

Al al-Bayt University

Prince Hussein Bin Abdullah College for Information Technology

Computer Science Department

The Performance of Paging Non-Contiguous Allocation Algorithm in 5D Mesh-Connected Multi computers

أداء خوارزمية الصفحات للتخصيص غير المتجاور في متعددات الحواسيب الشبكة خماسية الأبعاد

By

Ahmad Hasan Mahmoud Dar Assi

Supervisor

Dr. Saad Bani-Mohammed

Co-supervisor

Dr. Ibrahim Al-Oqily

A Thesis 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

April-2018

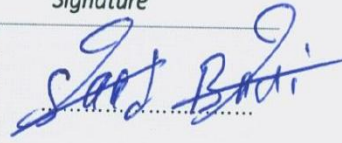
Committee Decision

This Thesis (The Performance of Paging Non-Contiguous Allocation Algorithm in 5D Mesh-Connected Multicomputers) was successfully defended and approved on 25/04/2018.

Examination Committee

Signature

Dr. Saad Bani-Mohammad
(Supervisor)



Prof. Ismael Ababneh



Prof. Omar Shatnawi



Prof. Ahmad Al Khasawneh



Dedication

This thesis is dedicated to my parents, my brothers, my sisters, my wife and my kids for their endless love, support and encouragement.

Acknowledgments

First of all, many thanks to Allah who helped me and gave me the ability to achieve this work. I have been indebted in preparation of this thesis to my main supervisor, Dr. Saad Bani-Mohammed, for his kindness and encouragement helped me all the time of research and writing of this thesis. I am grateful to my co-supervisor, Dr. Ibrahim Al-Qily, for his help, writing suggestions as well as his academic experience.

I would like to express my deep appreciation and gratitude to my family for their support and also, I would thank Dr.Salah Adasi form hashemite univeristy who helpd me in math issues. Thanks from the heart for each one helped in this work.

List of Contents

Dedication	c
Acknowledgments	d
List of Contents	e
List of Figures	g
List of tables	h
Abstract	i
Chapter 1: Introduction	1
1.1 Introduction:.....	1
1.2 Modern 5D Supercomputers :	3
1.2.1 IBM MIRA supercomputer :.....	3
1.2.2 Fermi Supercomputer :	4
1.3 Interconnection networks	4
1.3.1. 1D, Line or Ring topology :.....	4
1.3.2. 2D Torus or Mesh topology :.....	4
1.3.3. 3D Cube and 4D hypercube topologies :.....	5
1.4 Communication Patterns :.....	6
1.4.1 ALL-To-ALL communication pattern :.....	6
1.4.2 Near Neighbor communication pattern :.....	6
1.5 Motivational contribution :	6
Chapter 2: Related work	7
2.1 Related work:.....	7
2.2 Allocation Strategies :	7
2.2.1. Multiple Buddy Strategy (MBS) :	7
2.2.2. Random Processor Allocation Strategy :.....	7
2.2.3. Paging Processor Allocation Strategy :	8
Chapter 3: Paging allocation strategy on 5D topology	9
3.1 K-ary N-cube :.....	9
3.1.1 K-ary N-cube Properties :.....	9
3.2 Preliminaries :	9
3.3 Paging in 5D :	12
3.3.1 5D Paging Initialization :.....	12
3.3.2 5D Paging Allocation :.....	13

3.3.3 5D Paging Deallocation :.....	15
3.4 Mapping in 5D :.....	16
Chapter 4: Performance and Evaluation.....	17
4.1 Assumptions :	17
4.2 Simulator :	17
4.3 Why Simulation?.....	18
4.4 Simulation results :	18
4.4.1 Simulation parameters :	18
4.4.2 Turnaround time :	20
4.4.3 Performance Impact of Mesh System Size :.....	22
4.4.4 Utilization :	23
4.5 Conclusion :	26
Chapter 5: Conclusion and Future Directions.....	27
5.1 Summary of the results :	27
5.2 Future work directions :.....	27
References.....	28
الملخص باللغة العربية.....	32

List of Figures

Figure	Page
Figure (1): Example of 8x8 mesh system	2
Figure (2): Example of 5D network for $2^5=32$ processor	3
Figure (3): Example of 1D topology of $2^1=2$ processors	5
Figure (4): Example of 2D topology of $3^2=9$ processors	6
Figure (5): Example of 3D topology of $(2)^3=8$ processors	7
Figure (6): Example of 4D topology of $2^4=16$ processors	7
Figure (7): 5D system with $K=9$ and $n=5$ with 59049 nodes	14
Figure (8): 5D sub cube communication links in case of mesh	16
Figure (9): 5D communication links in case of turs	17
Figure (10): FPL initialization	18
Figure (11): paging process allocation	20
Figure (12): paging deallocation process	21
Figure (13): five dimensions to linear and linear to five dimensions coordinates	22
Figure (14): turnaround time vs. system load using uniform distribution and near neighbor communication with 59049 system size	29
Figure (15): turnaround time vs. system load using uniform decreasing distribution and near neighbor communication with 59049 system size	29
Figure (16): turnaround time vs. system load using uniform distribution and ALL TO ALL communication with 59049 system size	30
Figure (17): turnaround time vs. system load using uniform decreasing distribution and ALL TO ALL communication with 59049 system size	30
Figure (18): turnaround time vs. system load using uniform decreasing distribution and near neighbour communication with 1024 system size	31
Figure (19): system utilization vs. system load using uniform distribution and near neighbor communication with 59049 system size	33
Figure (20): turnaround time vs. system load using uniform decreasing distribution and near neighbor communication with 59049 system size	33
Figure (21): system utilization vs. system load using uniform distribution and ALL To ALL communication with 59049 system size	34
Figure (22): system utilization vs. system load using uniform decreasing distribution and ALL To ALL communication with 59049 system size	34

List of tables

Table	Page
Table 4.1: System Parameters that used in simulation process	26

Abstract

Single core systems had arrived to a bottleneck and the enhancement has become limited, but the applications that are developed these days need high speed computers with multi processors to solve more complex and larger problems. To achieve this, we need efficient allocation strategies to allocate and deallocate processes attached to such systems. Processor allocation strategies can be affected by the processors' topology, which can impact system performance in terms of job turnaround time and system utilization. In this thesis, we propose to enhance the job turnaround time and system utilization using a higher dimensional topology thus improving the system performance with low additional cost. Higher dimensional topologies have more links between nodes. This will not only reduce the message passing overhead but also will increase the degree of contiguity between processors, and it is expected to increase the probability of successful allocation and hence improves the system performance in terms of both job turnaround time and system utilization. Simulation evaluation shows that the performance of the system is improved while increasing the dimensions of the topology from $2D$ to $5D$ without increasing the number of processors.

,

Chapter 1: Introduction

1.1 Introduction:

Parallel computers are widely used for many applications that could not be run efficiently on a single core system. Supercomputers have many cores that run together to make job turnaround time much smaller than that in the single core systems, where the job turnaround time is the time that the job spends in the system from arrival to departure. However, the number of processors is not the only factor that influence the speed of supercomputers; the topology of the communication network is of special importance in this regard. Cores or processors in supercomputers are arranged in shapes called topologies that can handle issues related to communications and bandwidth [38].

Allocation of processors to a certain task can be contiguous or non-contiguous. In contiguous allocation, processors allocated to a job are physically contiguous and have the same shape as the job request, while in non-contiguous allocation, a job can execute on multiple disjoint smaller sub-meshes rather than always waiting until a single sub-mesh of the requested size and shape is available [38].

Contiguous allocation strategies are motivated by security issues and low message overhead since the distance between the cooperated processors is almost zero, but it suffers from internal fragmentation and external fragmentation, which affects on system performance in terms of system utilization and job turnaround time, where system utilization is the percentage of processors that are utilized over time. Internal fragmentation occurs when more processors are allocated to a job than it requires, where external fragmentation occurs when there are sufficient number of processors to satisfy a job request but they are not allocated to it because they are not contiguous [38]. Among the previous non-contiguous allocation strategies, paging is one of the simple and flexible non-contiguous allocation strategies that can be controlled by the page size in order to offer some contiguity and thus alleviate processor fragmentation [42].

The $2D$ and $3D$ topologies are widely used in multicomputer systems. Many applications were designed for these topologies. However, the $2D$ mesh networks are more widely used because of its simplicity and scalability [5, 12, 22]. Two-dimensional mesh systems are a very good choice for many applications such as dot matrix multiplication and image processing because they are shaped in the real world as $2D$ mesh systems [40]. The shape of the $2D$ mesh topology is rectangular. Figure 1 shows a squared $2D$ mesh, where width and height are the same (the same number of processor in each side).

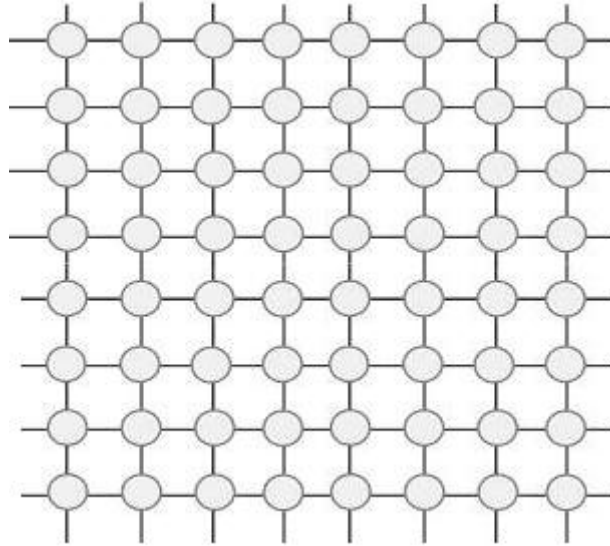


Figure (1): Example of 8x8 mesh system

If the topology is square, then it can be described as K -ary N -cube which can be represented as $N = k^n \rightarrow (n = \text{Log}_k N)$ [43], where n is the number of dimensions and k is the number of processors in each dimension and N is total number of processors in the system [43]. In case of 32×32 $2D$ mesh system, it can be represented in k -ary n -cube as follows $N = 32^2$. However, if we increase k to 5 dimensions, it becomes $4 \times 4 \times 4 \times 4 \times 4$, which is a $5D$ network, where each dimension has 4 processors and the k -ary n -cube formula becomes $1024 = 4^5$. In this case, $k = 4$ and $n = 5$ and $N = 1024$. This is shown in figure 2, which means that we can convert the $2D$ to a $5D$ network. This conversion implies more links and more paths to deliver messages faster and also paths becomes shorter.

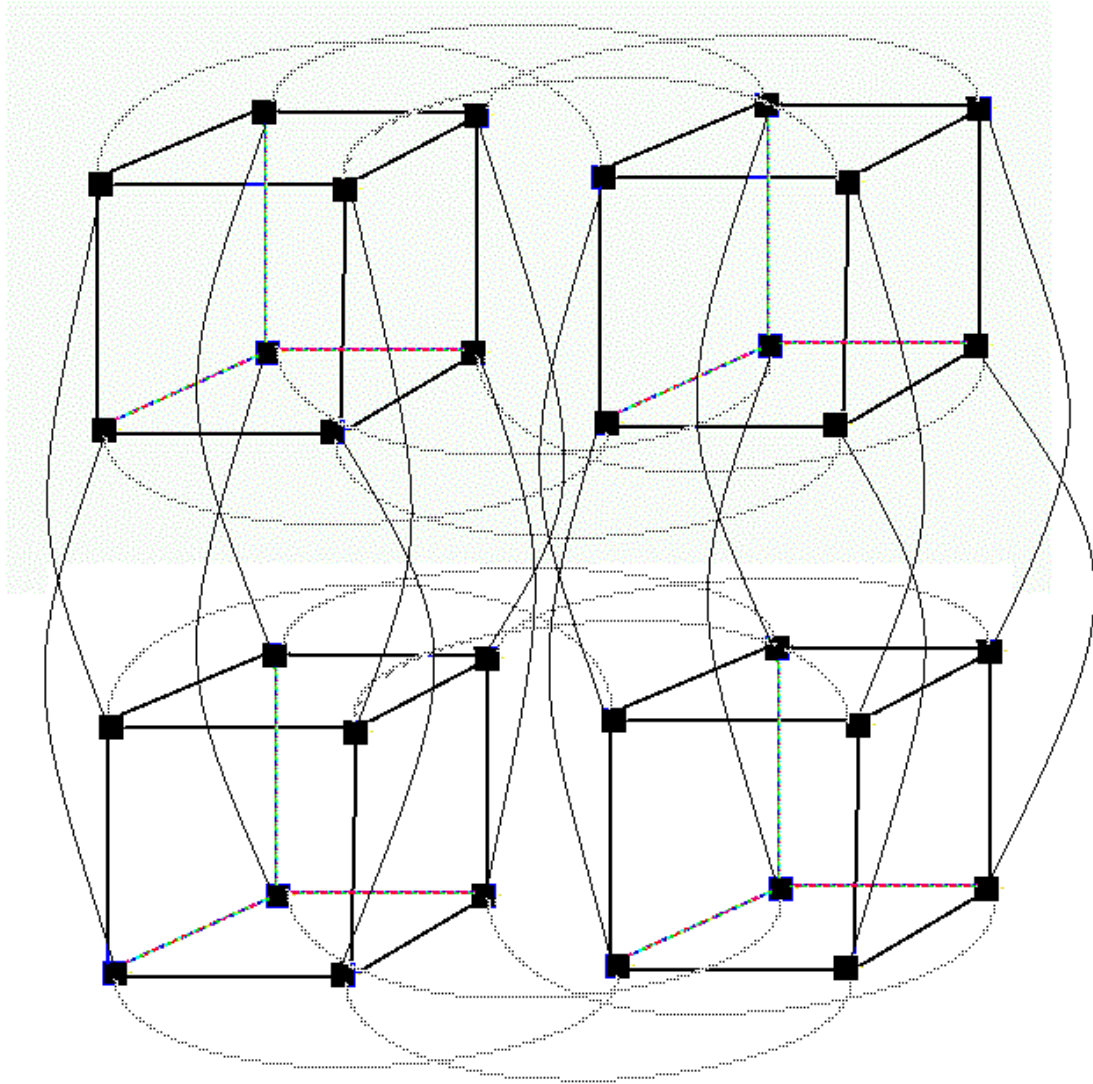


Figure (2): Example of 5D network for $2^5 = 32$ processor [39]

1.2 Modern 5D Supercomputers :

Many of modern super computers were designed as 5D mesh connected multicomputers and took place in TOP 500 where this list classified as super computers and this is based on powerful of computation and below are examples of these modern supercomputers.

1.2.1 IBM MIRA supercomputer :

IBM MIRA is a supercomputer that placed in Argonne National Laboratory that built as 5D mesh-connected supercomputer, where it contains 786,432 cores that consumed 3.9-Megawatt, system physical size is $1,632^2$ feet. This system has 768 TiB of memory and its speed reaches to more than 10 petaFLOPS, IBM MIRA Ranked 3 on TOP500 when built [49].

1.2.2 Fermi Supercomputer :

Fermi is a supercomputer that placed in CINECA, Casalecchio di Reno, Italy which is sponsored by ministry of education for universities and research. Fermi built as 5D Tours-connected supercomputer, where it contains 10.240 compute nodes and 163.840 cores that consumed 822 Kilowatt with 160 TiB of memory, where its speed is more than 2 petaFLOPS, Fermi ranked 7th on TOP500 when built [50].

1.3 Interconnection networks

Many interconnection networks are proposed for multicore systems, some of them were widely used because they are cheap and simple. The topology of the network effects on overall network cost (latency and turnaround time), where latency is the time that the message takes to travel through the network from source to destination [38].

1.3.1. 1D, Line or Ring topology :

As shown in figure 3, this topology consists of the number of processors that are shaped in line or ring where each processor has up to two neighbors and also up to two links or edges. This topology suffers from large number of nodes that messages have to pass through to arrive to its destination and thus increases the turnaround time, the system latency and power consumption and hence lower the reliability [38].

1.3.2. 2D Torus or Mesh topology :

As shown in figure 4, this topology consists of a number of processors that are shaped in mesh where each processor has up to 4 neighbors and also up to 4 links or edges. This topology has better performance than the 1D topology, because of the more links and more neighbors that make the system exchanges messages faster [38]. When additional link is added between each ends, then it becomes tours and in such case, the number of neighbors for each processor is 4, however mesh topology suffers from high diameter, which is the distance between the farthest two processors in the network [3].

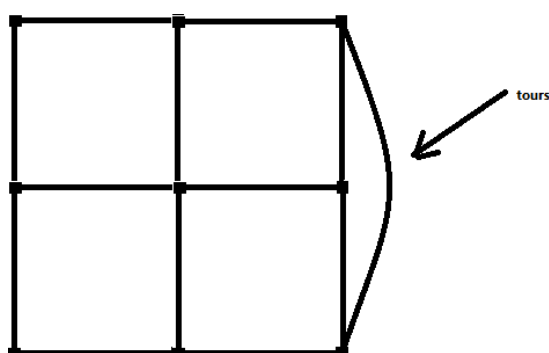


Figure (4): Example of 2D topology of $3 \times 3 = 9$ processors

1.3.3. 3D Cube and 4D hypercube topologies :

3D and 4D topologies as shown in figures 5 and 6 have more dimensions. In case of 3D (cube), the node or processor can be represented in 3 coordinates (x, y, z) , where each processor has up to 6 links or edges in case of 3D mesh and exact 6 links in case of 3D tours, which makes performance better than the 2D mesh or tours [1]. In case of 4D (hypercube), there are 4 dimensions and the nodes or processors are represented by 4 coordinates (x, y, z, w) , where each processor has up to 8 links or edges in case of mesh and exact 8 in case of tours. These extra dimensions can give us more options for system design and hence more links to deliver messages faster and hence bottleneck avoidance. Moreover, the diameter is lower than that of 3D, but it is more complicated and have more issues in scalability where it makes 4D is limited in use and thus is more suitable for only small systems with small configuration changes [3, 4].

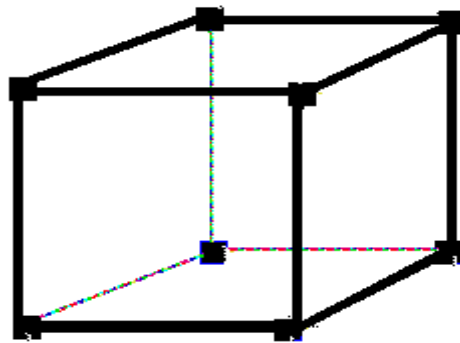


Figure (5): Example of 3D topology of $2^3 = 8$ processors

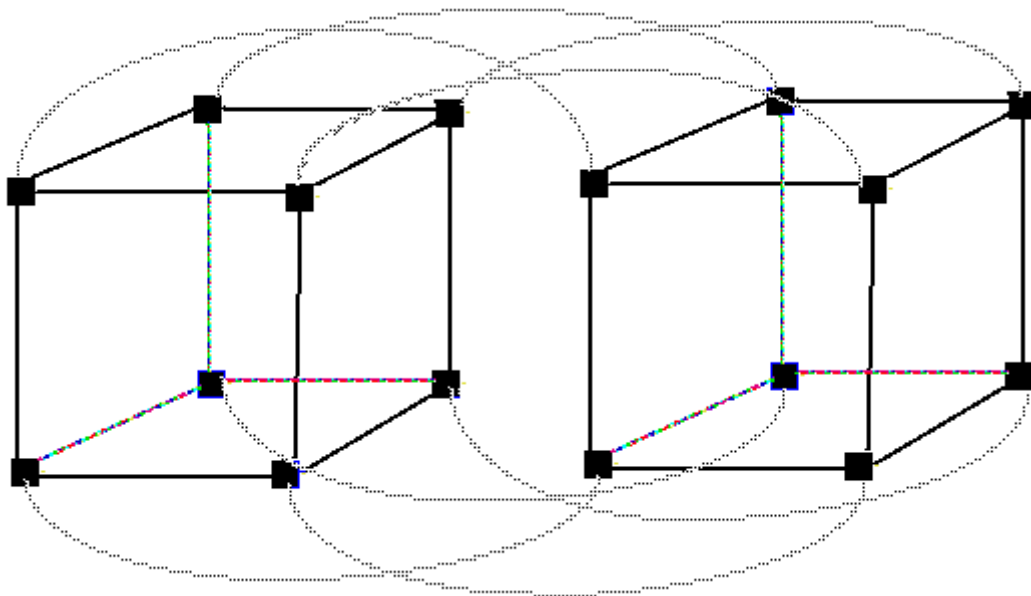


Figure (6): Example of 4D topology of $2^4 = 16$ processors

1.4 Communication Patterns :

In non-contiguous allocation strategies, processors within a job need to send and receive messages from each other's. In this thesis, we use two different patterns for communication, All-To-All and Near-Neighbor.

1.4.1 ALL-To-ALL communication pattern :

In this communication pattern, all processors need to transmit messages to all other processors which produces much messaging overhead, so this communication pattern is considered as a weak point for the non-contiguous allocation strategies. All-To-All is common communication pattern and it is used for many and important applications such as matrix-multiplication [18].

1.4.2 Near Neighbor communication pattern :

In this communication pattern processors need to transmit messages to its neighbors only which produces less messaging overhead than all-to-all communication. Moreover, the possibility of message interference is much smaller than that in all-to-all communication. Near Neighbor is very common communication pattern and it is used to simulate many of physical cases such as wave propagation and heat [37].

1.5 Motivational contribution :

Modern Huge systems need more powerful allocation strategies. The current systems use efficient allocation strategies but the performance of these strategies is limited for huge systems, and this is due to the higher communication overhead that results when the number of processors is increased in these systems. In this thesis, we aim to adapt these strategies to be applicable for modern huge systems.

To the best of our knowledge, there is no proposed approach for paging non-contiguous allocation strategy that is implemented for interconnection networks with dimensions (d) greater than 3. Therefore, the contribution of this thesis is to revisit one of the existing non-contiguous allocation strategies for interconnection networks with $d > 3$. Simulation evaluation shows that the performance is increased for high dimensions topologies. This is due to faster turnaround time and higher system utilization achieved and to the lower diameter when $d > 3$.

Chapter 2: Related work

2.1 Related work:

Many of non-contiguous allocation strategies are proposed for both 2D and 3D topologies such as paging [42], Random [42] and MBS [42], where these strategies are tested and evaluated based on synthetic and real workload models using simulation. In the following subsections, we will shed a light on these allocation strategies.

2.2 Allocation Strategies :

When a job enters the system, it is allocated to a group of processors to be executed. The way the system selects the set of processors to be allocated for a job request is called processor allocation strategy, which aims to avoid processor fragmentations as much as possible. However, there is no any allocation strategy that can eliminate the fragmentations without any additional cost [40].

2.2.1. Multiple Buddy Strategy (MBS) :

Multiple Buddy Strategy [42] is a non-contiguous allocation strategy, it is better than the older version (2D buddy strategy) which has more external and internal fragmentation [42].

This strategy maintains a degree of contiguity between allocated processors [42], This strategy builds the initials blocks by dividing the original mesh into small square shaped meshes with number of processors in each side is power of two, such as $3^2 \times 3^2$ or $4^2 \times 4^2$ with different total number of processors, and these blocks (sub-meshes) are stored as records in a list called free block record (FBR), where the (FBR) is a list that contains zero or more blocks, and each block in the (FBR) is represented as $\langle x, y, p \rangle$, where (x, y) represents the coordinates of the blocks and (p) represents the number of processors in each side, where the total number of processors in one sub mesh is p^2 . If $p > 1$ then the block is divided into smaller blocks such as $\langle x, y, \frac{p}{2} \rangle$, $\langle x + \frac{p}{2}, y, \frac{p}{2} \rangle$ and $\langle x, y + \frac{p}{2}, \frac{p}{2} \rangle$. These blocks (buddies) are stored in an ordered list called `block_list` and a variable called `block_num` is used to store the number of these blocks. The total number of free processors in the system is stored in a variable called `AVAIL` and the job is allocated only if the number of requested processors by the job is less than or equal to `AVAIL` [42].

2.2.2. Random Processor Allocation Strategy :

This strategy chooses the processors randomly. It works only on the number of processors that the job needs. If the available number of processors meet the required number of requested processors, then it will let the job to be allocated in the system with randomly chosen processors but without considering the contiguity condition.

This leads to high system utilization, where the system often busy but in the other hand message passing overhead may become very high and this makes job to stay longer in the system, and hence the job turnaround time is increased. In this strategy, allocation only fails if the number of available processors in the mesh system is less than the number of processors requested by the job [42].

2.2.3. Paging Processor Allocation Strategy :

In paging allocation strategy, the system is divided into pages, so the job requests a specific number of pages, where page size is the total number of processors in each page [22, 42].

Since paging strategy divides the system into pages that have in each side a number of processor 2^{page_size} , page becomes the unit of allocation, where the total number of processors (n) are calculated as follow:

$$n = (2^{page_size})^d$$

Where $page_size$ is a positive integer ≥ 0 and d is the number of dimensions [42]. When d increases, the strategy shows better performance and this is due to lower diameter and average communication distance of the systems with higher dimensions [35].

In paging strategy, there is a list called free page list (FPL). FPL is a data structure that includes an index that represents the available free pages in the system and also it includes the coordinates of these pages [42]. The number of allocated pages for a job that requests a number of processors (k) is calculated as follow:

Number of pages = $\lceil k/n \rceil$ [42]. n is the number of processors in each page.

Paging strategy has been selected in this study because it has been shown in [30, 51] to perform well as compared to other strategies.

Chapter 3: Paging allocation strategy on 5D topology

Most of multicomputers interconnection networks are built as K-ary N-cube or similar to K-ary N-cube, when all of the interconnection network dimensions are equal then the network is represented as K-ary N-cube, the following section is a brief description for some of K-ary N-cube properties [44].

3.1 K-ary N-cube :

k – ary *n* – cube [9, 43, 44, 47] is a graph where every node in this graph can be denoted by *n* – coordinates with base *k*, where *n* is number of dimensions and *k* is the number of nodes in each dimension.

3.1.1 K-ary N-cube Properties :

The total number of nodes in *k* – ary *n* – cube system is k^n .

There is a *K* sub cubes in *k* – ary *n* – cube systems where the number of edges for each sub cubes connected to each other's is k^n where $k \geq 3$ and k^{n-1} when $k = 2$.

Every node in *k* – ary *n* – cube has the same degree (number of edges that connected to the same node) where the degree is *n* for $k = 2$ and $2n$ where $k \geq 3$.

The total number of links (edges) in *k* – ary *n* – cube is nk^{n-1} where $k=2$ and nk^n where $k \geq 3$

3.2 Preliminaries :

In this thesis, we assume the system is a 5D mesh-connected system, which is denoted by $S(X, Y, Z, W, I)$, where *X* is the height of the system, *Y* is its width, *Z* is its depth and *W, I* are the fourth and fifth dimensions. Our system can be viewed as *k* – ary *n* – cube by making $K = X = Y = Z = W = I$ and make it torus rather than mesh (extra round link). Each node can be found in our system with five coordinates as $n(x, y, z, w, i)$, where $1 \leq x \leq X, 1 \leq y \leq Y, 1 \leq z \leq Z, 1 \leq w \leq W$ and $1 \leq i \leq I$. In other words, $1 \leq x, y, z, w, i \leq k$, where each node in the system are connected to exact 10 neighbors ($2n, 2 \times 5 = 10$). The total number of system nodes are $N = (XxYyZzWwI)$, which in our case $N = K^5$ and $K = 9$ and total number nodes in our system is $N = 9^5$ nodes. Figure 7 shows our target system.

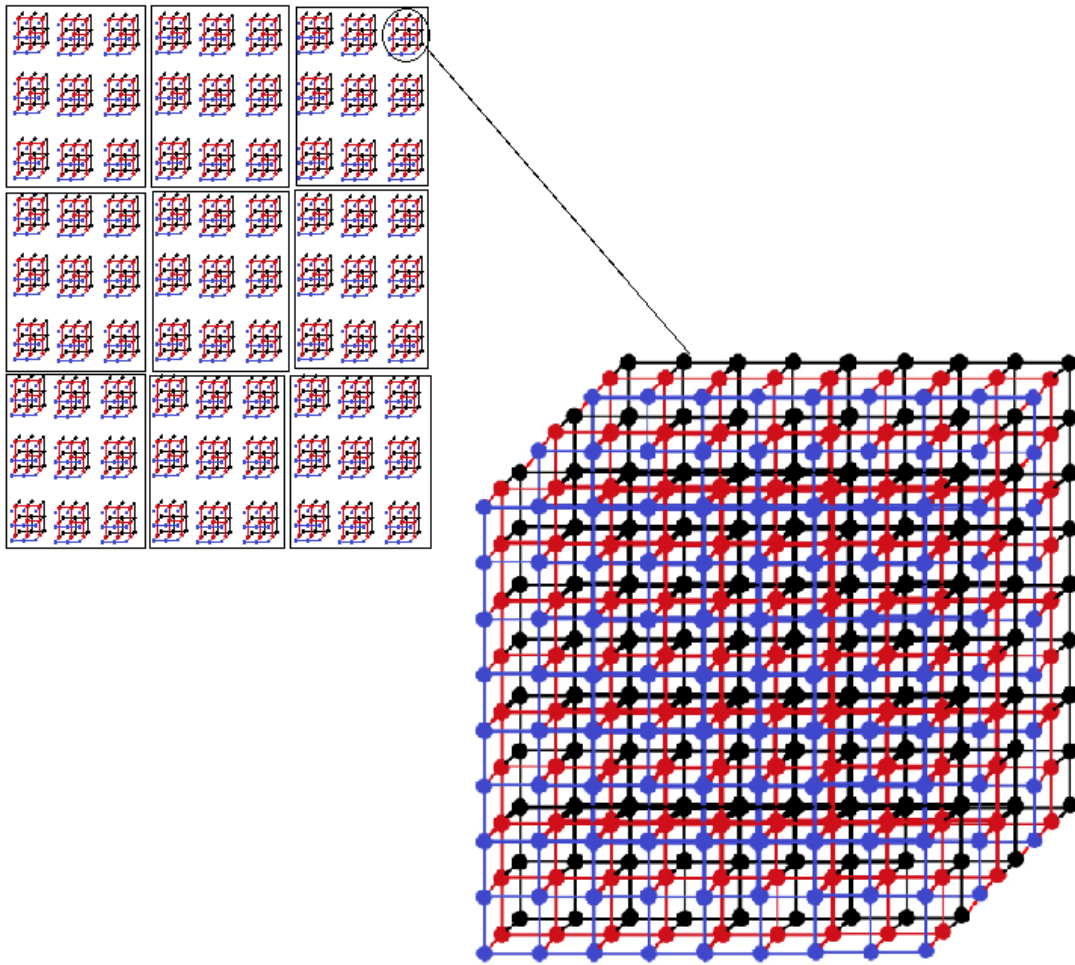


Figure (7): 5D system with $K = 9$ and $n = 5$ with 59049 nodes.

As shown in figure 8, $5D$ has 7 links in corner where always this point has less communications links, but in figure 9 there is exact 10 links in corner in $5D$ sub cube, and this is for tours mesh, where every node has the same number of communications links.

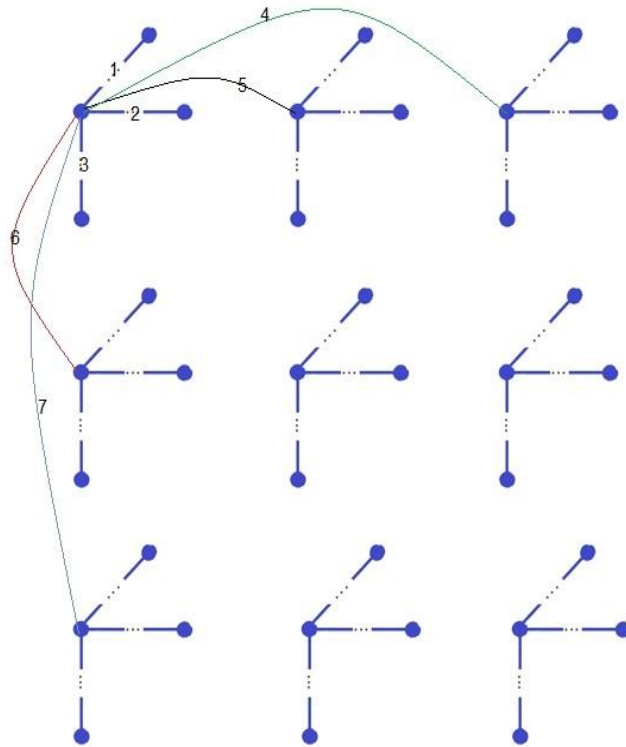


Figure (8): 5D sub cube communication links in case of mesh

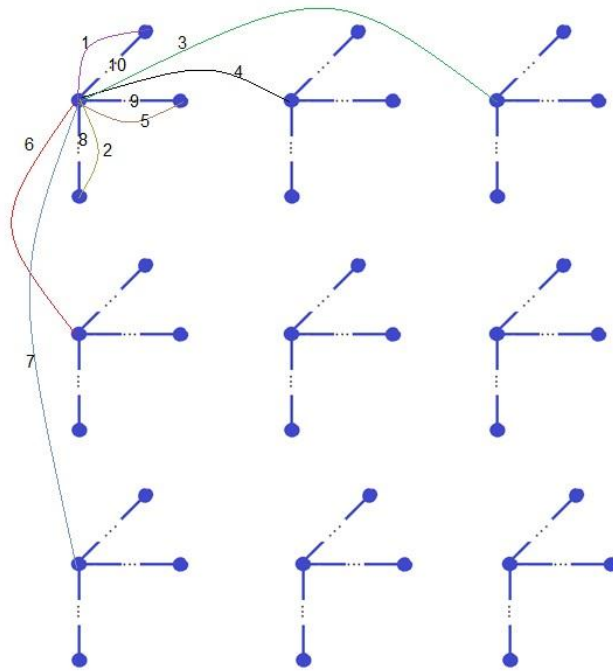


Figure (9): 5D communication links in case of turs

3.3 Paging in :

In our target system, *page_size* is 0 which is the best choice to eliminate the internal processor fragmentation [47].

3.3.1 5D Paging Initialization :

Every node in our system is a page, since $2^0 = 1$. Initially, every node is free and it is placed in FPL (Free Pages List) with its coordinates. Figure 10 shows how to initiate the FPL.

```

Procedure System_initialize(K)
    begin
         $i = 0, j = 0, k = 0, l = 0, m = 0;$ 
         $page\_size = 0;$ 
         $page\_processors = 2^{page\_size};$ 
         $AVAIL = Total\_Processors / page\_processors;$ 
         $block\_num = 0;$ 
         $FPL = empty();$ 

        for ( $i = 0; i < K; i++$ )
            for ( $j = 0; j < K; j++$ )
                for ( $k = 0; k < K; k++$ )
                    for ( $l = 0; l < K; l++$ )
                        for ( $m = 0; m < K; m++$ )
                             $index = block\_num;$ 
                             $block\_num++;$ 
                             $FPL.add(index, m, l, k, j, i);$ 
    end

```

Figure (10): FPL initialization.

3.3.2 5D Paging Allocation :

As we mentioned before, every processor or node is a page in our system since page size is zero, so, for every process request we need to know the number of requested processors and its unique id *process_id*. Initially, the system will check its own pages if they can cover the requested number of processors or not, and if not, the system will never let the process to be allocated and that called system insufficiency.

If the system size is larger than the requested number of pages, then the system will check the number of free available pages (AVAIL). If the AVAIL is larger than the requested number of pages, then the system will let the process to be allocated. Moreover, the system will insert the *process_id* and the requested pages into the data structure called *Task_list*, and then the system will remove the requested pages from the Free Page List (FPL) and the AVAIL will be decreased by the number of requested pages. Otherwise, the process will wait until the AVAIL is larger than the number of requested pages. Figure (11) shows the procedure for the process allocation.

Procedure Page_5D_allocate(process_id, process_size)

```
begin  
  
if(avail ≥ process_size)  
{  
  
Task_list.add(process_id, pages);  
Avail = Avail – process_size;  
FPL.remove(pages);  
Take_another_process();  
  
}  
  
Else  
{  
Take_another_process();  
}  
  
end
```

Figure (11): paging process allocation

3.3.3 5D Paging Deallocation :

Every allocated process has a unique identifier called *process_id*, and then each of the allocated pages for this process is marked with *process_id* and these pages are inserted into a list called *Task_List*, which means that these pages are busy and allocated for this process until its completion. When the process is completed, the system searches for all pages that marked with *process_id* in *Task_List* and free them to be used by another process and then these pages are returned to *FPL* and *AVAIL* is increased by this number of pages. Figure 12 shows paging deallocation process.

```
Procedure Paging_5D_deallocate(process_id)  
    begin  
  
        for each process in Task_List  
            if (process.process_id != process_id)  
                {  
                    process = Task_list.next();  
                }  
            else  
                break;  
            end for  
  
            Task_List.remove(process);  
            FPL.add(process.pages);  
            AVAIL = AVAIL + process.size();  
  
    end
```

Figure (12): paging deallocation process.

3.4 Mapping in :

In this section, we describe briefly how mapping is carried out in 5D based on 2D and 3D. When a process is allocated, all its pages are stored in a linear mapping array that shows the neighbors for each allocated page in the system [35], where the pages that are allocated for any process are neighbors to each other's in the mapping array and they can communicate until the process is completed. We convert all the coordinates of pages to be linear in order to store them easily in the mapping array and also we convert them back to its original coordinates when that is needed. Figure 13 shows how conversion is done.

```
Procedure linear(Page, K)  
begin  
D = 0;  
D = D + (K4 * Page.x);  
D = D + (K3 * Page.y);  
D = D + (K2 * Page.z);  
D = D + (K * Page.w);  
D = D + (Page.i);  
return D;  
end
```

```
Procedure FiveD(D, K)  
begin  
Page = EmptyPage();  
Page.i = D mod K;  
Page.w = (D / K) mod K;  
Page.z = (D / K2) mod K;  
Page.y = (D / K3) mod K;  
Page.x = (D / K4) mod K;  
return Page;  
end
```

Figure (13): Five dimensions to linear and linear to five dimensions mapping.

Chapter 4: Performance and Evaluation

In this chapter, the simulation has been used to evaluate the performance of the paging allocation strategy for 5D and its performance has been compared to that of the paging allocation strategy for 2D, where the results show that the performance of paging for 5D is better than that of paging for 2D in terms of job turnaround time and system utilization.

4.1 Assumptions :

In this theses, the network topology has been adapted from 2D to 5d and the simulation experiments have been conducted to evaluate our adapted topology. The system parameters used are as follows:

Processor allocation strategy is paging.

First-Come-First-Served (FCFS) is used to schedule the system jobs for each network topology.

Jobs are generated based on uniform distribution with near neighbour communication in the first evaluation for each network topology (2D, 5D).

Jobs are generated based on uniform decreasing distribution with near neighbour communication in the second evaluation for each topology (2D, 5D).

Jobs are generated based on uniform distribution with all-to-all communication in the third evaluation for each network topology (2D, 5D).

Jobs are generated based on uniform decreasing distribution with all-to-all communication in the fourth evaluation for each network topology (2D, 5D).

System size is 59049 processors and this number is represented as $(243 \times 234 = 59049)$ in 2D and $(9 \times 9 \times 9 \times 9 \times 9 = 59049)$ in 5D.

4.2 Simulator :

ProcSimity simulation tool [21, 24] is a flexible research tool that has been used to evaluate processor allocation and job scheduling, and it was developed in University of Oregon [21, 24]. This tool is developed using C programming language and it is an open source software. ProcSimity can provide environmet that allow reserchers to test and evaluate any new processor allocation or job scheduling algorithm with clear results [21, 24]. ProcSimity has been used in many researchs [8, 13, 14, 16, 17, 20, 21, 26, 27, 28, 29, 31, 32, 33, 34, 41, 42], and also there are different versions from ProcSimity which include more algorithms and more features that supports differents topologies [36].

4.3 Why Simulation?

Our system is very huge system, and the cost of real machine is very high. Since there are two ways to evaluate system performance (analytical and simulation) [25], simulation was chosen because it is simple, cheap and easy to be used to integrate any new algorithm within it with minimal cost. Also, the simulation can be used to reflect the real system behavior.

4.4 Simulation results :

Extensive simulation experiments have been conducted for various system loads to compare the performance of the suggested paging allocation strategy for $5D$ with that of the paging allocation strategy for $2D$ for the same system size (i.e., the same number of processors for each tested systems). In this thesis, the paging allocation and deallocation algorithms have been implemented in C language and integrated into ProcSimity simulation tool [21, 24]. Our proposed $5D$ system has been used to evaluate the paging algorithm, where the system size is 59049 processors that are represented as $9 \times 9 \times 9 \times 9 \times 9$ $5D$. Also, in this thesis, we adapted the paging algorithm to be applicable for our proposed system and compared its performance with that of the paging algorithm for $2D$ system, where the $2D$ system has the same number of processors that are represented as 243×243 $2D$ system. Jobs inter-arrival times are generated exponentially. Jobs are scheduled on First-Come-First-Served (*FCFS*) basis to achieve fairness [13, 19, 20, 45]. Its measured by floating point values, but not normal time units [24]. Job size is generated based on two forms of distribution, the first is uniform distribution, where job size is ranged from 1 to 59049 and the second distribution is uniform-decreasing, where job sizes in this distribution are controlled by four ranges r_1 , r_2 , r_3 , and r_4 with four probabilities p_1 , p_2 , p_3 , and p_4 and three integer values l_1 , l_2 , and l_3 , where $r_1 = [1: l_1]$, $r_2 = [l_1+1: l_2]$, $r_3 = [l_2+1: l_3]$ and $r_4 = [l_3+1: L]$, and L is 59049 and also $p_1=0.4$, $p_2=0.2$, $p_3=0.2$, and $p_4=0.2$, and $l_1 = 7381$, $l_2 = 14762$, and $l_3 = 29524$. In uniform-decreasing distribution, most of jobs are small relative to the system size, and this distribution has been used in the previous studies [6, 7, 10, 11, 13, 19, 20, 33, 46, 48].

4.4.1 Simulation parameters :

Simulation parameters are clarified in table 4.1

Table 4.1: System Parameters that are used in simulation process.

<i>Simulator Parameter</i>	<i>2D</i>	<i>5D</i>
<i>System Size</i>	59049	59049
<i>Mesh Dimensions</i>	243x243	9x9x9x9x9
<i>Architecture</i>		
<i>Allocation Strategy</i>	Paging	Paging
<i>Page Size</i>	0	0
<i>Scheduling Strategy</i>	First-Come-First-Served (FCFS)	First-Come-First-Served (FCFS)
<i>Communication pattern</i>	All To All, Near Nighbour	All To All, Near Nighbour
<i>Job Size Distribution</i>	Uniform, Uniform-Decreasing	Uniform, Uniform-Decreasing
<i>Inter-arrival Time</i>	Exponential with different values for the mean, where values selected by experiments.	Exponential with different values for the mean, where values selected by experiments.
<i>Number of Runs</i>	Number of runs is not fixed but it is determined with the relative errors do not exceed 5%.	Number of runs is not fixed but it is determined with the relative errors do not exceed 5%.
<i>Number of Jobs per Run</i>	1000	1000

Simulation experiments have been conducted with different runs ranged from 50-100 runs and 1000 jobs have been generated in each run. This is to make sure that confidence level is over 95% and relative error does not exceed 5%.

4.4.2 Turnaround time :

Turnaround time is the time that the job stays in the system since it enters until it leaves. Figures 14, 15, 16 and 17 show turnaround time against the system load. In figure 14, 15, 16 and 17 the performance of paging for 5D is superior over that of paging for 2D when near neighbour communication pattern is used as shown in figures 14 and 15 and when all to all communication pattern is used as shown in figures 16 and 17. For example, in figures 14, 15, 16 and 17 when the system load is very low, The performance of paging for 5D in terms of turnaround time is 99% of that of paging for 2D, and this is due to the number of the processes that is currently in the system is small and hence the performance for both of the networks is almost the same, but in figure 14 when the system load is 2%, The performance of paging for 5D in terms of turnaround time is 9% of that of paging for 2D, and also in figure 15 when the system load reaches 5%, The performance of paging for 5D in terms of turnaround time is 4.8% of that of paging for 2D. In figure 16, we change the communication pattern to be all to all and 5D is still superior over that of paging for 2D , for example in figure 16, when the load is 2%, The performance of paging for 5D in terms of turnaround time is 64% of that of paging for 2D and in figure 17, when the system load reaches 2%, The performance of paging for 5D in terms of turnaround time is 29% of that of paging for 2D. Obviously, when the system load is increased, the performance of paging for 5D is better than that of paging for 2D. The superiority of the paging allocation strategy on 5D over that on 2D is due to the success of the allocation for the scheduled jobs whenever the load is increased. This increase in the number of allocated processes generates more messages between processors and the available messages paths in 2D is limited which cause congestion in 2D . This congestion can be alleviated when the network topology is 5D since there are several alternative paths for messages. Moreover, the distance between the farthest two processors that are allocated to the same process in 2D is longer than 5D, and hence, the possibility of messages interference from other processes in 2D is high and this deteriorates the performance in terms of turnaround time.

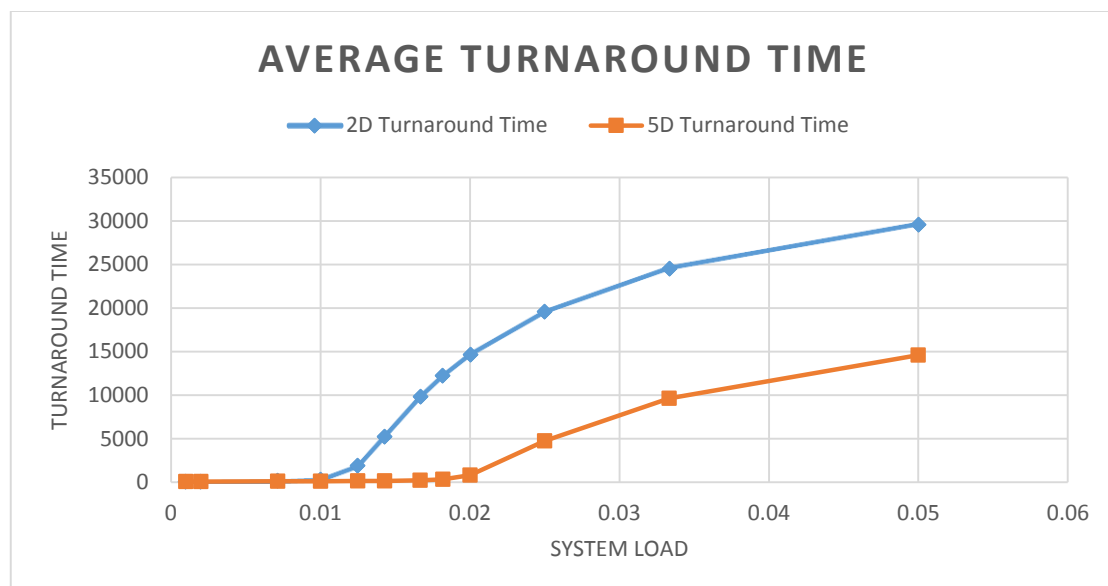


Figure (14) turnaround time vs. system load using uniform distribution and near neighbour communication with 59049 system size.

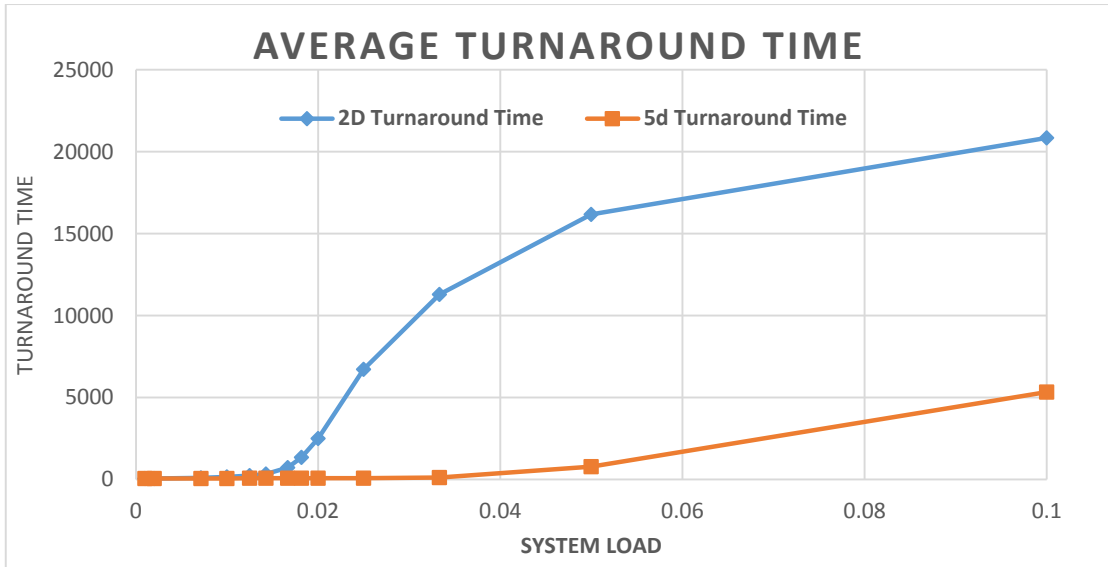


Figure (15): turnaround time vs. system load using uniform decreasing distribution and near neighbour communication with 59049 system size.

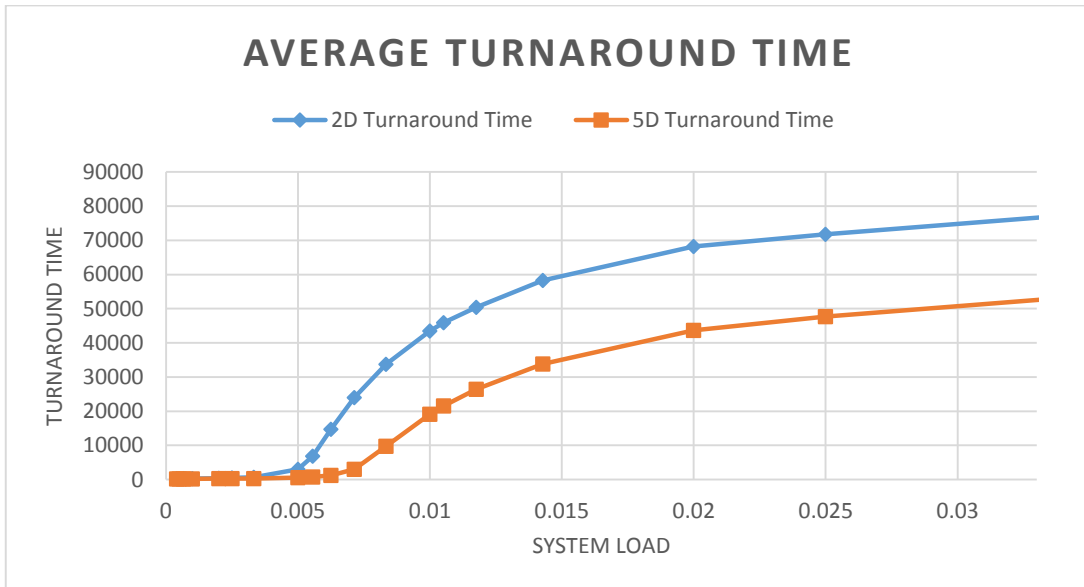


Figure (16): turnaround time vs. system load using uniform distribution and ALL TO ALL communication with 59049 system size.

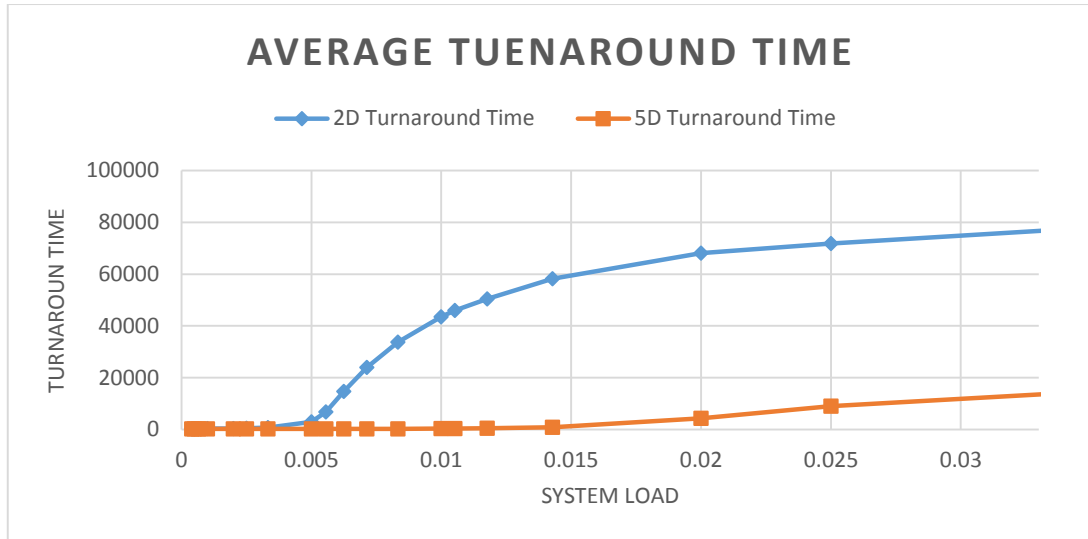


Figure (17): turnaround time vs. system load using uniform decreasing distribution and ALL TO ALL communication with 59049 system size.

4.4.3 Performance Impact of Mesh System Size :

Simulation experiments has been carried out for paging allocation strategy on small system that consists of 1024 processors only , where in case of 2D mesh system $k = 32$ and $n = 2$ represented as $32 \times 32 = 1024$ and in other hand, in 5D mesh system $k = 4$ and $n = 5$ represented as $4 \times 4 \times 4 \times 4 \times 4 = 1024$. In this simulation experiments, job sizes are generated using uniform-decreasing distribution and communication pattern is near neighbour. Simulation results show that paging allocation strategy on 5D system still has the superiority over that of paging allocation strategy on 2D system as shown in figure 18. For example, when system load reaches 5%, the paging allocation strategy on 5D needs only 65% from that paging allocation strategy in 2D. This superiority is due to existence of alternative paths for congestion in 5D, shorter links in 5D and lower diameter in 5D, so the possibility of message passing interference become lower in 5D which decreases the jobs' turnaround time in 5D.

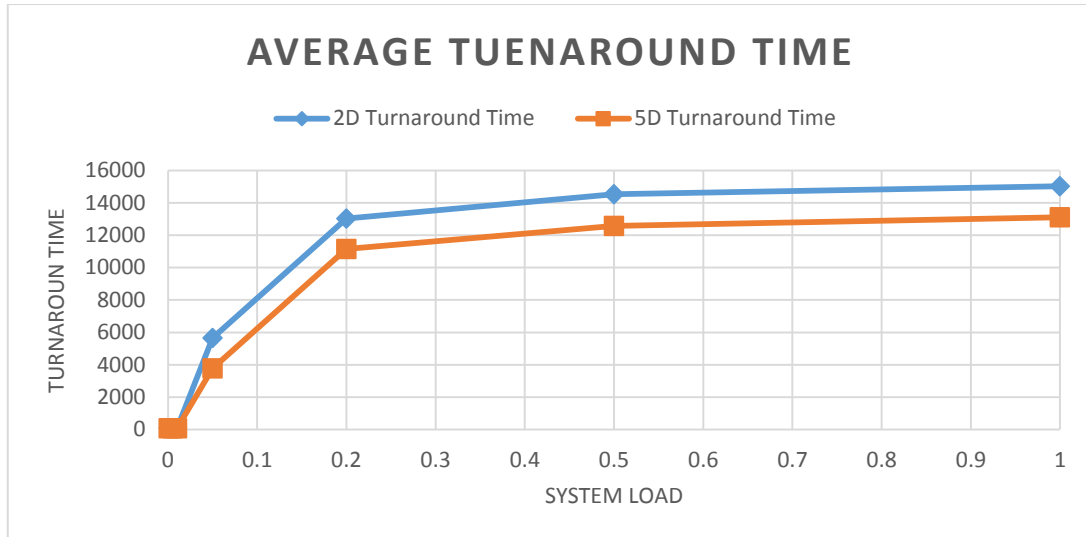


Figure (18): turnaround time vs. system load using uniform decreasing distribution and near neighbour communication with 1024 system size.

4.4.4 Utilization :

Utilization is the percentage of processors that are utilized over time. High utilization is an indicator for successful allocation and deallocation. Figures 19, 20, 21 and 22 show the system utilization of paging allocation strategy. In figures 19 and 20, the uniform and uniform decreasing distributions with near neighbor communication pattern have been used, and in figures 21 and 22, uniform and uniform decreasing distributions with ALL to ALL communication pattern have been used. The importance of utilization appears when the load is increased. 5D interconnection network shows better system utilization than 2D. In figure 19, for example, the maximum 2D system utilization is 70% and it is 72% in figure 20. Figures 19 and 20, show that the maximum 5D system utilization are 75% and 78% respectively. Figure 21 shows that the maximum 2D system utilization is 68% and it is 66% in figure 22. Figures 21 and 22, show that the maximum 5D system utilization percentages are 69% and 67% respectively. Better utilization achieved by successful allocation. The existence of alternative paths for messages in 5D solved the congestion problem which leads to successful allocation. Moreover, the distance between the farthest two processors that are allocated to the same process in 5D is shorter than 2D. Therefore, the possibility of message passing interference from other processes in 5D is low which leads to a higher utilization.

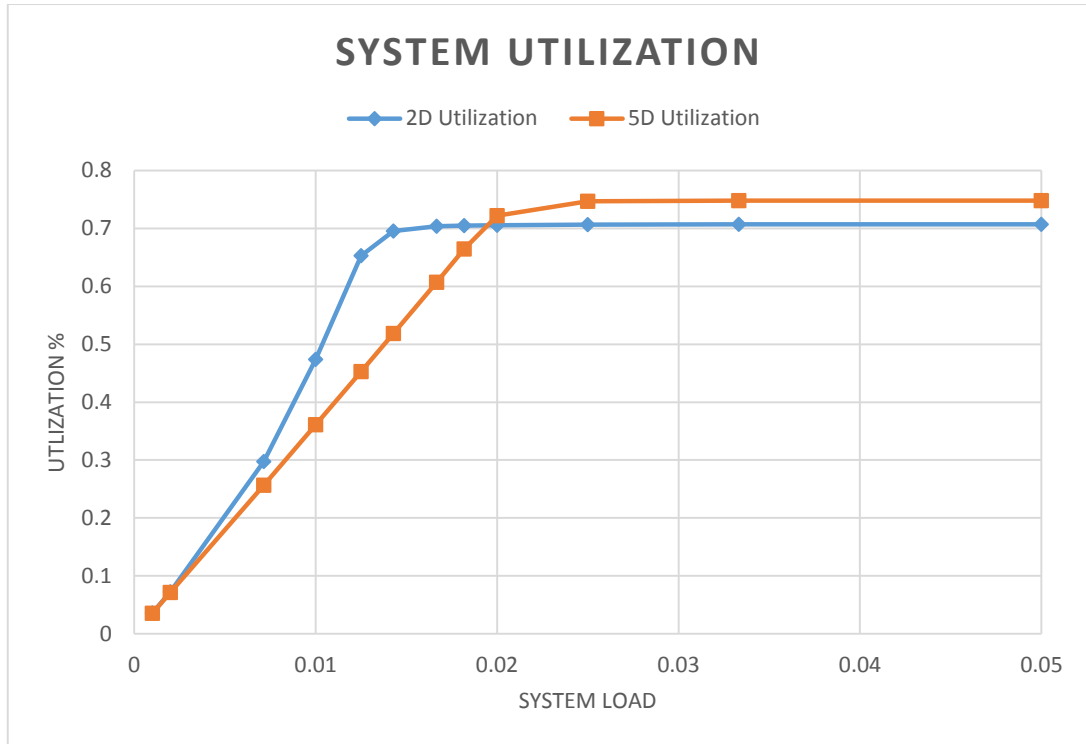


Figure (19): system utilization vs. system load using uniform distribution and near neighbour communication with 59049 system size.

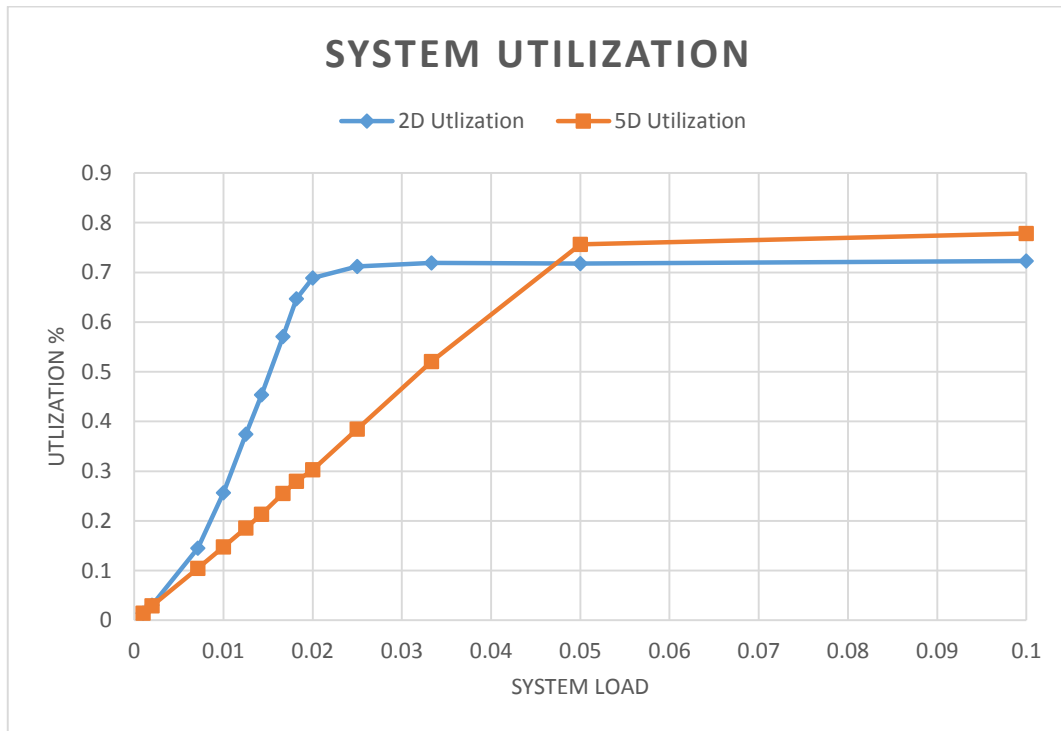


Figure (20): turnaround time vs. system load using uniform decreasing distribution and near neighbour communication with 59049 system size.

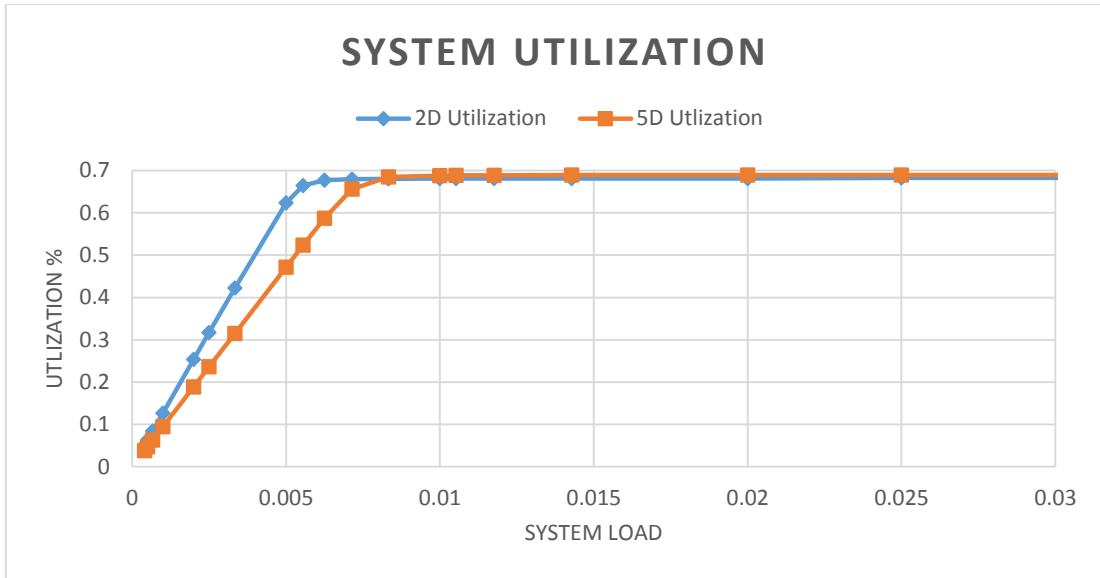


Figure (21): system utilization vs. system load using uniform distribution and ALL To ALL communication with 59049 system size.

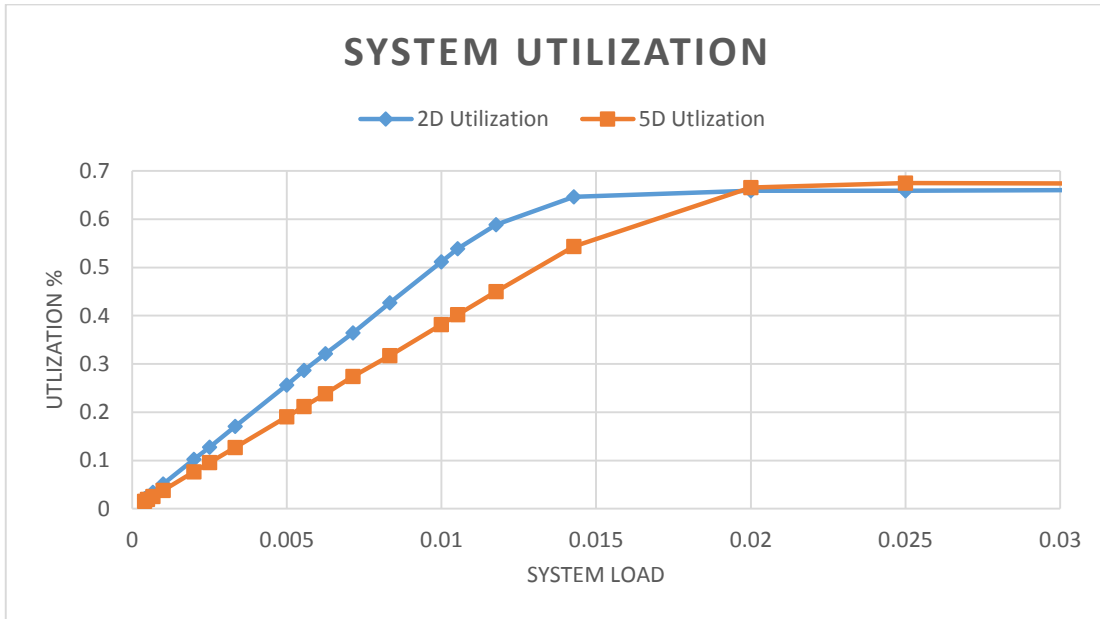


Figure (22): system utilization vs. system load using uniform decreasing distribution and ALL To ALL communication with 59049 system size.

4.5 Conclusion :

The paging allocation strategy on 2D mesh performs successful allocations and deallocations, but with high average turnaround times and low system utilization. In this thesis, we adapt the paging allocation strategy to be applicable on 5D interconnection network and this results in an efficient allocation and hence improves system performance in terms of both average turnaround time and system utilization. The performance of the paging allocation strategy on 2D has been compared with that of the paging allocation strategy on 5D interconnection network. The simulation results show that when the system size and system load is increased, then the 5D interconnection network will decrease the average turnaround time and increase the system utilization. This improvement in performance for the 5D interconnection network is due to the lower diameter, and hence less communication distance and also is for the existence of alternative paths for congestion paths.

Chapter 5: Conclusion and Future Directions

5.1 Summary of the results :

In this thesis, we focused on the developing of the non-contiguous allocation strategy (paging allocation strategy) on 5D interconnection network. The main results are listed below:

The results of the previous research suggested that interconnection networks upgraded for mesh-connected multicomputers are needed. The high average turnaround time and low system utilization of non-contiguous allocation strategies on the existing interconnection networks gives us the motivation to upgrade interconnection networks to a higher dimensional interconnection networks such as the 5D interconnection network.

To compare the performance of the paging non-contiguous allocation strategy on 2D interconnection networks with that of the adapted paging on 5D interconnection networks, extensive simulation experiments under a variety of system loads have been carried out. Our simulation results have exposed that the paging non-contiguous allocation strategy on 5D interconnection network has better performance than that on 2D interconnection network.

When comparing paging non-contiguous allocation strategy on 2D interconnection network with the adapted paging on 5D interconnection network, the results show that for the high system loads, the average turnaround time of jobs for the adapted paging on 5D is only 5% of that of the paging on 2D. Moreover, the system utilization for the adapted paging on 5D reaches 78% while the system utilization for the paging on 2D reaches 72%.

5.2 Future work directions :

Many open problems can be investigated, which are interesting for researchers. The following are some open problems that are related to our work and can be considered for further investigation in future.

In our experiments, we use the First-Come-First-Served (*FCFS*) scheduling but that would be interesting to use another scheduling methods such as Short-Job-First-Served (SJFS)[23] and Out-Of-Order (OO)[15].

Our research has been focused on paging allocation strategy, there are many other strategies that are used widely for non-contiguous allocation, implementing one or more of them on 5D interconnection network would be interesting.

We chose 5D as a higher dimensional network. What about using higher dimensional interconnection network such as 6D? That also would be interesting and more challenging.

References

- [1] Abbas Sheibanyrad, Frdric Ptrot, and Axel Jantsch, 3D Integration for Noc-Based SoC Architectures, ISBN: 9781441976185, 2010.
- [2] A. I. D. Bucur and D. H. J. Epema, Scheduling Policies for Processor Co Allocation in Multicenter Systems, IEEE Transaction on Parallel and Distributed Systems, vol. 18, no. 7, pp. 958-972, 2007.
- [3] A. Louri and H. Sung, "An optical multi-mesh hypercube: a scalable optical interconnection network for massively parallel computing," in Journal of Lightwave Technology, vol.12, no.4, pp.704-716, 1994.
- [4] Bani Mohammad, Saad, Efficient processor allocation strategies for mesh-connected multicenter. PhD thesis, University of Glasgow, 2008.
- [5] Byung S. Yoo and Chita R. Das. 2002. A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicenter. IEEE Transactions on Computers vol.51, no.1, pp 46-60, 2002.
- [6] B.-S. Yoo and C.-R. Das, A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicenter, IEEE Transactions on Parallel & Distributed Systems , vol. 51, no. 1, pp. 46-60, 2002.
- [7] C. A. F. De Rose, H.-U. Heiss, and B. Linnert, Distributed Dynamic processor Allocation for Multicenter, Parallel Computing, vol. 33, no. 3, pp. 145-158, 2007.
- [8] D. P. Bunde, V. J. Leung and J. Mache, Communication Patterns and Allocation Strategies, Sandia Technical Report SAND2003-4522, Jan. 2004.
- [9] D.M. Nicol, W. Mao, On bottleneck partitioning of k-ary n-cubes, Parallel Process. vol. 6, no.3, pp.389-399, 1996.
- [10] D. Babbar and P. Krueger, A performance Comparison of Processor Allocation and Job Scheduling Algorithms for Mesh-Connected Multiprocessors, Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing, pp. 46-53, 1994.
- [11] F. Wu, C.-C. Hsu and L.-P. Chou, Processor Allocation in the Mesh Multiprocessors Using the Leapfrog Method, IEEE Transactions on Parallel and Distributed Systems , vol. 14, no. 3, pp. 276-289, 2003.
- [12] Ismail Ababneh. 2006. An efficient free-list sub mesh allocation scheme for two-dimensional mesh-connected multicenter. Journal of Systems and Software. vol.79, no.8, pp.1168-1179, 2006.
- [13] I. Ababneh, An efficient free-list submesh Allocation Scheme for two-dimensional mesh-connected multicenter, Journal of Systems and Software, vol. 79, no. 8, pp. 1168-1179, August 2006.
- [14] I. Ababneh and F. Fraij, Folding contiguous and non-contiguous space sharing policies for parallel computers, Mu'tah Lil-Buhuth wad-Dirasat, Natural and Applied Sciences Series, vol. 16, no. 3, pp. 9-34, 2001.
- [15] I. Ababneh, Job scheduling and contiguous processor allocation for three-dimensional mesh multicenter, AMSE Advances in Modelling & Analysis , vol. 6, no. 4, pp. 43-58, 2001.
- [16] J. Mache, V. Lo, and K. Windisch, Minimizing Message-Passing Contention in Fragmentation-Free Processor Allocation, Proceedings of the 10th International Conference on Parallel and Distributed Computing Systems, pp. 120-124, 1997.

- [17] J. Mache, V. Lo, and S. Garg, Job Scheduling that Minimizes Network Contention due to both Communication and I/O, Proceedings of the 14th International Parallel and Distributed Processing Symposium(IPDPS'00), pp. 457-463, 2000.
- [18] Jun Zhang, Parallel and Distributed Computing, University of Kentucky Lexington from URL: <https://www.cs.uky.edu/~jzhang/CS621/chapter5.pdf>, 2017, December.
- [19] K.-H. Seo and S.-C. Kim, Improving system performance in contiguous processor allocation for mesh-connected parallel systems, The Journal of Systems and Software, vol. 67, no. 1, pp. 45-54, 2003.
- [20] K.-H. Seo, Fragmentation-Efficient Node Allocation Algorithm in 2D Mesh-Connected Systems, Proceedings of the 8th International Symposium on Parallel Architecture, Algorithms and Networks (ISPAN'05), IEEE Computer Society Press, pp. 318-323, 2005.
- [21] K. Windisch, J. V. Miller, and V. Lo, ProcSimity: an experimental tool for processor allocation and scheduling in highly parallel systems, Proceedings of the 5th Symposium on the Frontiers of Massively Parallel Computation (Frontiers'95), Washington, DC, USA, IEEE Computer Society Press, pp. 414-421, 1995.
- [22] Kyung-Hee Seo and Sung-Chun Kim. 2003. Improving system performance in contiguous processor allocation for mesh-connected parallel systems. Journal of Systems and Software vol.67, no.1, pp. 45-54 2003.
- [23] N. Xoxa, M. Zotaj, I. Tafa, Julian Fejzaj, Simulation of First Come First Served (FCFS) and Shortest Job First (SJF) Algorithms, International Journal of Computer Science and Network, vol.3, no.1, pp 444-449, 2014.
- [24] ProcSimity V4.3 User's Manual, University of Oregon, 1997.
- [25] R. Jan, The Art of Computer Systems Performance Analysis, John Wiley & Sons, Inc., New York, 1991.
- [26] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and Lewis M. Mackenzie, An Efficient Turning Busy List Sub-mesh Allocation Strategy for 3D Mesh Connected Multicomputers, Proceedings of the 7th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking & Broadcasting, (PGNET 2006) , Liverpool John Moores University, UK, pp. 37-43, 26-27 June 2006.
- [27] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, A Fast and Efficient Processor Allocation Strategy which Combines a Contiguous and Non-contiguous Processor Allocation Algorithms, Technical Report; TR-2007-229, DCS Technical Report Series, Department of Computing Science, University of Glasgow, January 2007.
- [28] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, A Performance Comparison of the Contiguous Allocation Strategies in 3D Mesh Connected Multicomputers, Proceedings of The 5th International Symposium on Parallel and Distributed Processing and Applications (ISPA'07), LNCS 4742, Springer-Verlag Berlin Heidelberg, pp. 645-656, 2007.

- [29] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, Comparative Evaluation of the Non-Contiguous Processor Allocation Strategies based on a Real Workload and a Stochastic Workload on Multicomputers, Proceedings of the 13th International Conference on Parallel and Distributed Systems (ICPADS'07) , vol. 2, pp. 1-7, IEEE, Hsinchu, Taiwan, 2007.
- [30] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and Lewis M. Mackhenzie, An Efficient Processor Allocation Strategy that Maintains a High Degree of Contiguity among Processors in 2D Mesh Connected Multicomputers, 2007 ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2007) , IEEE Computer Society Press, Philadelphia University, Amman, Jordan, pp. 934-941, 2007.
- [31] S. Bani-Mohammad, M. Ould-Khaoua, and I. Ababneh, A New Processor Allocation Strategy with a High Degree of Contiguity in Mesh-Connected Multicomputers, Journal of Simulation Modelling, Practice & Theory, vol. 15, no. 4, pp. 465-480, 2007.
- [32] S. Bani-Mohammad, M. Ould-Khaoua, and I. Ababneh, An Efficient Non-Contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers, Journal of Information Sciences , vol. 177, no. 14, pp. 2867-2883, 2007.
- [33] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, Non-contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers Based on Sub-meshes Available for Allocation, Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06) , Minneapolis, Minnesota, USA, IEEE Computer Society Press, vol. 2 , pp. 41-48, 2006.
- [34] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, A Fast and Efficient Strategy for Sub-mesh Allocation with Minimal Allocation Overhead in 3D Mesh Connected Multicomputers, Ubiquitous Computing and Communication Journal, vol. 1, no. 1, pp. 26-36, ISSN 1992-8424, 2006.
- [35] S. B. Mohammad, M. O. Khaoua, L. M. Mackenzie, and I. Ababneh. Processor allocation and job scheduling on 3D mesh interconnection networks International Journal of Computer Applications Vol.29, No.3, pp. 309-317, 2007.
- [36] S. Bani-Mohammad, Efficient Processor Allocation Strategies for Mesh-Connected Multicomputers, Ph.D. Thesis, The Faculty of Information and Mathematical Sciences, University of Glasgow, 2008.
- [37] S. Bani-Mohammad, I. Ababneh, On the performance of non-contiguous allocation for common communication patterns in 2D mesh-connected multicomputers, Simulation Modelling Practice and Theory, Vo. 32, pp. 155-165, 2013.
- [38] Stephen W. Keckler, Kunle Olukotun H. Peter Hofstee, Multicore Processors and Systems, ISBN: 9781441902634, 2009.
- [39] Turing, IBM Blue Gene/Q. Retrieved from URL: <http://www.idris.fr/eng/turing/hw-turing-eng.html>, 2016, April.
- [40] Varavithya, V., Multicasting in wormhole routed multicomputers. Ph.D. Thesis. Department of Electrical and Computer Engineering, Iowa State University, 1998.

- [41] V. Lo and J. Mache, Job Scheduling for Prime Time vs. Non-prime Time, Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'02), pp. 488-493, 2002.
- [42] V. Lo, K. J. Windisch, Wanqian Liu and B. Nitzberg, "Noncontiguous processor allocation algorithms for mesh-connected multicomputers," in IEEE Transactions on Parallel and Distributed Systems, vol. 8, no. 7, pp. 712-726, 1997.
- [43] William J. Dally. 1990. Performance Analysis of k-ary n-cube Interconnection Networks. IEEE Transactions on Computers . vol.39, no.6 pp. 775-785, 1990.
- [44] W.J. Dally, Performance analysis of k-ary n-cube interconnection networks, IEEE Trans. Comput. vol.39 pp.775–785, 1990.
- [45] 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 (HPCAS IA'05), IEEE Computer Society Press, pp. 53-60, 2005.
- [46] W. Qiao and L. Ni, Efficient processor allocation for 3D tori, Proceedings of the 9th International Conference on Parallel Processing Symposium , IEEE Computer Society Press, pp. 466-471, 1995.
- [47] Y. Ashir, I. Stewart, A. Ahmed, Communication algorithms in k-ary n-cube nterconnection networks, Inform. Process. vol. 61, pp.43–48, 1997.
- [48] Y. Zhu, Efficient processor allocation strategies for mesh-connected parallel computers, Journal of Parallel and Distributed Computing, vol. 16, no. 4, pp. 328-337, 1992.
- [49] https://en.wikipedia.org/wiki/IBM_Mira, last seen 30/4/2018.
- [50] [https://en.wikipedia.org/wiki/Fermi_\(supercomputer\)](https://en.wikipedia.org/wiki/Fermi_(supercomputer)), last seen 30/4/2018.
- [51] د. اسماعيل عبابنة و د.سعد بني محمد "التخصيص غير المتجاور للمعالجات في متعددات الحواسيب الشبكية ثلاثية الأبعاد" المنارة، العدد ٢ المجلد ١٢، تموز ٢٠٠٣

الملخص باللغة العربية

أنظمة الحواسيب ذات المعالج الأحادي قد وصلت لمرحلة أصبح فيها التطوير قليل جدا، لكن البرمجيات الحديثة والتي يتم تصنيعها هذه الايام بحاجة الى أنظمة حواسيب متعددة المعالجات فائقة السرعة لتسطيع حل المشاكل الكبيرة والمعقدة، لتحقيق ذلك نحتاج لإليات تخصيص معالجات فعالة لتتمكن من تخصيص المهمات الخاصة بهذه الأنظمة، حيث ان أليات تخصيص المعالجات تتأثر بالشبكات التي يتم تشكيل المعالجات بها من ناحية الاداء عن طريق وقت مكوث المهمة في النظام ونسبة استغلال النظام. في هذه الرسالة نحاول ان نحسن وقت المكوث الخاص بالمهمة في النظام وايضا نسبة استغلال النظام عن طريق استخدام شبكات ذات أبعاد أعلى والتي بدورها من الممكن ان تحسن أداء النظام دون تكاليف كبيرة اضافية. الشبكات ذات الأبعاد العالية لديها العديد من الوصلات بين المعالجات الخاصة بالنظام، وهذا بحد ذاته كفيل بتقليل تراحم الرسائل المرسله بين المعالجات وايضا يزيد من درجة التجاور بين المعالجات، علاوة على ذلك فقد تم زيادة عدد المهمات التي تم تخصيصها بنجاح في النظام والذي بدوره أدى الى تحسين أداء النظام من حيث وقت مكوث المهمة فيه وايضا نسبة استغلال النظام. أظهرت تجارب المحاكاة تحسن ملحوظ حين تم زيادة أبعاد الشبكة من بعدين الى خمسة أبعاد دون الحاجة لزيادة عدد المعالجات في النظام.