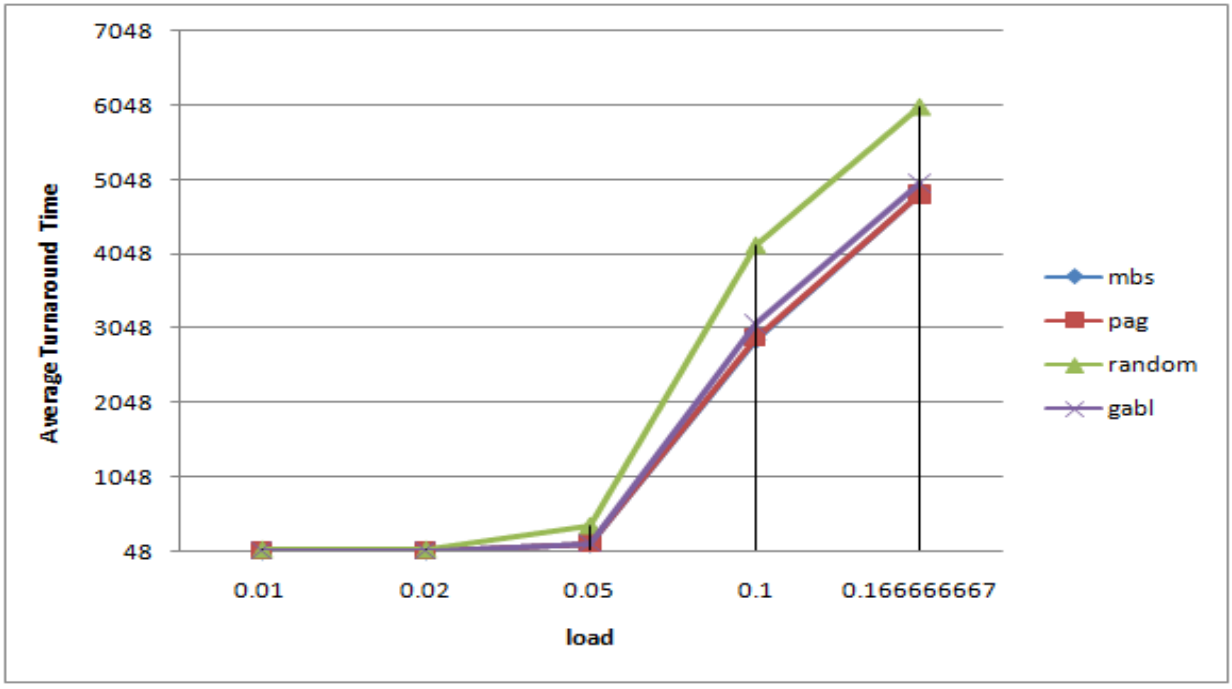

Figure 4.11: Average turnaround time of the non-contiguous allocation strategies vs. system load for the random communication pattern under the window-based strategy and uniform job size distribution in a 16 × 16mesh.

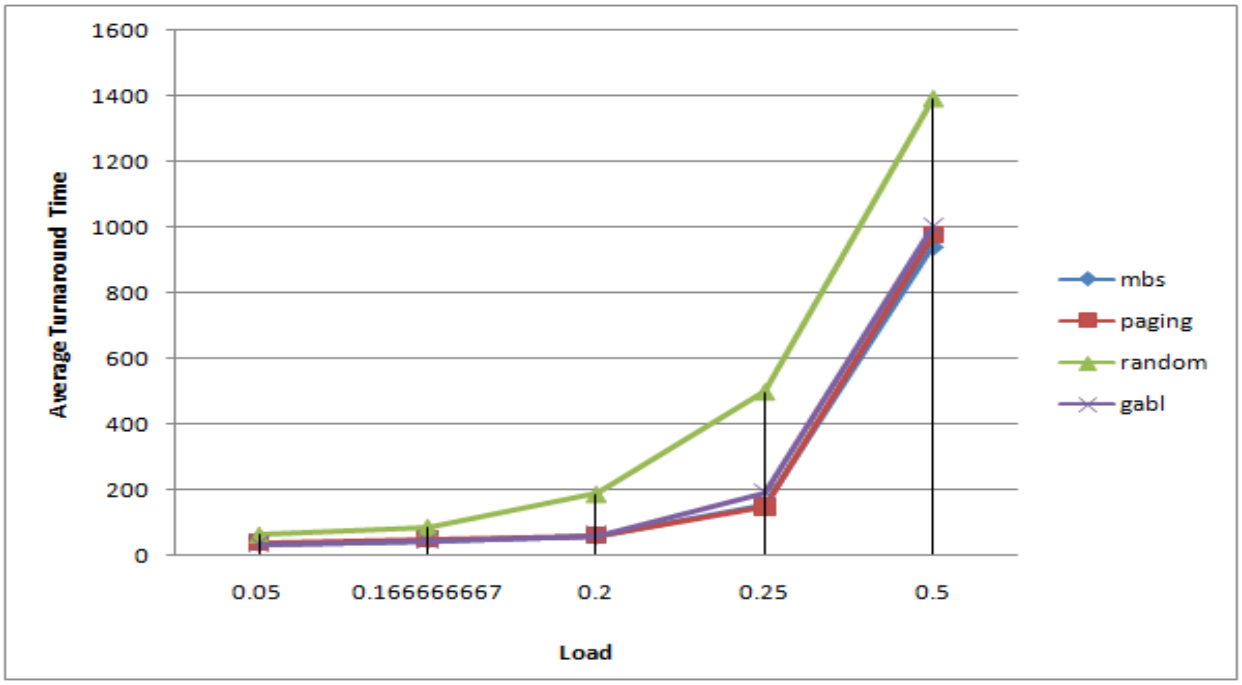

Figure 4.12: Average turnaround time of the non-contiguous allocation strategies vs. system load for the random communication pattern under the window-based strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.13 and 4.14, the average turnaround time of jobs is plotted against the system load for the near neighbour communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under FCFS scheduling. The results for both job size distributions considered in this research demonstrate that GABL, MBS and Paging(0) have a better performance than that of Random allocation. This is because GABL, MBS and Paging(0) have a higher degree of contiguity among the allocated processors than that of the Random strategy. As stated early in Figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

Near neighbour communication has better performance than that of the one-to-all communication for both job size distributions under all scheduling strategies. This is due to the sender bottleneck problem, that one-to-all communication pattern suffers from; in which the sending processor is blocked because a connection link it needs is busy with the previous send operation; that is sending is actually serialized, not parallel.

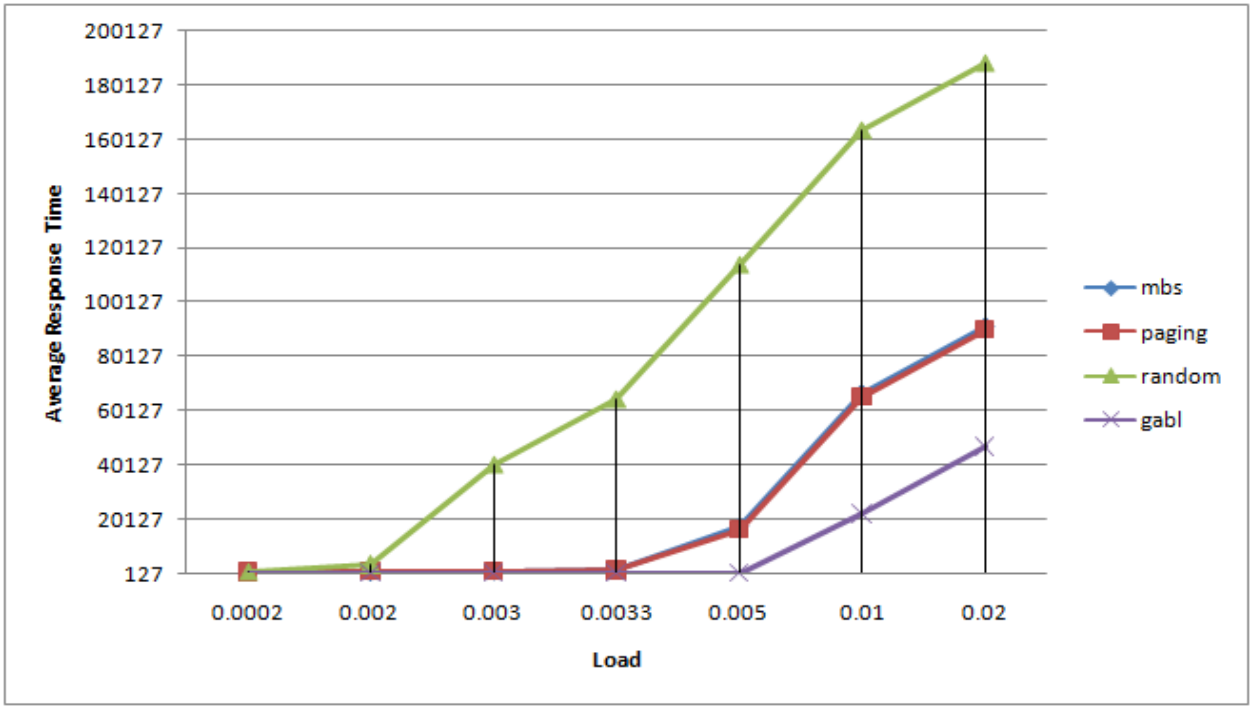

Figure 4.13: Average turnaround time of the non-contiguous allocation strategies vs. system load for the near neighbour communication pattern under the FCFS strategy and uniform job size distribution in a 16 × 16mesh.

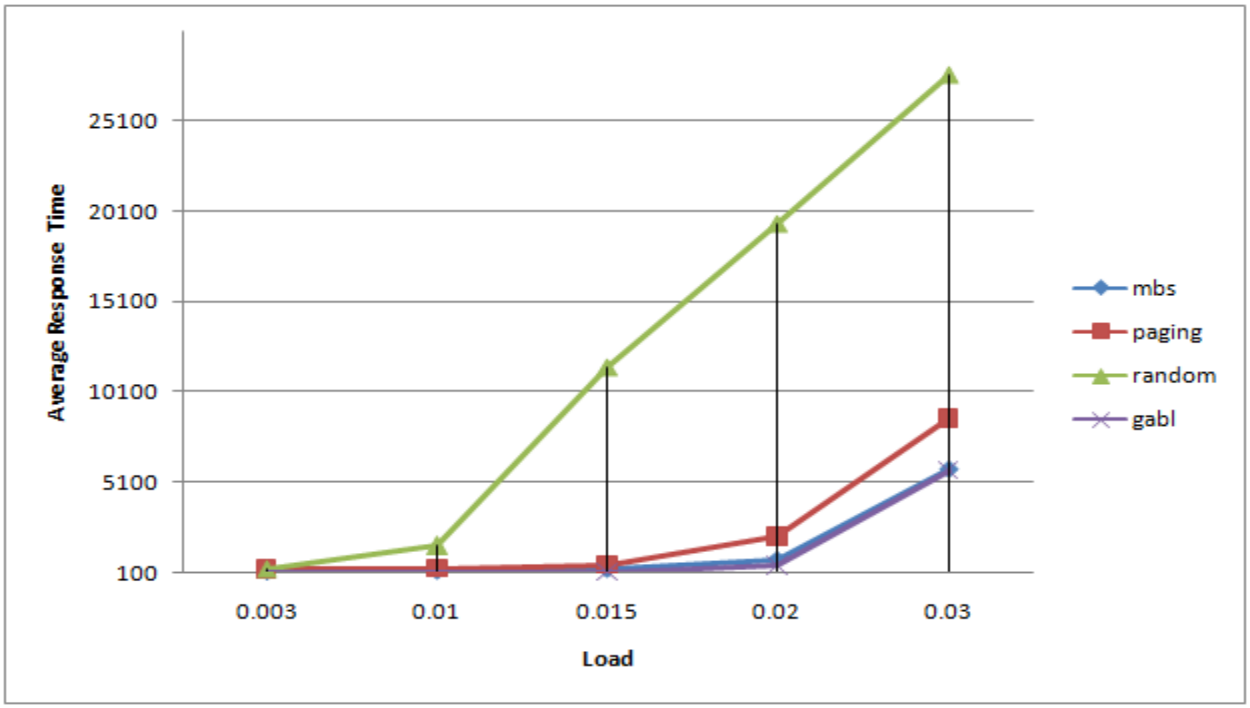

Figure 4.14: Average turnaround time of the non-contiguous allocation strategies vs. system load for the near neighbour communication pattern under the FCFS strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.15 and 4.16, the average turnaround time of jobs is plotted against the system load for the near neighbour communication and the two job size distributions (i.e., uniform distribution and bounded pareto distribution) under OO scheduling.

It can be observed from these figures that Random strategy has the worst performance for both job size distributions considered in this research. Again, this is because the lack of contiguity between the allocated processors in the random allocation strategy. The results show that OO scheduling is superior to FCFS scheduling. As indicated previously in Figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

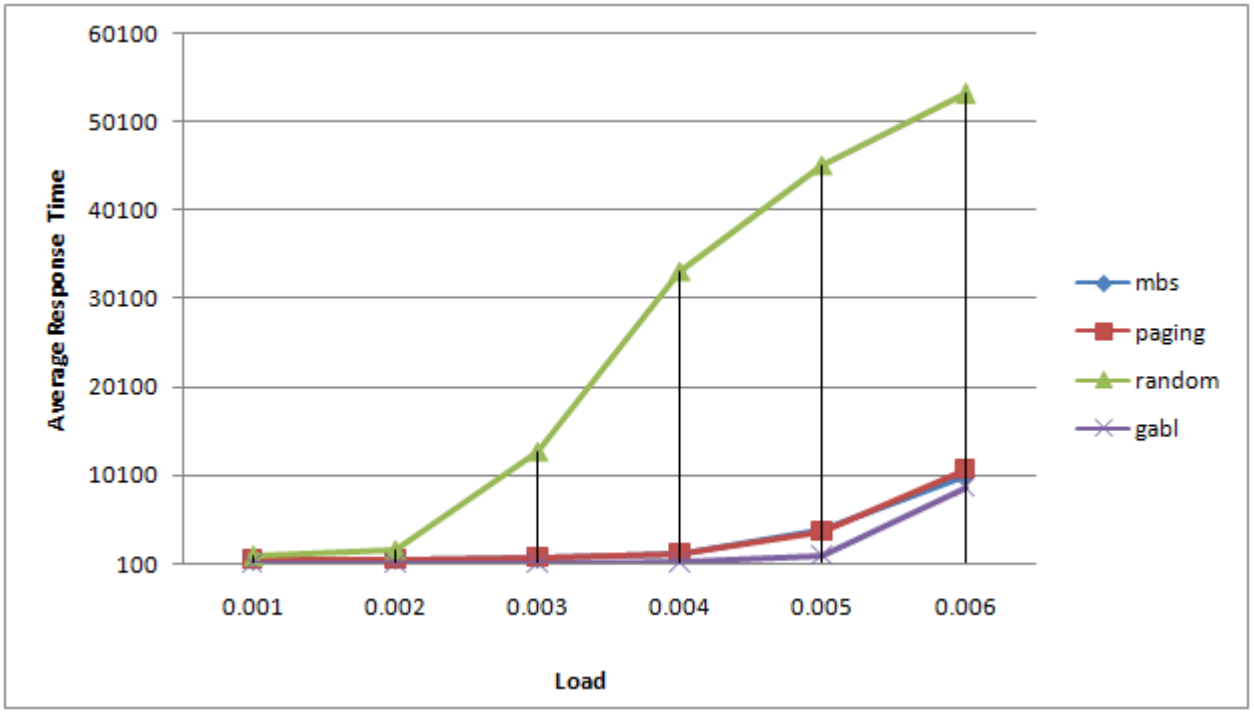

Figure 4.15: Average turnaround time of the non-contiguous allocation strategies vs. system load for near neighbor communication pattern under the OO strategy and uniform job size distribution in a 16 × 16mesh.

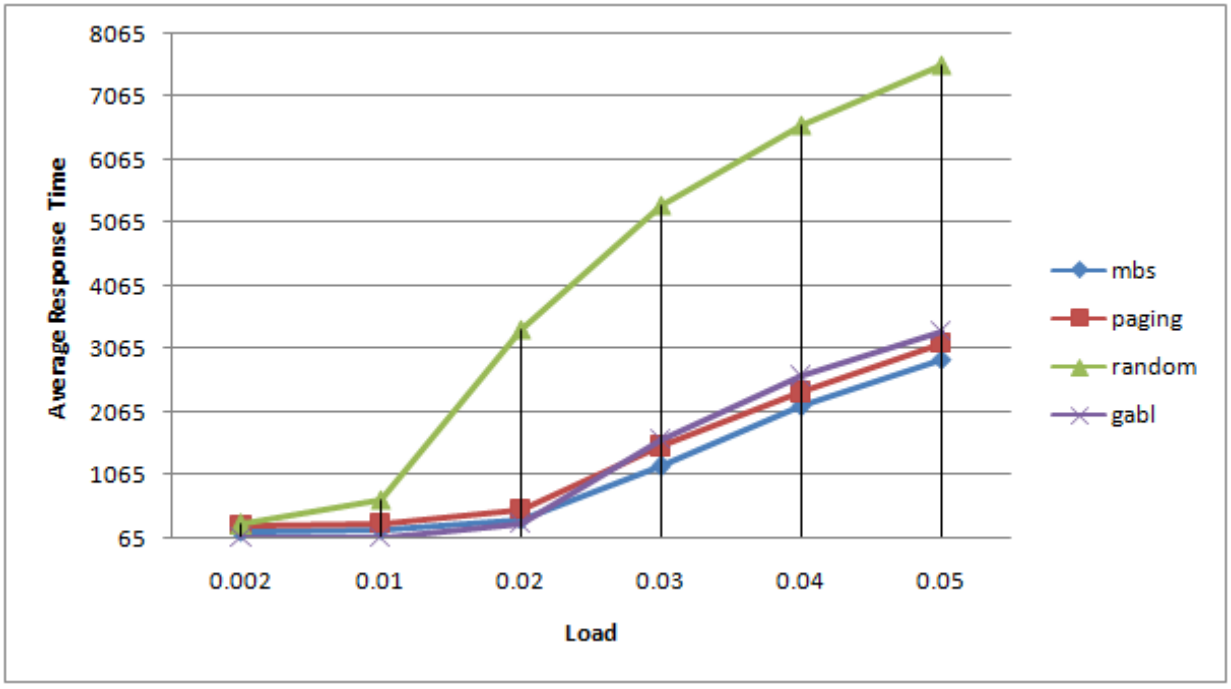

Figure 4.16: Average turnaround time of the non-contiguous allocation strategies vs. system load for the near neighbor communication pattern under the OO strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.17 and 4.18, the average turnaround time of jobs is plotted against the system load for the near neighbour communication and the two job size distributions(i.e., uniform distribution and bounded pareto distribution) under window-based scheduling.

The results for both job size distributions considered in this research show that MBS, GABL and Paging(0) are better than the random allocation strategy. This is due to the absence of contiguity between the allocated processors in Random allocation. The results show that window-based scheduling is much better than FCFS scheduling.

As mentioned previously in Figures 4.1 and 4.2, the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

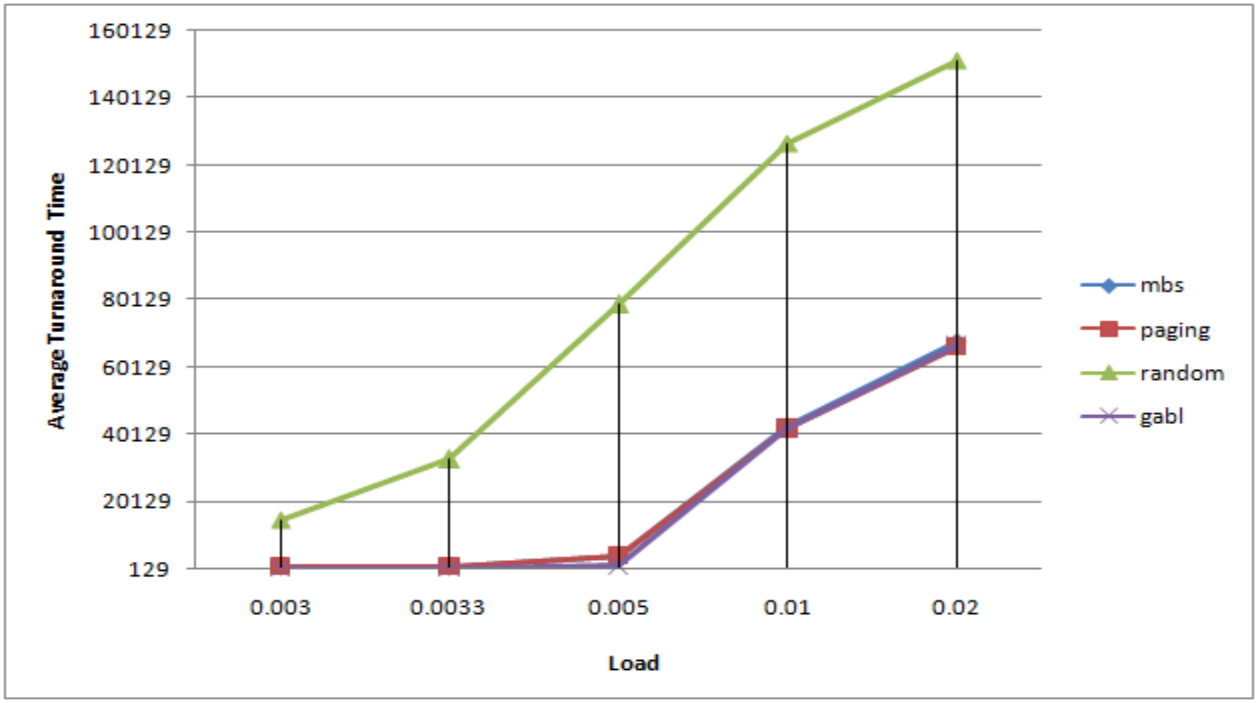

Figure 4.17: Average turnaround time of the non-contiguous allocation strategies vs. system load for the near neighbour communication pattern under the window-based strategy and uniform job size distribution in a 16 × 16 mesh.

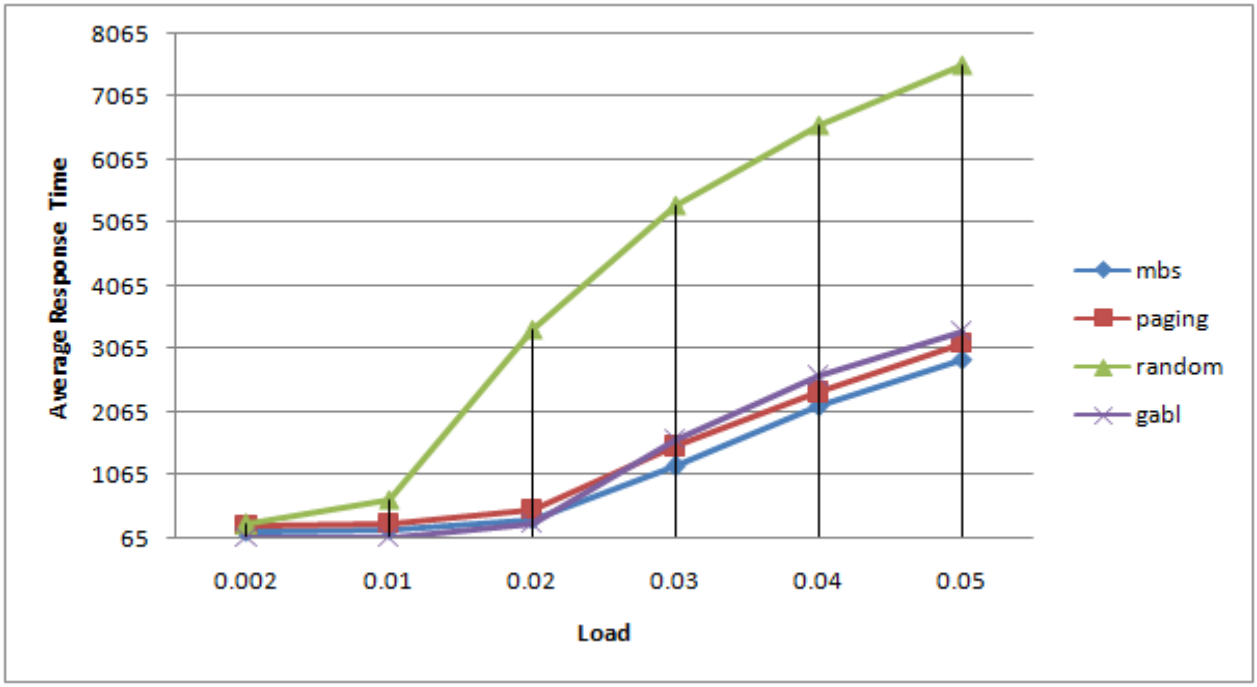

Figure 4.18: Average turnaround time of the non-contiguous allocation strategies vs. system load for the near neighbour communication pattern under the window-based strategy and bounded pareto job size distribution in a 16 × 16 mesh.

4.3 Average System Utilization Results

In Figures 4.19 and 4.20, the average system utilization of the allocation strategies is plotted against the system load for the one-to-all communication pattern and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under FCFS scheduling. We notice that all non-contiguous allocation strategies considered in this research have a comparable performance. However, we notice that the Random allocation has the highest utilization for most system loads considered. This is because the lack of contiguity in the random allocation, which increases the contention and as a consequence increases the average turnaround time. Also, the results show that the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution; the average system utilization of non-contiguous allocation under FCFS in bounded pareto is increased and consequently lead to an improvement in system performance.

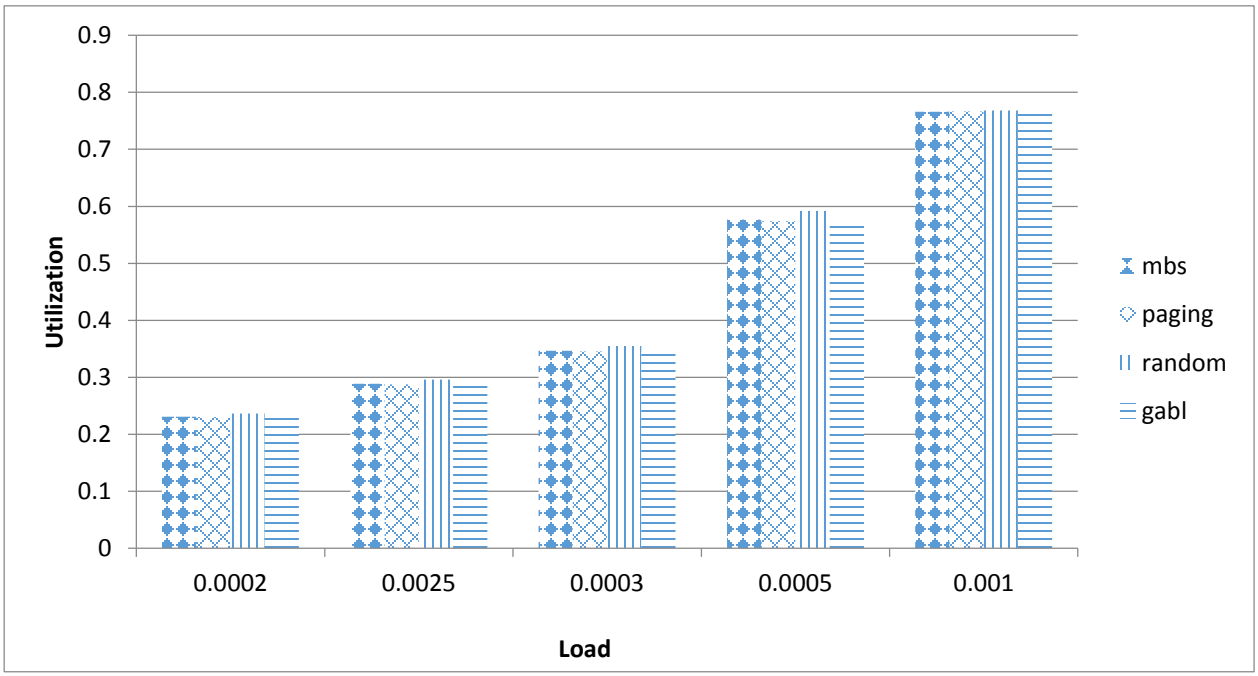

Figure 4.19: Average system utilization of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the FCFS strategy and uniform job size distribution in a 16 × 16mesh.

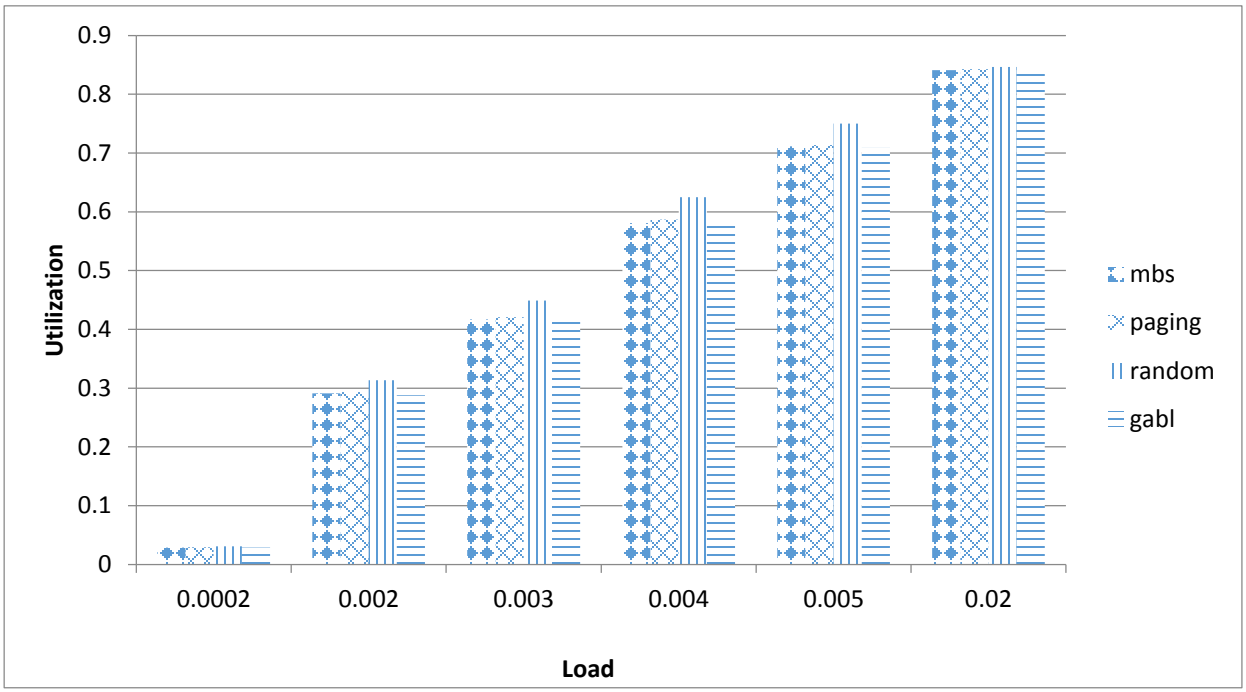

Figure 4.20: Average system utilization of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the FCFS strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.21 and 4.22, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the one-to-all communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under OO scheduling. The results for both job size distributions considered in this research reveal that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

We notice that the Random allocation under both job size distributions has the highest utilization ratio for most system loads considered. This is due the lack of contiguity in the Random allocation; which increases the contention and hence increases the average turnaround time. The results also reveal that the average system utilization of non-contiguous allocation under OO is

higher than that of non-contiguous allocation under FCFS. Which leads to improvement in system performance under OO scheduling strategies over the FCFS scheduling.

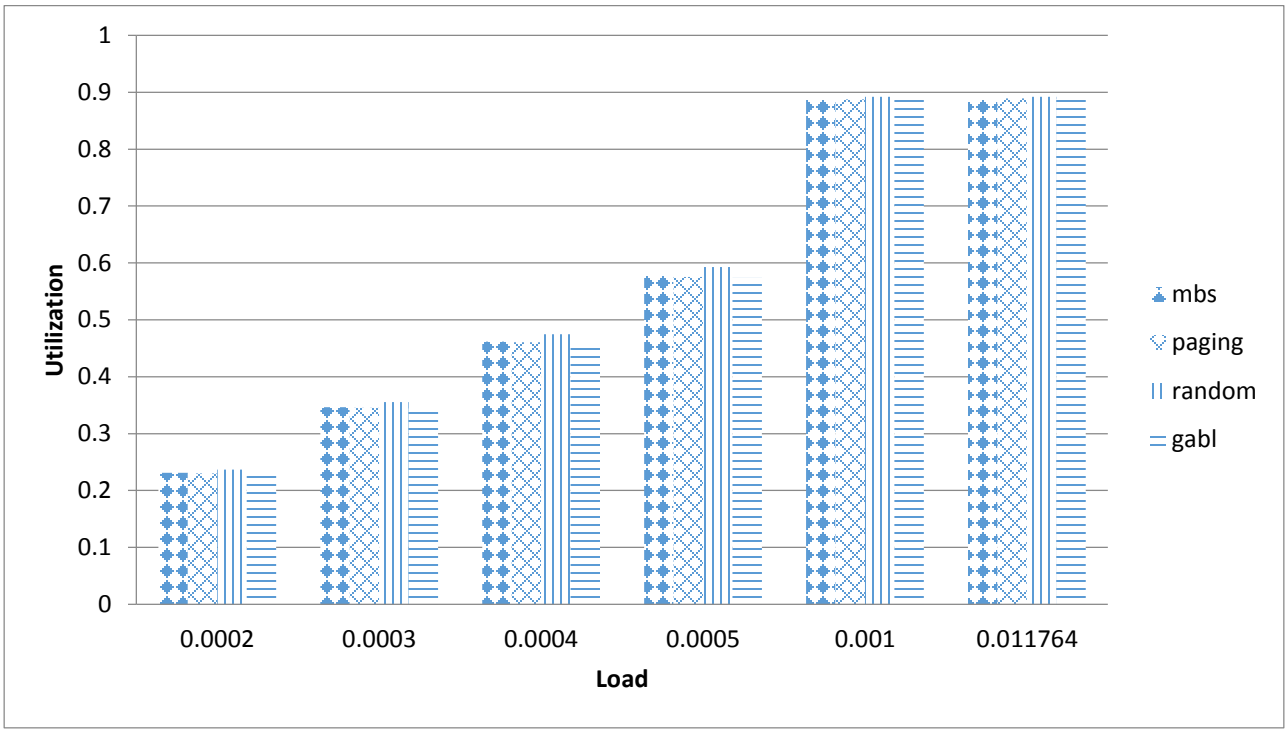

Figure 4.21: Average system utilization of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the OO strategy and uniform job size distribution in a 16 × 16mesh.

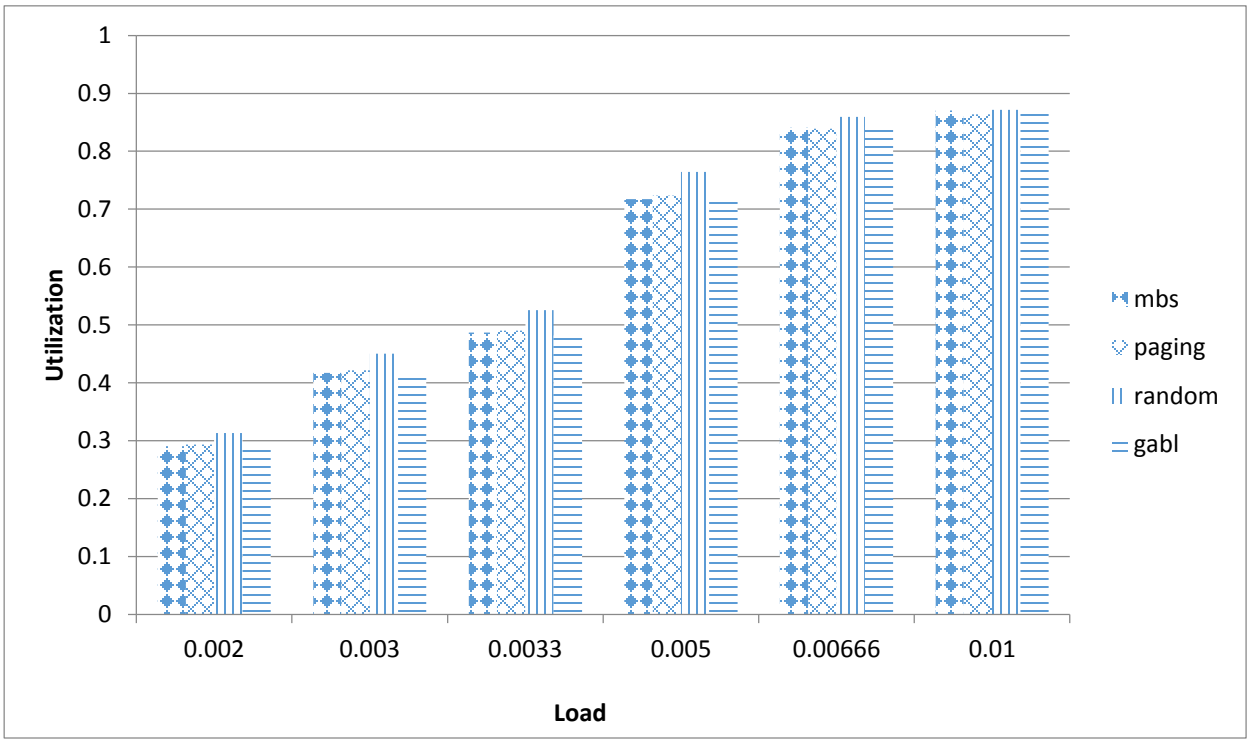

Figure 4.22: Average system utilization of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the OO strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.23 and 4.24, the average system utilization of the allocation strategies is plotted against the system load for the one-to-all communication pattern and the two job size distributions (i.e., uniform distribution and bounded pareto distribution) under window-based scheduling. We notice that the Random allocation under both job size distributions has the highest utilization ratio for most system loads considered. This is because of the lack of contiguity in the random allocation, which increases the contention and as a consequence increases the average turnaround time.

The results also show that the performance of the allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. The results show that the non-contiguous allocation strategies that use window-based scheduling has the highest average system utilization

among the non-contiguous allocation strategies considered. Which means that the window-based scheduling strategy is superior to FCFS and OO scheduling strategies.

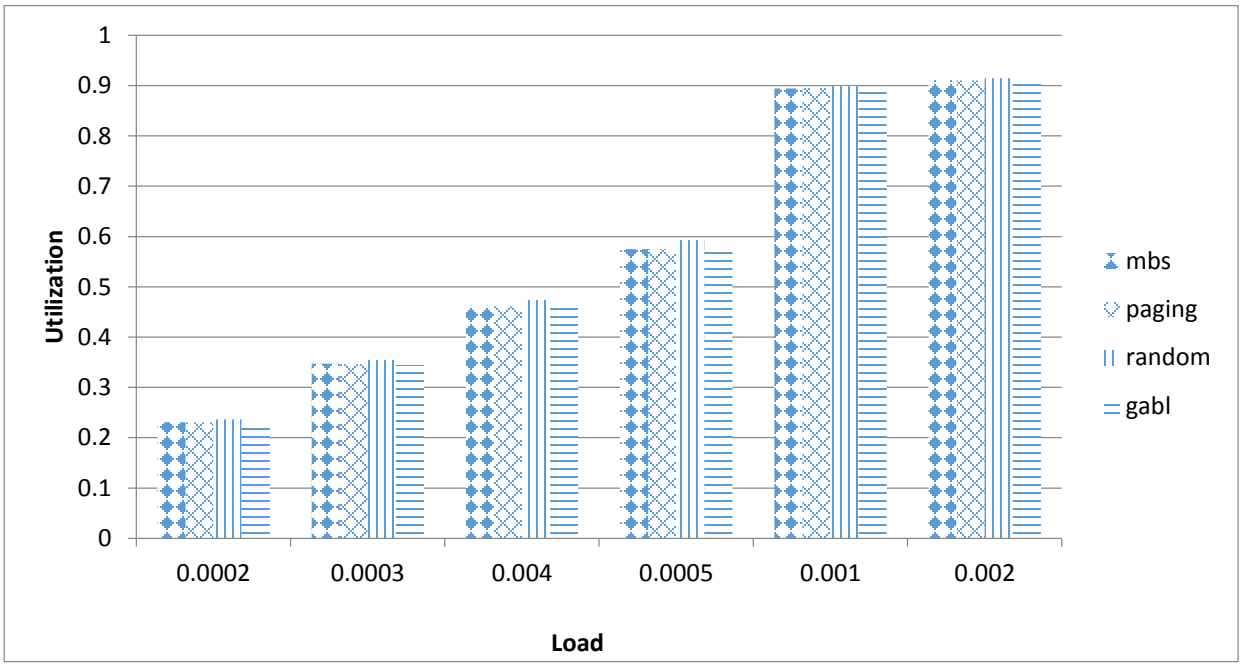

Figure 4.23: Average system utilization of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the window-based strategy and uniform job size distribution in a 16 × 16 mesh.

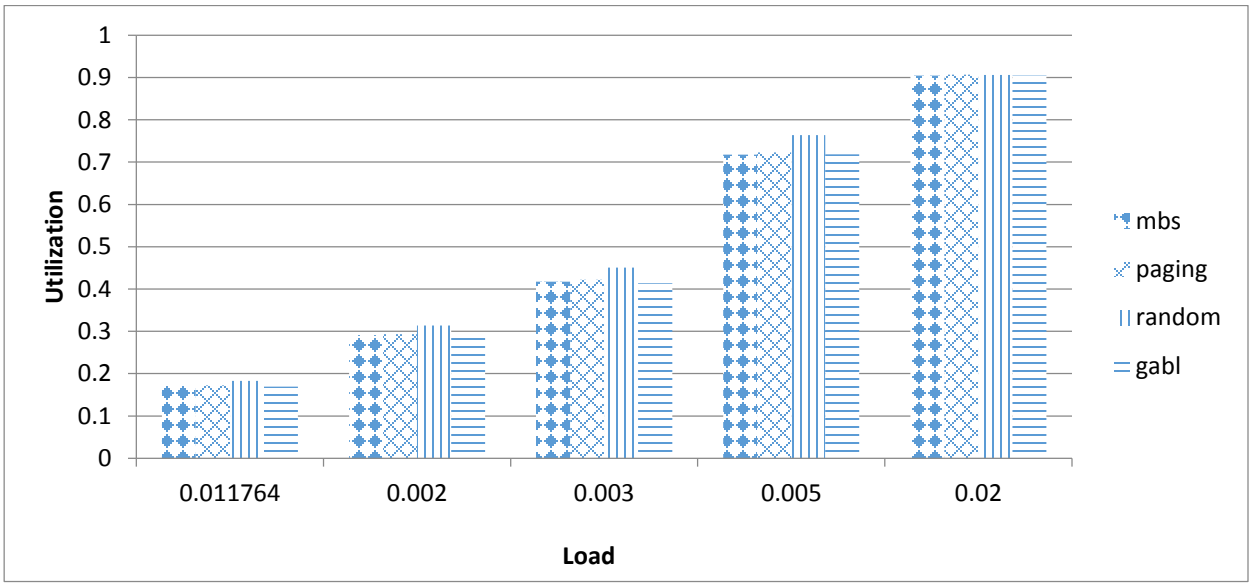

Figure 4.24: Average system utilization of the non-contiguous allocation strategies vs. system load for the one-to-all communication pattern under the window-based strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.25 and 4.26, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the random communication and the two job size distributions (i.e., uniform distribution and bounded pareto distribution) under FCFS scheduling. The results for both job size distributions considered in this research reveal that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. The simulation results for both job size distributions also show that the Random allocation strategy has the highest utilization ratio for most system loads considered. This is due to the absence of contiguity among the allocated processors in this policy.

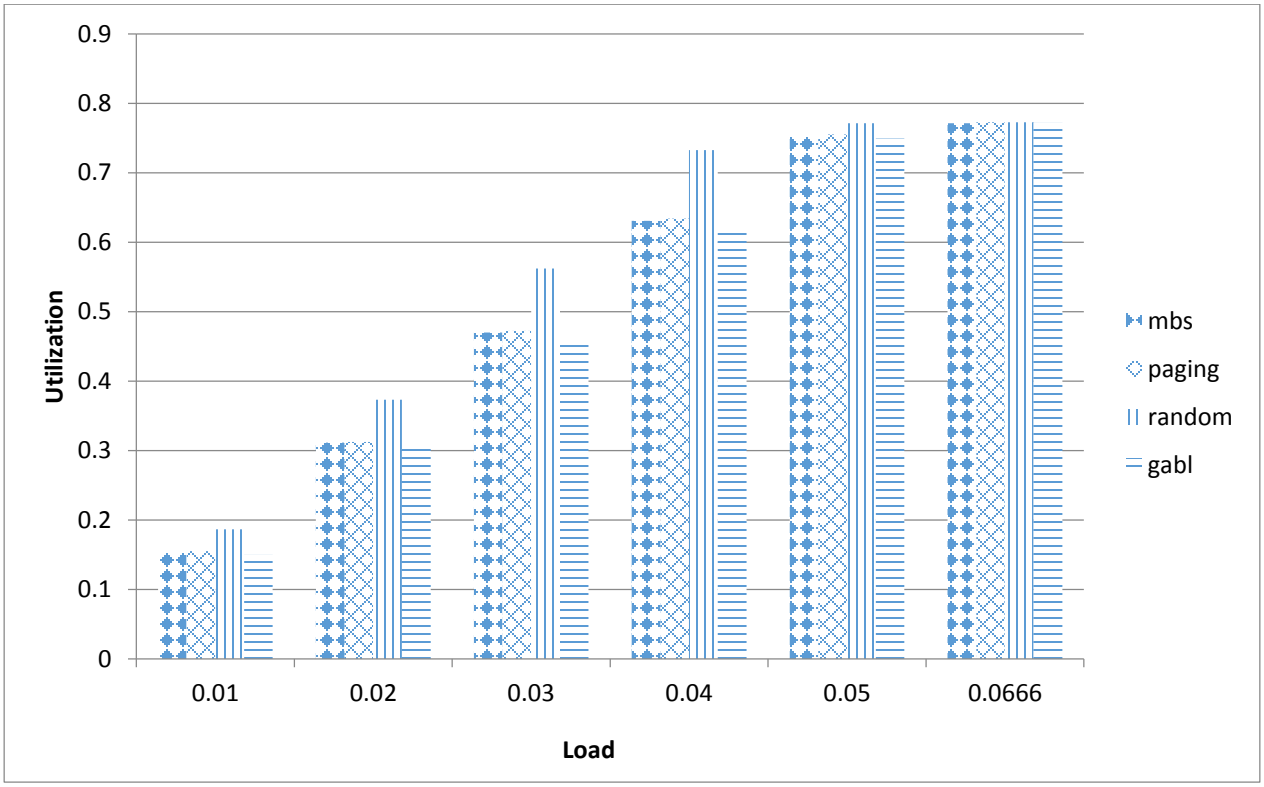

Figure 4.25: Average system utilization of the non-contiguous allocation strategies vs. system load for the random communication pattern under FCFS strategy and uniform job size distribution in a 16 × 16 mesh.

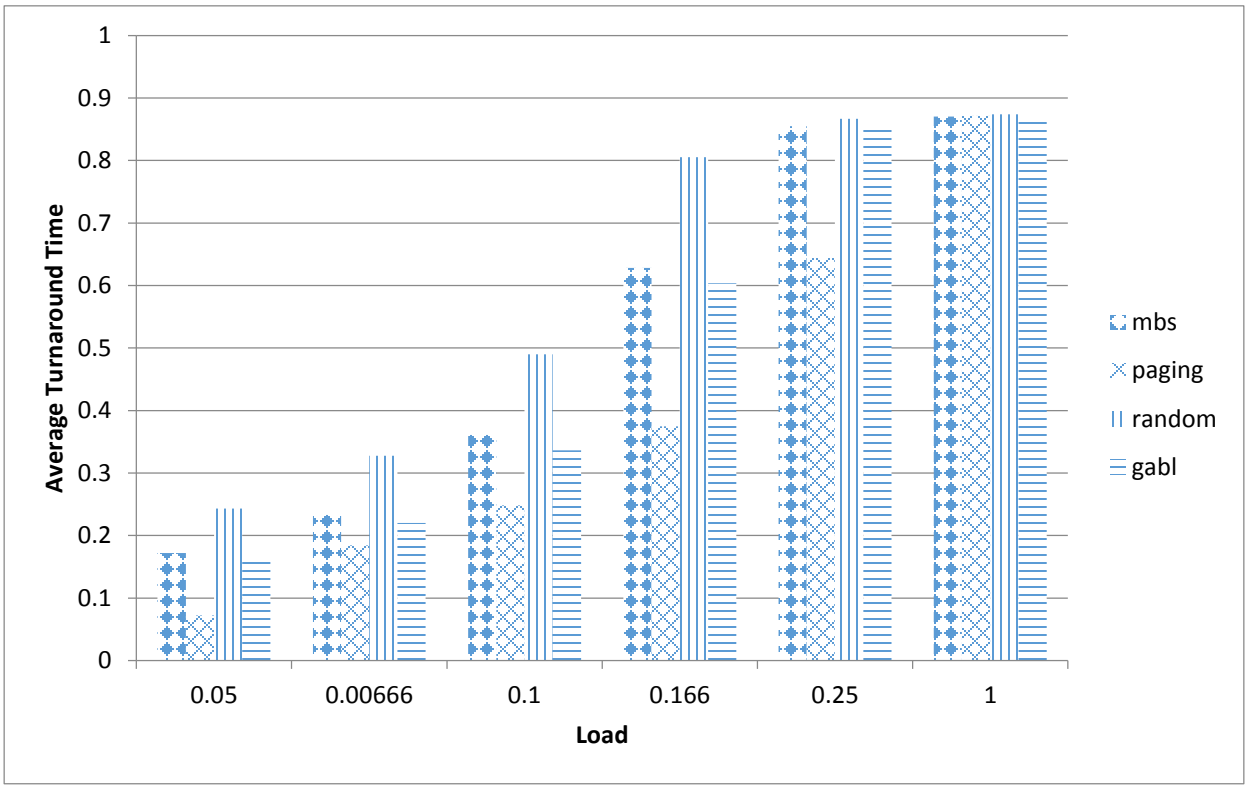

Figure 4.26: Average system utilization of the non-contiguous allocation strategies vs. system load for the random communication pattern under the FCFS strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.27 and 4.28, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the random communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under OO scheduling. As mentioned previously the results for both job size distributions considered in this research show that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto distribution.

We perceive that the Random allocation under both job size distributions outperforms all the other non-contiguous allocation strategies. This is due to the lack of contiguity in the random allocation strategy. The results also reveal that the average system utilization of non-contiguous allocation for both job size distributions under OO is higher than that of non-contiguous

allocation under FCFS. This leads to improvement in system performance under OO scheduling over the FCFS scheduling.

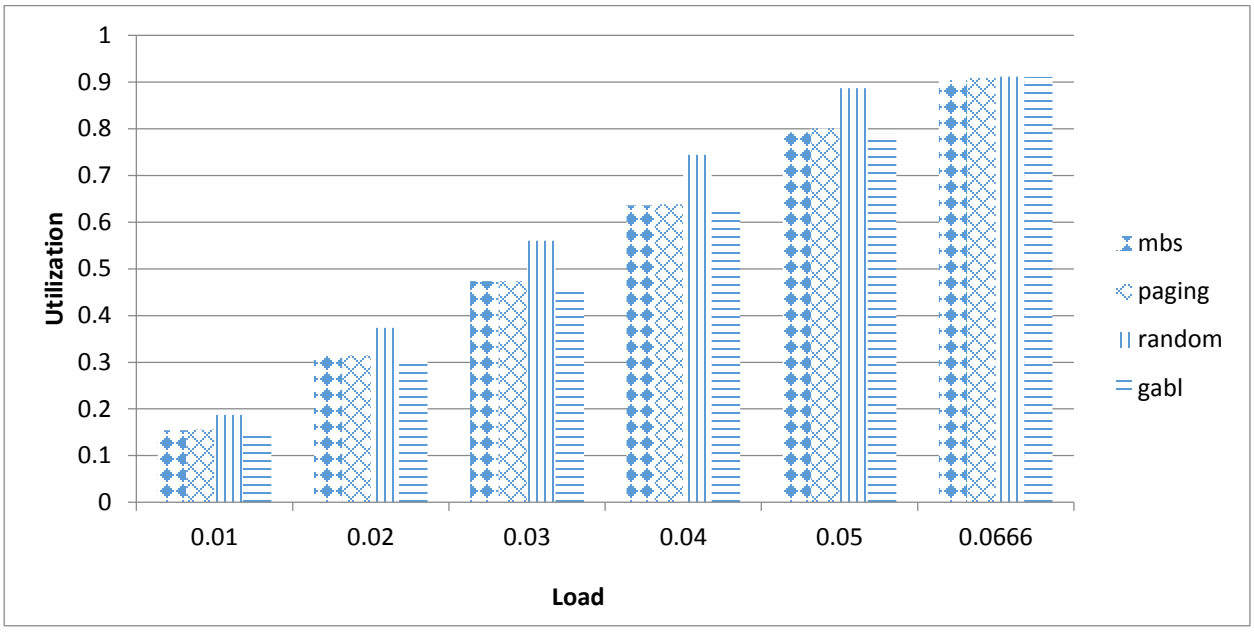

Figure 4.27: Average system utilization of the non-contiguous allocation strategies vs. system load for the random communication pattern under OO strategy and uniform job size distribution in a 16 × 16 mesh.

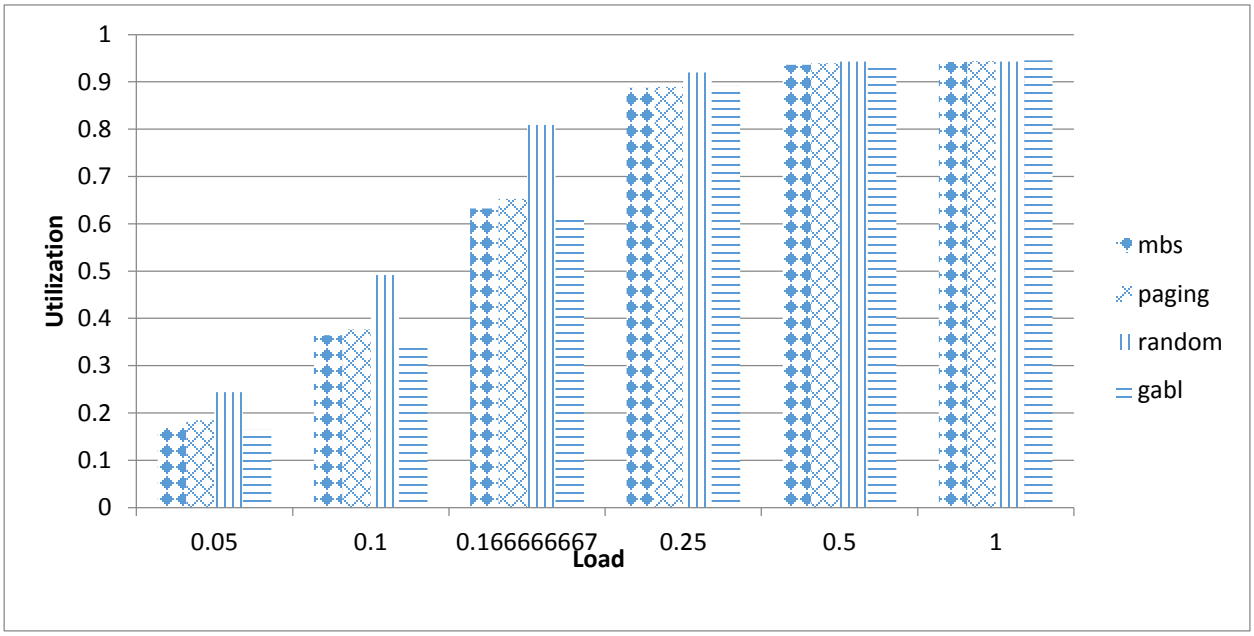

Figure 4.28: Average system utilization of the non-contiguous allocation strategies vs. system load for the random communication pattern under the OO strategy and bounded pareto job size distribution in a 16 × 16mesh.

In Figures 4.29 and 4.30, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the random communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under window-based scheduling. The results for both job size distributions considered in this research reveal that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. We notice that the Random allocation under both job size distributions has the highest utilization ratio for most system loads considered. This is due to the absence of contiguity between the allocated processors in this policy, which increases the job service times and gives good system utilization. The results also show that the average system utilization of non-contiguous allocation for the two job size distributions considered under window-based is higher than that of non-contiguous allocation under FCFS. This leads to improvement in system performance under window-based scheduling strategy over the FCFS scheduling.

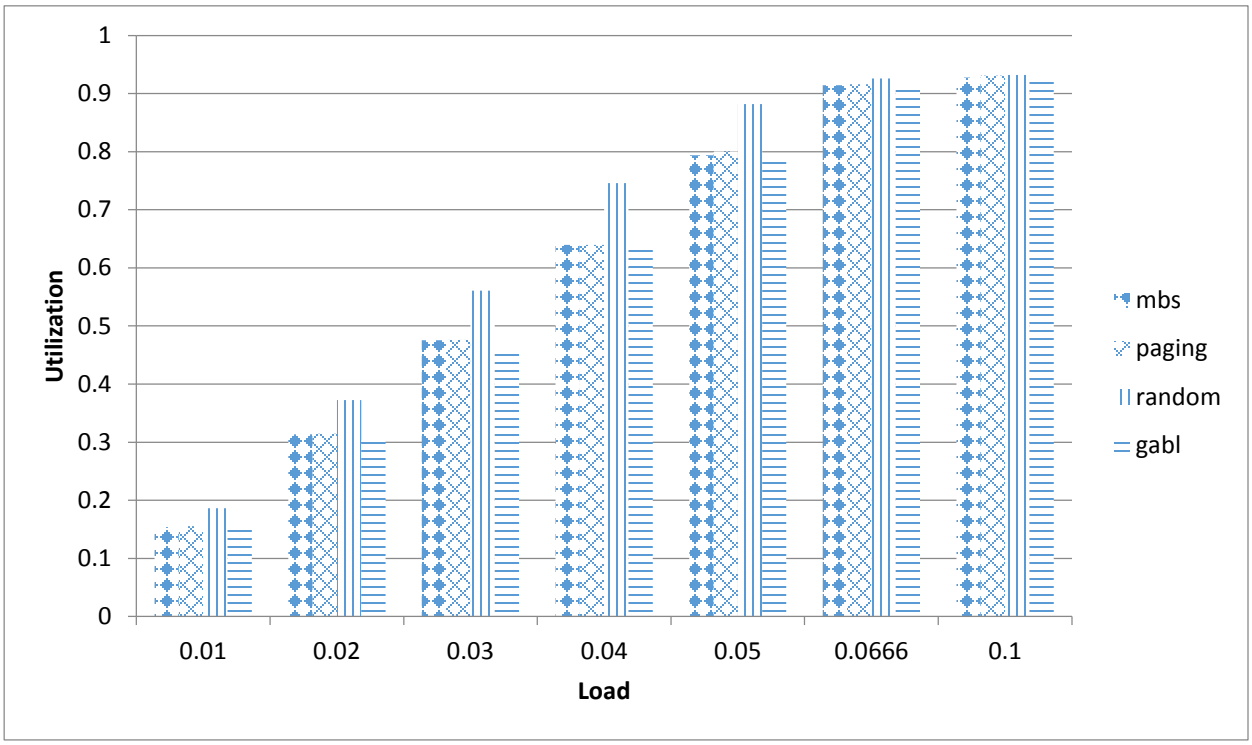

Figure 4.29: Average system utilization of the non-contiguous allocation strategies vs. system load for the random communication pattern under window-based strategy and uniform job size distribution in a 16 × 16 mesh.

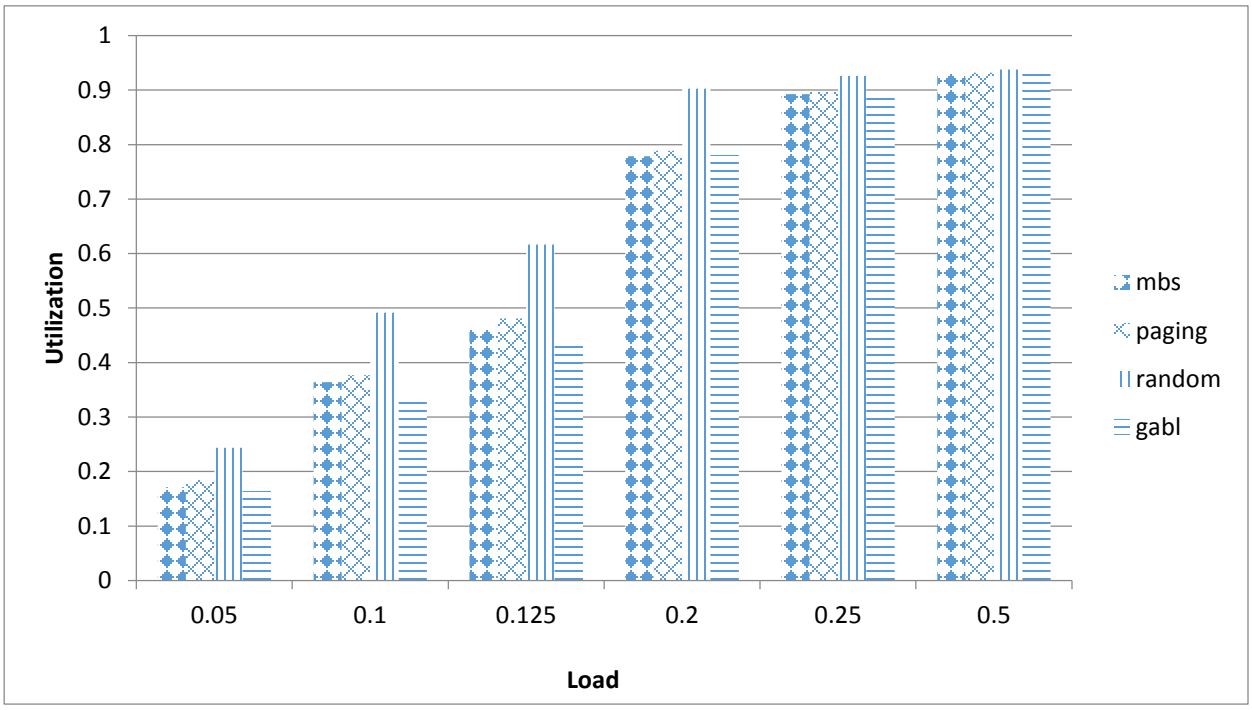

Figure 4.30: Average system utilization of the non-contiguous allocation strategies vs. system load for the random communication pattern under the window-based strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.31 and 4.32, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the near neighbor communication and two job size distributions (i.e., uniform distribution and bounded pareto distribution) under FCFS scheduling. The results for both job size distributions considered in this research show that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. The results also show that the Random strategy is superior to all other non-contiguous allocation strategies for both job size distributions. This is due to the lack of contiguity among the allocated processors in this policy, which increases the job service times and gives good system utilization.

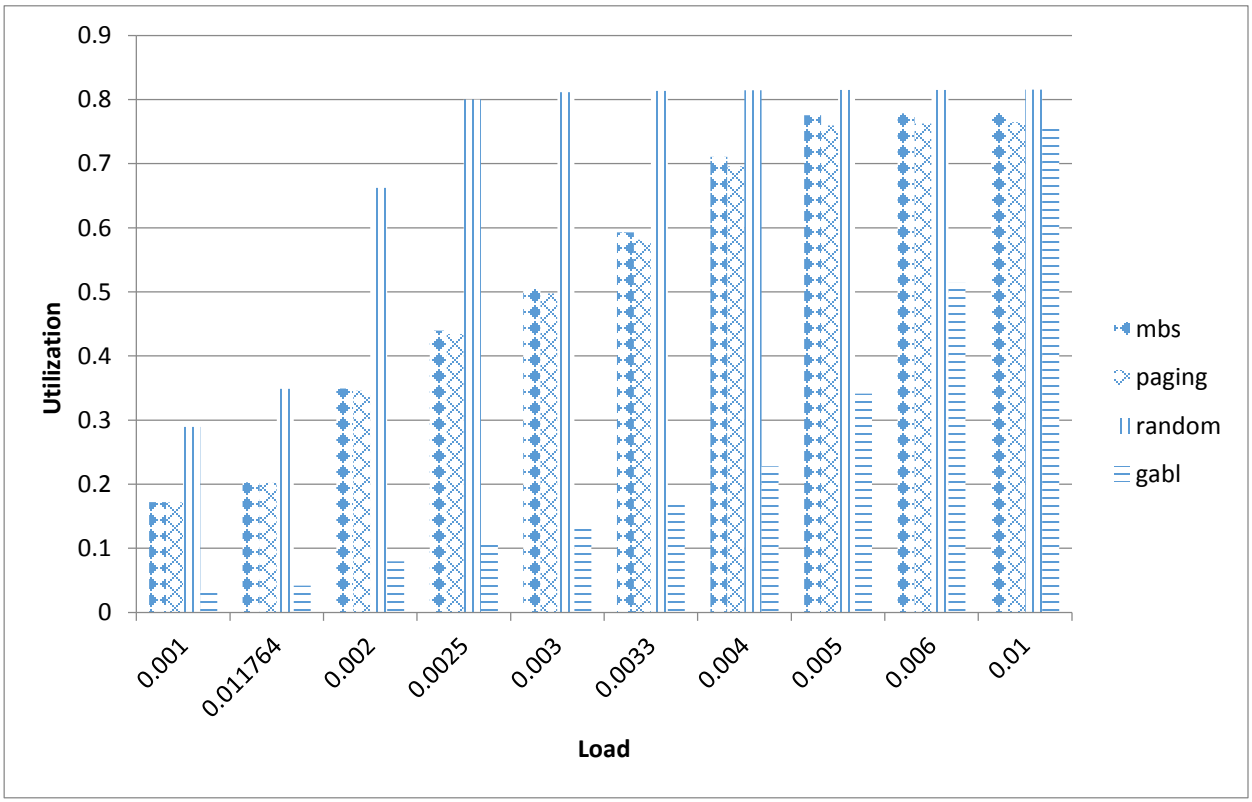

Figure 4.31: Average system utilization of the non-contiguous allocation strategies vs. system load for the near neighbor communication pattern under FCFS strategy and uniform job size distribution in a 16 × 16mesh.

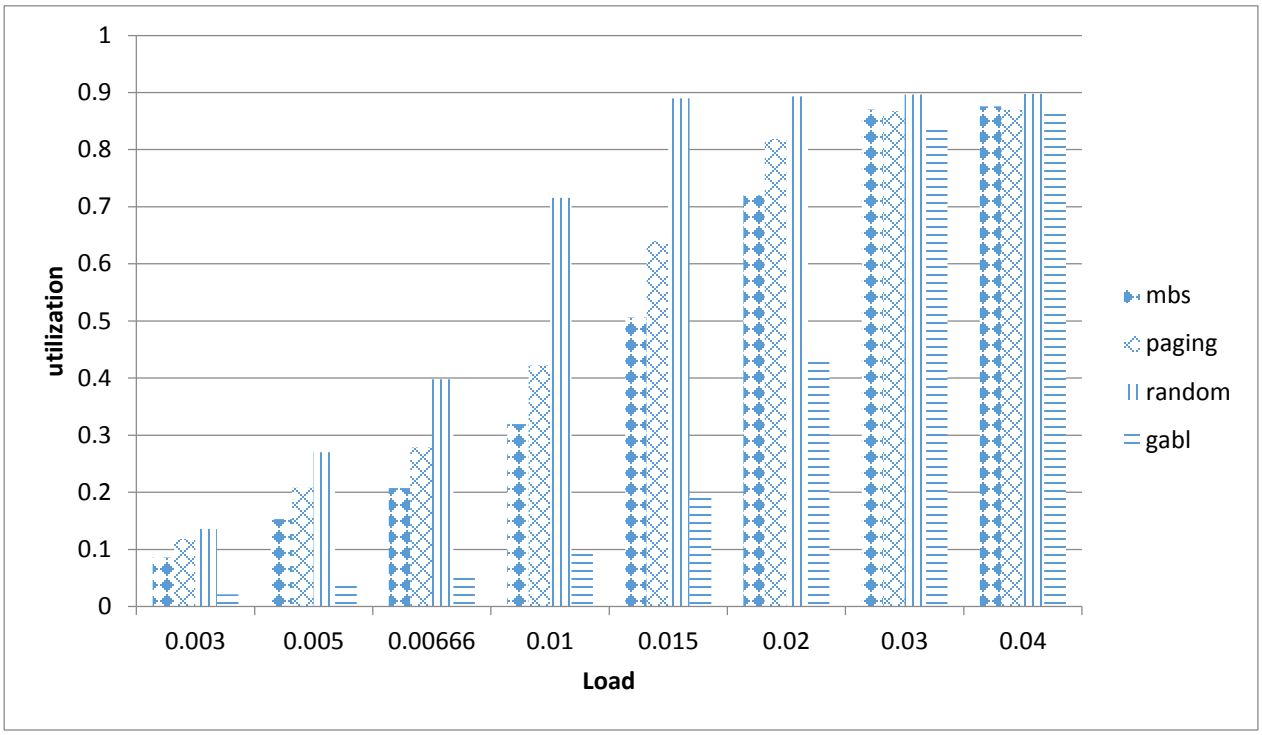

Figure 4.32: Average system utilization of the non-contiguous allocation strategies vs. system load for the near neighbor communication pattern under the FCFS strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.33 and 4.34, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the near neighbour communication and the two job size distributions (i.e., uniform distribution and bounded pareto distribution) under OO scheduling. The results for both job size distributions considered in this research reveal that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto distribution. The results also show that the GABL strategy is superior to all other non-contiguous allocation strategies under both job size distributions for heavy system loads. This is because the set of allocated processors in GABL are not physically contiguous and therefore the probability of inter-job interference is increased and thus increases the turnaround time as shown previously in figure 4.15 which in turn increases the system utilization. Moreover, the results show that the average system utilization of non-contiguous allocation for both job size

distributions under OO is higher than that of non-contiguous allocation under FCFS. This leads to improvement in system performance under OO scheduling over the FCFS scheduling.

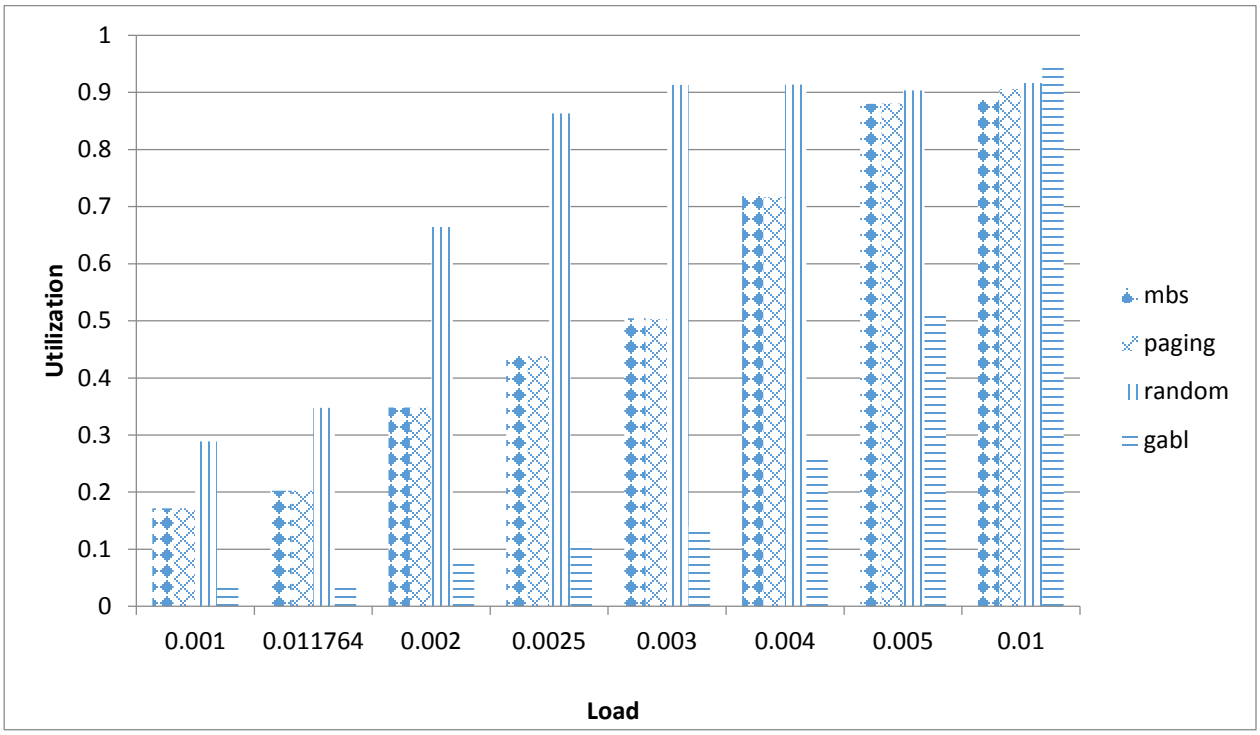

Figure 4.33: Average system utilization of the non-contiguous allocation strategies vs. system load for the near neighbour communication pattern under OO strategy and uniform job size distribution in a 16 × 16mesh.

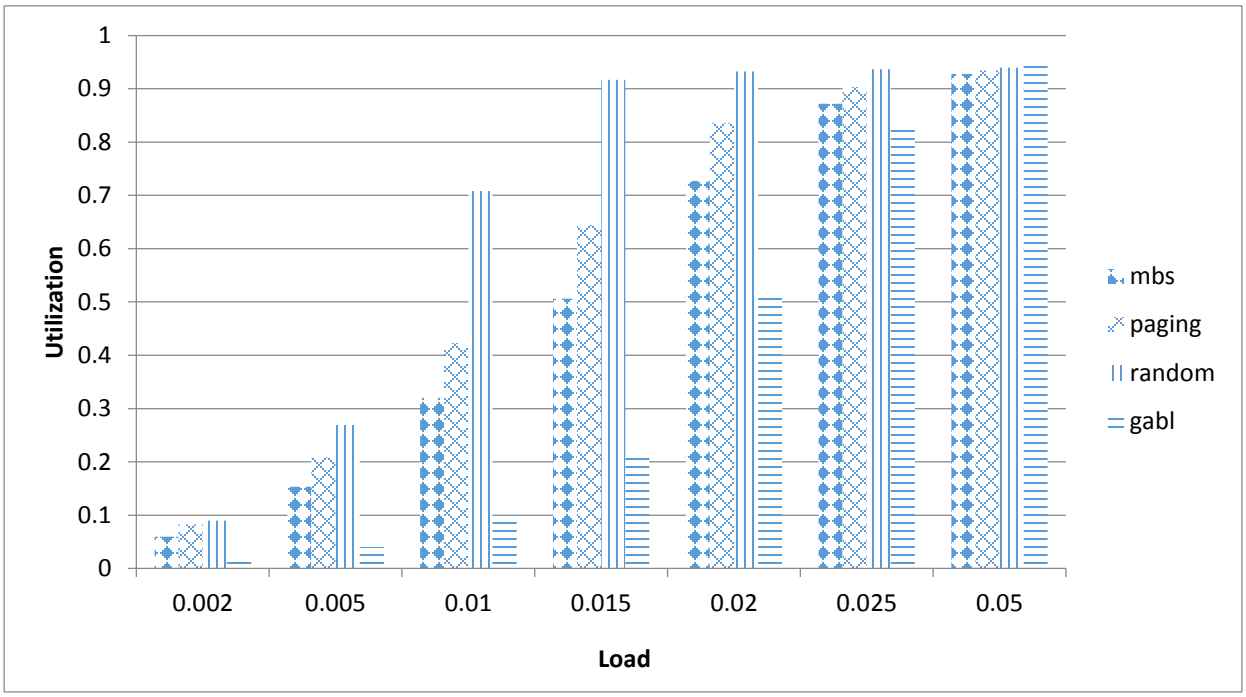

Figure 4.34: Average system utilization of the non-contiguous allocation strategies vs. system load for the near neighbor communication pattern under the OO strategy and bounded pareto job size distribution in a 16 × 16 mesh.

In Figures 4.35 and 4.36, the average system utilization of the non-contiguous allocation strategies is plotted against the system load for the near neighbor communication and the two job size distributions (i.e., uniform distribution and bounded pareto distribution) under window-based scheduling.

We notice from Figure 4.35 and Figure 4.36 that the Random allocation strategy achieves the highest system utilization ratio for most system loads considered. This is due to the lack of contiguity among the allocated processors in this policy, which increases the job service times and gives good system utilization. The results also reveal that the average system utilization of non-contiguous allocation for both job size distributions under window-based is higher than that of non-contiguous allocation under FCFS. This leads to improvement in system performance under window-based scheduling strategy over the FCFS scheduling.

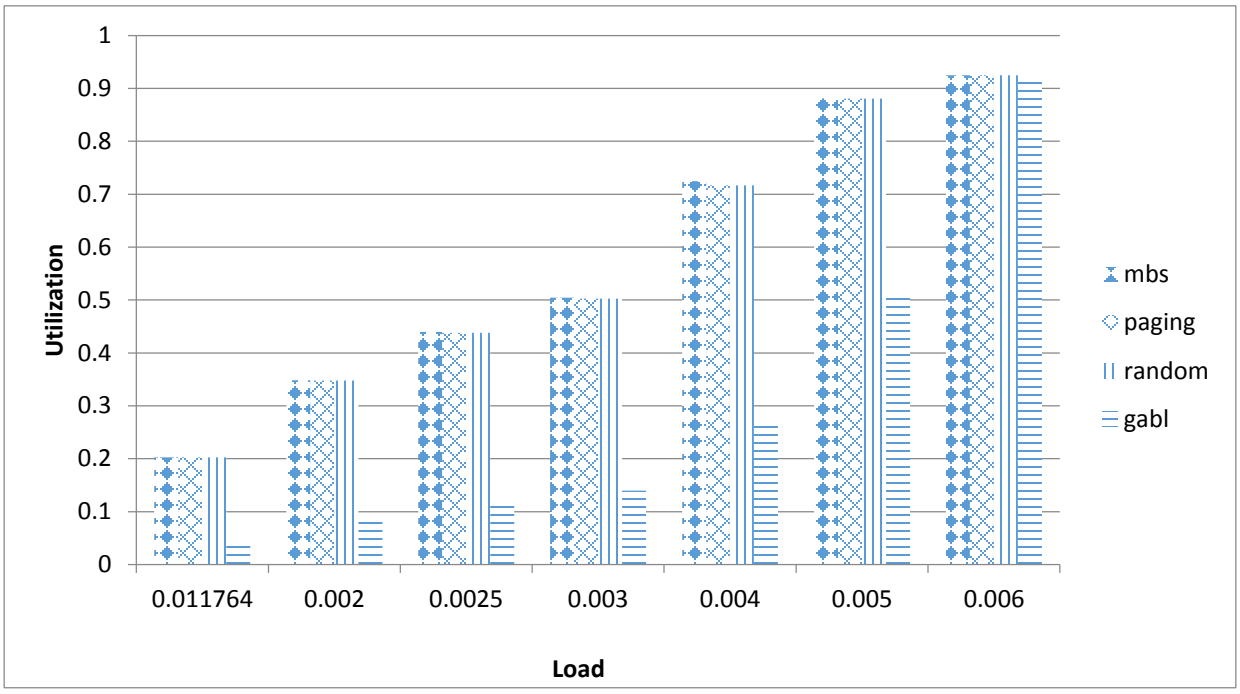

Figure 4.35: Average system utilization of the non-contiguous allocation strategies vs. system load for the near neighbour communication pattern under window-based strategy and uniform job size distribution in a 16 × 16mesh.

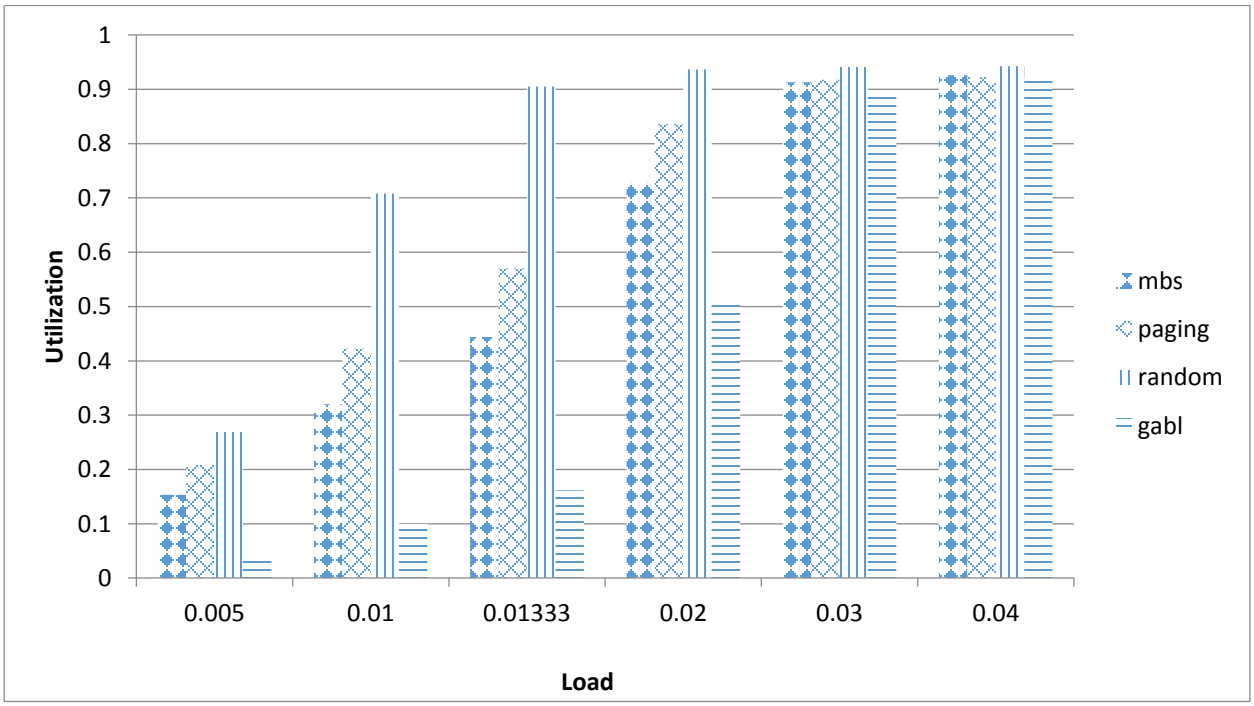

Figure 4.36: Average system utilization of the non-contiguous allocation strategies vs. system load for the near neighbor communication pattern under the window-based strategy and bounded pareto job size distribution in a 16 × 16 mesh.

Ultimately, we notice that the performance of bounded pareto job size distribution is better than that of the uniform job size distribution. However, both job size distributions results demonstrate that the GABL, MBS and Paging(0) perform better than the Random allocation for heavy system loads. The results also prove that the scheduling strategies OO and window-based have better performance than the FCFS scheduling and that the near neighbor communication outperforms all other communication patterns considered in this research work.

Chapter 5: Conclusions and Future Research

5.1 Conclusions

In this research work, we have examined the effect of the bounded pareto job size distribution on the performance of the well-known non-contiguous allocation strategies (Random, GABL, MBS, and Paging (0)) using different communications patterns and scheduling strategies. Moreover, and the simulation results with the bounded pareto job size distribution have been compared against those results with the uniform job size distribution.

We observed that the performance of the non-contiguous allocation strategies is improved when the distribution of job sizes is bounded pareto. Also, the results for the two job size distributions considered (i.e., uniform distribution, bounded pareto distribution) show that the GABL, MBS and Paging(0) allocation strategies outperform the Random allocation strategy. This is because the GABL, MBS and Paging(0) strategies maintain a higher degree of contiguity among the allocated processors than that of the Random strategy. The results also reveal that the scheduling strategies OO and window-based improve the system performance over the FCFS scheduling and that the near neighbour communication has the best performance among the communication patterns considered in this research work.

5.2 Future Research

As a continuation of this research, there are various interesting issues that require further investigation. These may be outlined as follows:

- The performance of the existing allocation strategies has been usually executed by means of simulation based on synthetic workload models to generate a stream of incoming jobs. It would be interesting to evaluate the allocation strategies based on real workload traces from different parallel machines, and to compare the results with the results that we gained in this work.

- In this research work, we have studied the effect of bounded pareto distribution on the performance of the well-known non-contiguous allocation strategies (Random, GABL, MBS, and Paging (0)) that are proposed for 2D mesh connected multicomputers. A natural expansion of this work would be to evaluate the effect of the bounded pareto distribution on the performance of the contiguous and non-contiguous allocation strategies suggested for different interconnection networks.

References

Ababneh, Ismail (2008), Availability-based noncontiguous processor allocation policies for 2D mesh-connected multicomputers, **The Journal of Systems and Software**, 81(7), 1081–1092.

Ahmed-Chandio, A. Zhong-Xu, C. Tziritas, N. Bilal, K. Khan, S. (2013), A Comparative Study of Job Scheduling Strategies in Large-Scale Parallel Computational Systems, **Proceedings of the 12th IEEE International Conference on Security and Privacy in Computing and Communications (TrustCom)**, 949 - 957.

Bani-Ahmad, S. (2011), Processor Allocation with Reduced Internal and External Fragmentation In 2D Mesh-Based Processor, **The Journal of Applied Sciences**, 11 (7), 943-952.

Bani-Mohammad, S. Ababaneh, I. and Yaseen, M (2015), A New Compacting Non-Contigous Processor Algorithm for 2D mesh multicomputers , **The Journal of Information Technology Research**, 8 (4), 1-75.

Bani-Mohammad, S. and Ababneh, I. (2013), On the performance of non-contiguous allocation for common communication patterns in 2D mesh-connected multicomputers, **Simulation Modelling Practice and Theory**, 32, 155-165.

Bani-Mohammad, S. and Ababaneh, I. (2011), A New Window-Based Job Scheduling Scheme for 2D Mesh Multicomputers, **The Journal of Simulation Modelling Practice and Theory**, 19 (1), 482-493.

Bani-Mohammad, S. Ababaneh, I. and Hamdan, M (2011), performance Evaluation of Non-contiguous Allocation Algorithms for 2D Mesh Interconnection Networks, **The Journal of Systems and Software**, 84 (12), 2156-2170.

Bani-Mohammad, S. Ababaneh, I. and Hamdan, M (2010), Comparative Performance Evaluation of Non-Contiguous Allocation Algorithms in 2D Mesh Connected Multicomputers, **Proceedings of the 10th 2010 IEEE International Conference on Computer and Information Technology (CIT 2010)**, 2933-2939.

Bani-Mohammad, S. (2009), The Effect of Heavy-Tailed Distribution on The Performance of Non-contiguous Allocation Strategies in 2D Mesh Connected multicomputers, **Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing**, IEEE Computer Society Press, 1-8.

Bani-Mohammad, S. Ould-Khaoua, M. Ababneh, I. and Mackenzie, L.M. (2009), Comparative Evaluation of Contiguous Allocation Strategies on 3D Mesh Multicomputers, **The Journal of Systems and Software**, 82 (2), 307-318.

Bani-Mohammad, Saad (2008), **Efficient Processor Allocation Strategies for Mesh-Connected Multicomputers**, PhD thesis, The Faculty of Information and Mathematical Sciences, University of Glasgow, Glasgow, UK.

Bani-Mohammad, S. Ould-Khaoua, M. Ababneh, I. and Mackenzie, L.M. (2007a), An Efficient Processor Allocation Strategy that Maintains a High Degree of Contiguity among Processor in 2D Mesh Connected Multicomputers, **Proceedings of the 2007 International Conference on Computer Systems and Applications (AICCSA 2007)**, IEEE Computer Society Press, 934-941.

Bani-Mohammad, S. Ould-Khaoua, M. Ababneh, I. and Mackenzie, L.M. (2007b), A Fast and Efficient Processor Allocation Strategy which Combines a Contiguous and Non-contiguous Processor Allocation Algorithms, **Technical Report Series**, Department of Computing Science, University of Glasgow.

Bani-Mohammad, S. Ould-Khaoua, M. Ababneh, I. and Mackenzie, L.M. (2006), Non-contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers based on Sub-meshes Available for Allocation, **Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06)**, 41-48.

Chiu, G. and Chen, S. (1999), an Efficient Submesh Allocation Scheme for 2D-Meshes with Little Overhead, **IEEE Transactions on Parallel and Distributed Systems**, 10(5), 471-486.

Crovella, M. and Lipsky L. (1997), long-lasting Transient Conditions in Simulation with the Heavy-Tailed Workloads, **Proceedings of the 1997 winter simulation Conference**, 1005-1012.

Chuang, P. and Tseng, N. (1991), an Efficient Submesh Allocation Strategy for Mesh Computer Systems, **Proceedings of the1991 Int'1 Conf. of Distributed Computer Systems**, 10(5), 471-486.

Dillenberger, D. He, L. Jarvis, S. Spooner, D. Jiaung, H. and Nudd, G. (2006), Allocating Non-Real Time and Soft Real Time Jobs in Multicluster, **IEEE Transactions on Parallel and Distributed Systems**, 17(2), 99-112.

Foster, I. (1995), **Designing and Building Parallel Programs Concepts and Tools for Parallel Software Engineering**, First Edition, Addison Wesley, ISBN: 0201575949.

Grama, A. Gupta, A. Karypis, G. and Kumar, V. (2003), **Introduction to Parallel Computing**, Second Edition, Addison Wesley, ISBN: 0-201-64865-2.

Hamdan, M. (2010), **Comparative Performance Evaluation of Non-contiguous Allocation Algorithms in 2D Mesh-Connected Multicomputers**, Master thesis, Prince Hussein bin Abdullah Faculty of Information Technology, Al-alBayt university, Mafraq, Jordan.

Harchol-Balter, M. (1999), the Effect of Heavy Tailed Job Distribution on Computer System Design, **Proceedings of the 1999 ASA-IMS conference on Applications of heavy Tailed Distributions in Economics**, Engineering and Statics, 1-16.

Liu, K. and Cheng, K. (1991), a Two-Dimensional Buddy System for Dynamic Resource Allocation in a Partitionable Mesh Connected System, **J. Parallel and Distributed computing**, 12, 79-83.

Lo, V. Windisch, K.J. Liu, W. and Nitzberg, B. (1997), Noncontiguous Processor Allocation Algorithms for Mesh-Connected Multicomputers, **IEEE Transactions on Parallel and Distributed Systems**, 8(7), 712-126.

Windisch, K. Miller, J.V. and Lo, V. (1995), ProcSimity: An Experimental Tool for Processor Allocation and Scheduling in Highly Parallel Systems, **Proceedings of the1995 Frontiers of Massively Parallel Computation**, 414 – 421.

Yen-Chang, C. and Mohapatra, P. (1998), Performance Improvement of Allocation Schemes for Mesh-Connected Computers, **The Journal of Parallel and Distributed Computing**, 52 (81459), 40-68.

Yoo, B. and Das, C. (2002), A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers, **IEEE Transactions on Parallel & Distributed Systems**, 51(1), 46-60.

Yoo, S. Youn, H. Shirazi, B. (1997), an Efficient Task Allocation Scheme for 2D-Mesh Architectures, **IEEE Transactions on Parallel and Distributed Systems**, 8(9), 934-942.

Zhu, Y. (1992), Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers, **Journal of Parallel and Distributed Computing**, 16(4), 328-337.

ملخص البحث

العديد من الدراسات السابقة استخدمت التوزيع المنتظم لأبعاد الوظيفة أثناء تقييم أداء خوارزميات التخصيص. لذلك تم استخدام التوزيع المنتظم لأبعاد الوظيفة في تقييم أداء خوارزميات التخصيص الغير متجاور. لكن هناك العديد من الدراسات استخدمت توزيع ثقيل الذيل لأبعاد الوظيفة لأنه أكثر واقعيه من التوزيع المنتظم.

في هذه الرسالة قمنا بدراسة تأثير التوزيع الإحصائي ثقيل الذيل على أداء خوارزميات التخصيص الغير متجاور المعروفة ((Random, GABL, MBS, and Paging (0) ضمن استراتيجيات الجدولة المختلفة (First-Come-First-Served (FCFS), Out-of-Order (OO) and window-based) وباستخدام أنماط اتصال مختلفة (الجار القريب، الواحد-للجميع، العشوائي) في متعددات الحواسيب ثنائية الأبعاد. وعلاوة على ذلك، فقد تمت مقارنة تأثير توزيع ثقيل الذيل لأبعاد الوظيفة مع تأثير التوزيع المنتظم لأبعاد الوظيفة، وتمت مقارنة السياسات من خلال معدل الإستخدام الكلي للنظام ومعدل المكوث في النظام باستخدام المحاكي (ProcSimity). وقد أظهرت النتائج أن أداء خوارزميات التخصيص الغير متجاور باستخدام توزيع ثقيل الذيل لأبعاد الوظيفة أفضل من أدائها عند استخدام التوزيع المنتظم لأبعاد الوظيفة، كما أظهرت النتائج بالنسبة لكلا التوزيعين الإحصائيين تفوق أداء كل من الخوارزميات (GABL, MBS, and Paging (0) على أداء خوارزمية (Random)، وأظهرت النتائج أيضا تفوق أداء كل من استراتيجيات الجدولة (OO) and window-based))على أداء إستراتيجية (FCFS) وتفوق نمط الإتصال "الجار القريب" على أداء أنماط الإتصال المختلفة.