Al al-Bayt University
Computer Science Department

Prince Hussein Bin Abdullah College for Information Technology

Job Migration for 2D Mesh Multicomputers using Dynamic Compaction

ترحيل الأعمال في الحواسيب الشبكية ثنائية الأبعاد باستخدام التحشير على الأطراف

By
Amer Mamdooh Mtaw'a Al-mohaisen
1120901004

Supervisor:
Dr. Saad Bani-Mohammad

Co-supervisor:
Prof. Ismail ababneh

A Thesis Submitted to the Deanship of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Computer Science
Al al-Bayt University
Mafraq, Jordan

January 2016

Table of Contents

List of Figures

List of Tables

Abstract

The use of an efficient processor allocation algorithm is necessary to utilize the computational power of multicomputer systems. Processor allocation selects a set of processors to execute parallel jobs, processor allocation strategies can be divided into two main categories: contiguous and noncontiguous.

In contiguous allocation, the selected set of processors must be joint. This strategy could lead to high processor fragmentation which degrades system performance in terms of, for example, the turnaround time and system utilization.

Processor fragmentation can be of two types: internal and external. Many strategies have been devoted to reducing fragmentation; most of these strategies use dynamic allocation to solve this problem, for example migration, we will show the problems with the existing migration strategies such as non beneficial migration where the strategy migrate jobs but the freed sub-mesh is not suitable in size or shape for allocation.

A new migratory scheme is proposed in this thesis to solve the fragmentation problem in the allocation strategies and as improvement to four previous schemes that uses dynamic contiguous allocation, which are Online Dynamic Compaction-Single Corner (ODC-SC), Online Dynamic Compaction-Four Corners(ODC-FC), Conditional Online Dynamic Compaction-Four Corners (CODC-FC), and Conditional Online Dynamic Compaction-Four Corners Quarter By Quarter (CODC-FCQQ). This proposed scheme concentrates on expanding the search grid so that it contains the full mesh which increases the number of migrations. Also, migration is carried out only when it results in a successful allocation. In addition, request orientation is switched if allocation fails for the original allocation request, so as to increase the probability of successful allocation.

The performance analysis in this thesis indicates that the new proposed algorithm improves the performance in terms of average turnaround time substantially. Moreover, the system utilization is improved.

Chapter 1

Introduction

1.1 Introduction

A parallel computer is a set of processors that cooperate with each other to solve computational problems. There are two types of parallel computer systems–proposed by Michael Flynn, 1966-: systems that operate under a single control unit (they are referred to as single instruction stream, multiple data stream (SIMD) systems), and systems in which each processing element is capable of executing a different program independently of the other processing elements (they are referred to as multiple instruction stream, multiple data stream (MIMD) systems) [19, 25, 30].

Multicomputer systems are MIMD systems that are very popular substitutes for supercomputers (such as IBM BlueGene in 2008 and Cray Titan in 2012). They use multiple interconnected inexpensive computers to form a supercomputing environment (such as Solaris MC in 1995, and Intel Xeon Phi in 2010). A multicomputer system helps in solving large-scale computational problems, provides concurrency in solving multiple problems at the same time, and overcomes memory constraints.

There are many network topologies that are used in multicomputers. The nodes in these networks represent processors and the edges represent communication paths between the connected processors. Choosing between topologies is based on several criteria; an important criterion is the maximum value of the shortest path length between any two nodes, which is known as diameter. Another criterion is the number of links that connect a certain node to its neighbors, which is known as node degree [19, 22, 26, 33]. Examples of these networks are the hypercube that is shown in Figure 1 and the 2D mesh that is shown in Figure 2.

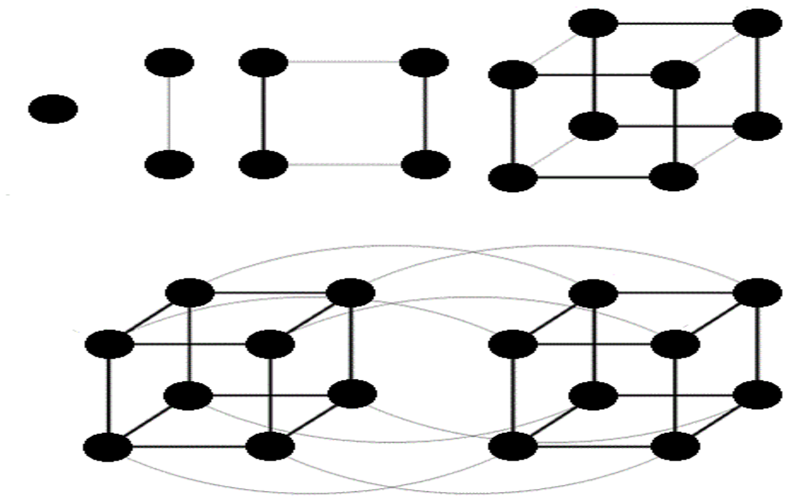

Figure 1: Hypercube topology

One of the advantages of hypercube topology is its small diameter, but the major disadvantage of this type of networks is its degree, since the degree depends on the number of processors in the system which is in hypercube a power of two. A hypercube has 2^d processors, where d is the hypercube's diameter and node degree.

The mesh topology is a common topology in multicomputer systems because of several good characteristics, such as simplicity, scalability, structural regularity, and ease of implementation [2, 9, 22]. So, the 2D mesh topology is the target system in this thesis. The following definitions are used in this thesis [9, 22].

Definition 1.1 A two-dimensional (2D) mesh, M(W,H), contains W × H processors (or nodes), where W is the mesh width and H is the mesh height. A processor is represented by a pair of coordinates (x, y), where $0 \leq x < W$ and $0 \leq y < H$.

Definition 1.2 A 2D mesh, M(W, H), can be partitioned into a number of sub-meshes. A sub-mesh S(w, h) contains w×h processors, where $0 < w \leq W$ and $0 < h \leq H$. It is represented by the coordinates (x1, y1, x2, y2), where (x1, y1) is the lower left corner of the sub-mesh (base node), and (x2, y2) is the upper right corner (end node).

Here, w = x2 – x1 +1 and h = y2 – y1 + 1.

Definition 1.3 A sub-mesh is said to be a free sub-mesh if and only if all its processors are unallocated to any job, and it is an allocated sub-mesh if and only if all its processors are allocated to the same job.

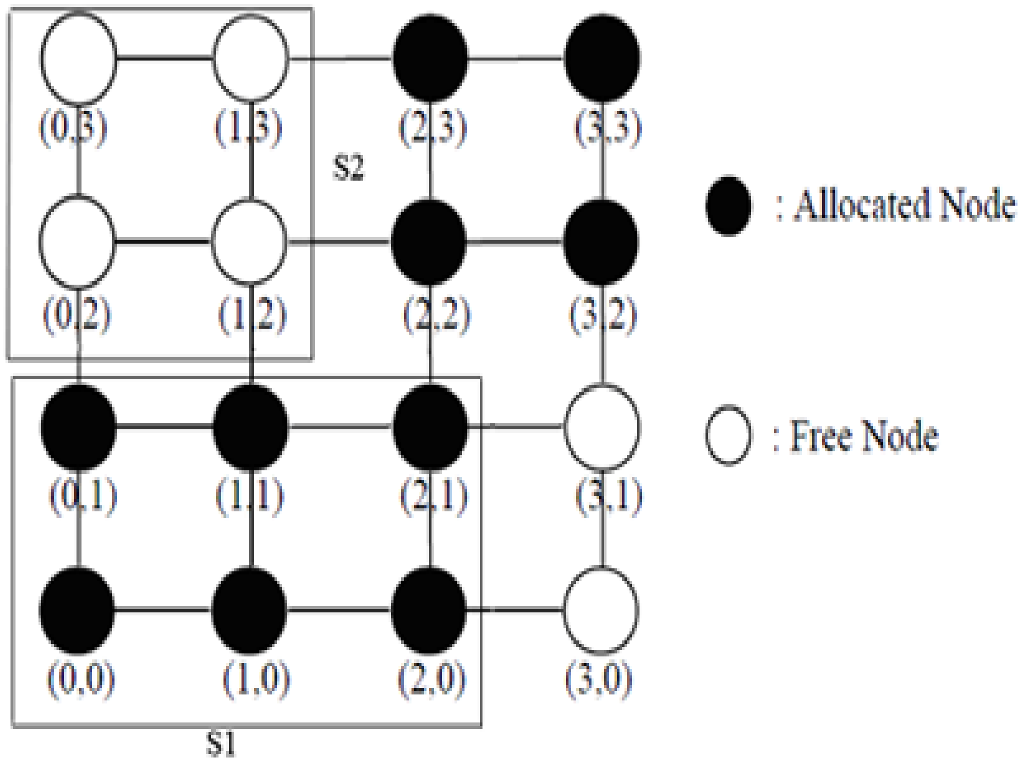

Figure 2: An example of a 4×4 2D mesh

The mesh in Figure 2 is a 4 × 4 2D mesh. In this figure, the black circles represent allocated processors, while the white circles represent free processors. For example, ((0, 0), (2, 1)) represents the 3 × 2 allocated sub-mesh S1, where (0, 0) represents the coordinates of the base node of S1 and (2, 1) represents the coordinates of the end node of S1. The sub-mesh ((0, 2), (1, 3)) is the 2 × 2 free sub-mesh S2 with the base node (0, 2) and the end node (1, 3).

1.2 Processor Allocation

Processor allocation in a mesh multicomputer is defined as the selection of the set of processors to execute a specific job [13, 14]. There are many processor allocation strategies that have been suggested for 2D mesh connected multicomputers. They can be classified as: contiguous and non-contiguous.

In contiguous allocation, a (contiguous) sub-mesh is reserved for a distinct job for the duration of its execution. The advantage expected from contiguous allocation is that there is no contention with the communication of other jobs in the interconnection network. On the other hand, there are some problems, such as the reduced chance of successful allocation although there is a sufficient number of free processors. This degrades system performance in terms of performance parameters such as job turnaround time (the time that the job spends in the system from from arrival to departure) and mean system utilization (percentage of processors allocated over time) [2, 4, 5, 16, 21, 25, 27, 28, 34, 36, 38, 39,40,41].

In non-contiguous allocation, a job can execute on multiple separate smaller sub-meshes rather than waiting until a single sub-mesh of the requested size and shape is available. The main disadvantage of non-contiguous allocation is the increase in message contention inside the network, however dropping the contiguity condition can reduce processor fragmentation and increase system utilization [2, 4, 5, 16, 21, 25, 27, 28, 34].

Processor fragmentation can be *internal* fragmentation, which occurs when the number of processors allocated to a job is more than the number the job needs. This means that some extra processors are allocated but they are not used. Processor fragmentation can be *external* fragmentation, which occurs when there are free processors enough in number for allocation, but they are not allocated to a job because they are not contiguous or they do not have the shape that is requested by the allocation request. Fragmentation is the main limitation for the performance of contiguous allocation schemes that needs to be minimized.

Contiguous allocation can be a good strategy despite of its drawbacks. This is because of its ability to minimize the distance of communication paths and avoiding interference among jobs. The drawbacks of contiguous allocation can be solved using dynamic allocation, where the sub-mesh of processors allocated to a job can be changed during job execution in order to achieve efficiency as opposed to static processor allocation, where a job is allocated a fixed sub-mesh of processors throughout its execution time [12]. To achieve dynamic allocation, we use job migration, which is defined as re-allocating jobs from their current position to another sub-mesh to produce large area of adjacent processors that can accommodate new incoming jobs [21], or the process of defragmentation which rearranges the nodes themselves.

1.3 Motivation and Contribution

In contiguous processor allocation, if allocation for a job request fails, the job will have to wait until a suitable free sub-mesh is available. Migration of executing jobs when allocation fails can improve the performance of multicomputer systems in terms of parameters such as system utilization, job turnaround time, and number of migrations. As a result of migration, allocation to one or more waiting jobs may become successful.

The previous migration strategies proposed in [21, 31] for 2D mesh multicomputers have several limitations such as: a large number of migrations is needed under heavy system loads and the sizes of free sub-meshes that result after migration are small as compared to the size of the mesh system, where allocating a large incoming jobs degrades system performance in terms of job turnaround time and system utilization as a result of the small freed sub-meshes. The study in [21] is an example of migration studies. The mesh system is subdivided into four quarters, and migration is attempted for the four quarters. The strategy migrates within the first quarter then it checks if allocation can be successful. If not then it migrates within the second quarter then checks for allocation. This process is repeated for the third and fourth quarters. Motivated by the above observations, we propose in this thesis a new migration process that differs from the earlier migration strategies in that it is based on considering migration in the system as a whole. This can be expected to give better performance as compared to the previous migration processes. The proposed migration process has a larger number of migrations than that in the CODC-FC strategy as an example, where in this strategy (which CODC-FCQQ is built upon) it uses four quarters and migrate jobs to the four corners at the same time, it results in more migrations with a bigger freed sub-mesh [21], and it can serve more jobs per migration cycle than that in [11], the proposed migration process migrate jobs to the bottom-left corner one at a time. Also, the proposed strategy modifies the allocation process by switching the orientation of allocation requests when allocation fails for the initial request orientation.

1.4 Outline of the Thesis

The rest of the thesis is organized as follows. Chapter two reviews some allocation and migration strategies that have been proposed for 2D mesh-connected multicomputer systems. Also, it presents the tool of the study (ProcSimity Simulator) used in this thesis.

Chapter three introduces a new migration process for 2D multicomputer system that is based on the first fit (FF) allocation strategy and presents the main features of this scheme. Moreover, it presents extensive simulation experiments that have been carried out to compare the performance of the proposed migration strategy against the existing migration strategies.

Chapter four gives a summary of the main results and some directions for future research.

Chapter Two

Background and Preliminaries

2-1 Related Work

In this section, a brief description of some allocation and migration strategies that have been proposed for 2D mesh network is presented [4, 16, 18, 21, 27, 28, 34, 39].

2.1.1 First Fit (FF) and Best Fit (BF) strategies.

These strategies depend on the idea of detecting the base node for a free sub-mesh that is large enough for the current allocation request, where a base node represents the lower-left corner of a free sub-mesh using two bit arrays (busy and coverage array). These arrays represent the status of the processors in terms of idle/busy processors, and the two strategies scan them from left to right to find base nodes of large enough free submeshes. The difference between BF and FF is that FF scans the two arrays for the first base node with enough free processors to allocate and use it, but BF uses the base node that has enough free processors and has the smallest area [9, 14, 39]. Figure 3 shows an illustration of the first fit and best fit strategies.

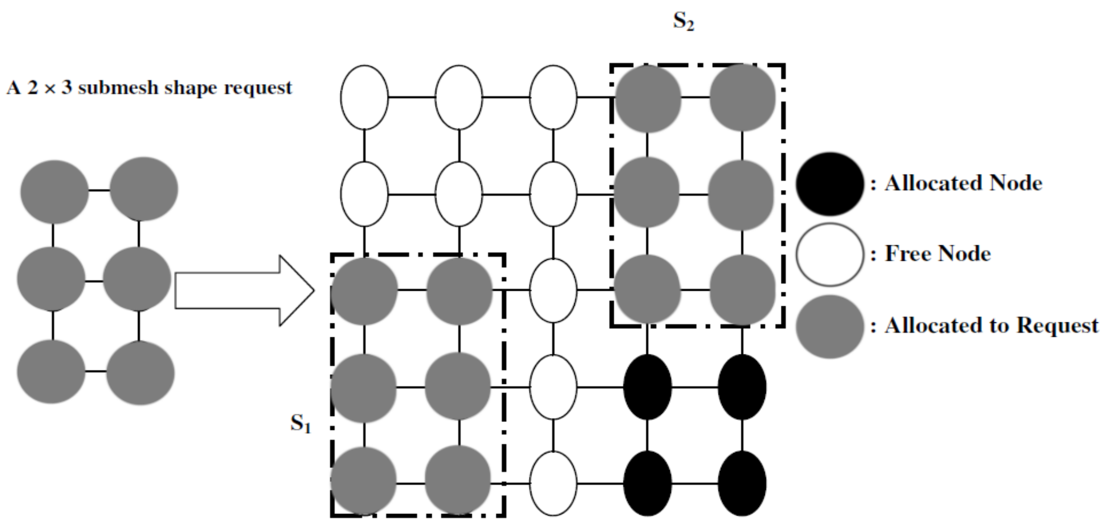

Figure 3: An allocation using First Fit and Best Fit strategies [14, 39]

2.1.2 Two-dimensional Buddy strategy (2D BUDDY).

It is a first-fit strategy but with the following conditions: the mesh must be a square with a power of two side lengths and allocation units are square sub-meshes. The 2D Buddy scheme suffers from severe internal fragmentation because a sub-mesh with dimensions that have 2^i side length is always required. Also, it has significant external fragmentation [9, 22, 27]. Figure 4 shows an allocation using the 2D Buddy strategy.

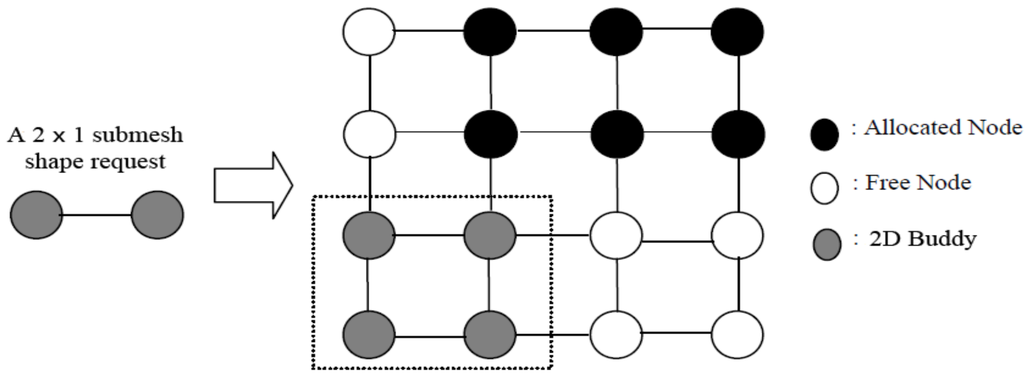

Figure 4: An allocation using the 2D Buddy strategy [27]

2.1.3 Frame sliding strategy (FS).

FS allocation strategy is a first-fit strategy. FS works with 2D meshes of all sizes and shapes. It allocates a free sub-mesh for the current allocation request by searching for a sub-mesh (frame) of processors large enough for the allocation request. The search method is done by searching horizontally from left to right and vertically from bottom to top and starts with the first free processor found at the bottom-left corner of the 2D mesh. The action for searching is done by sliding the frame horizontally and vertically by the width or the length of the allocation request [9, 16, 22]. This process is shown in Figure 5, where the first frame S1 is not as an appropriate frame for the allocation request because it is not free. The request is then moved horizontally by the width of the job request, which goes outside of the mesh. After that, the frame is moved vertically by the length of the job request, which also goes outside of the mesh. We notice that it ends without finding an appropriate frame although one exists. As shown by this example, FS strategy is not a complete recognition strategy.

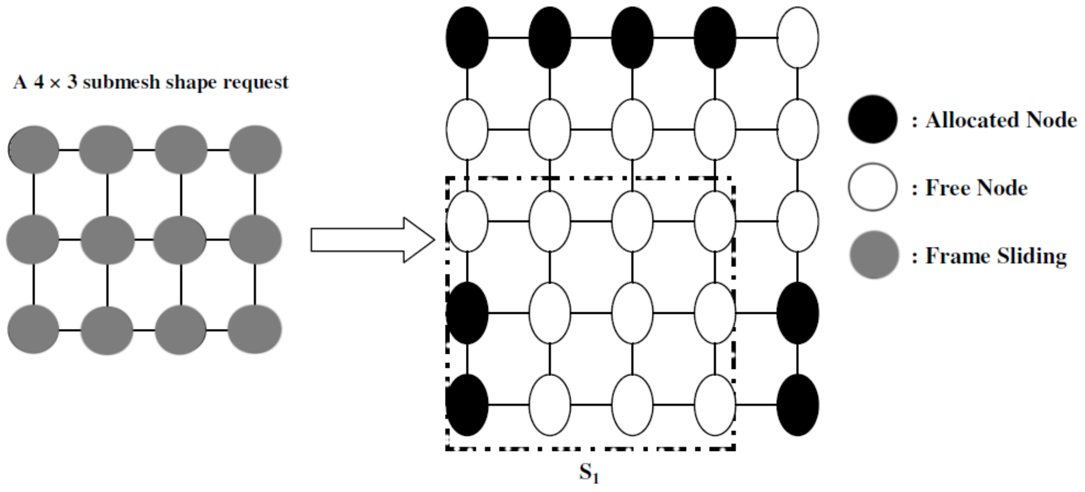

Figure 5: An allocation using the Frame Sliding strategy [16]

2.1.4 Adaptive Scan strategy (AS).

AS is a first fit allocation strategy. This strategy is an improvement of the FS strategy and it is considered as a first-fit recognition complete strategy [18]. Adaptive scan uses a free sub-mesh detection and scanning operation instead of sliding operation; where the algorithm scans the mesh by a frame -the same size of the incoming job- and if the frame is not free then the frame strides to the next point it moves in a single step vertically, and it moves horizontally based on the allocated sub-mesh. If the job cannot be accommodated in its original orientation, the orientation of the job is switched and allocation is re-attempted for the new request shape. In figure 6, the orientation of the job is switched when allocation of the original job orientation fails.

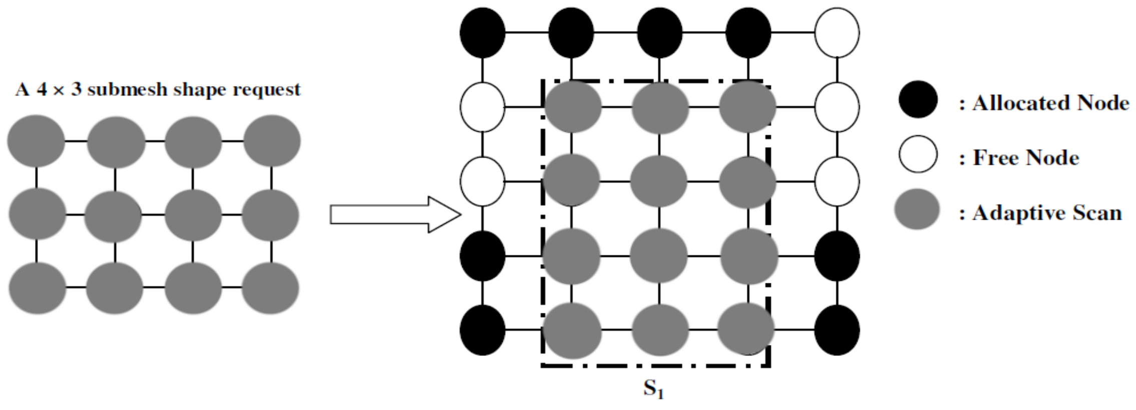

Figure 6: An allocation using the Adaptive Scan strategy [18]

2.1.5 First-Fit Mesh Bifurcation (FFMB) strategy.

The FFMB strategy tries to reduce the fragmentation problem that the first-fit allocation strategies suffer from by using different ways for detecting the base node of the allocated sub-mesh. When a job requests the allocation of a free sub-mesh, FFMB starts the search process from bottom-left corner or top-left corner of the mesh unlike first-fit strategies, which always start the search at the bottom-left corner. Choosing between bottom-left corner and top-left corner depends on the status of the upper-half and lower-half of the mesh by starting the search with the less busier half [21]. Figure 7 shows the main advantage of FFMB, where the search starts from the top-left corner of the mesh instead of the bottom of the mesh.

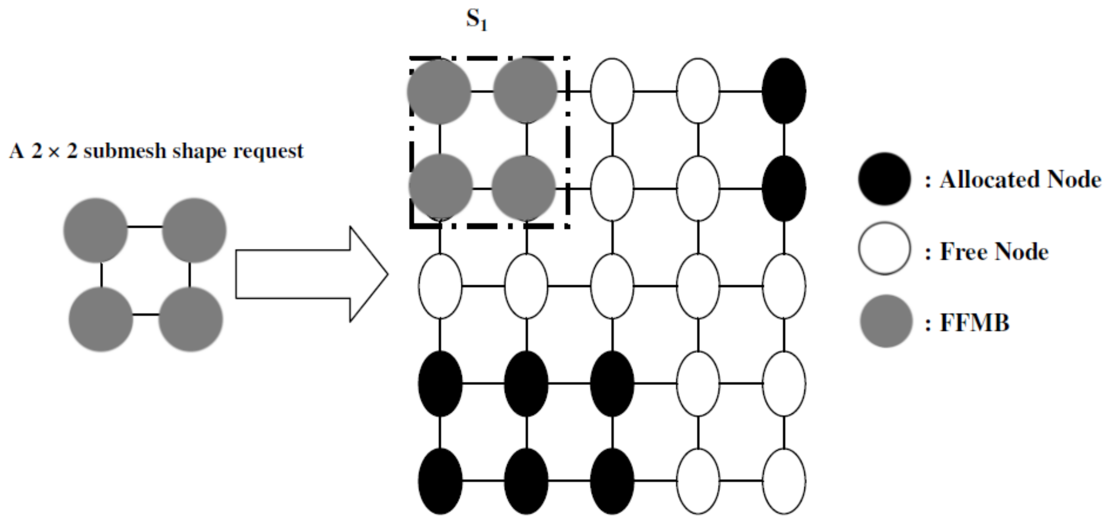

Figure 7: An allocation using First-Fit Mesh Bifurcation strategy [21]

2.1.6 Online Dynamic Compaction-Single Corner (ODC-SC) strategy.

The ODC-SC strategy was proposed to reduce the external fragmentation problem that contiguous processor allocation strategies suffer from. ODC-SC applies job migration on the running jobs to produce an area of adjacent free processors that is larger than that before migration in order to serve the current allocation request or future allocation requests. ODC-SC does not apply job migration unless the allocation strategy failed to serve the current allocation request in spite of the presence of a sufficient number of free processers because these processors are not contiguous. After job migration, the allocation strategy is used again to search for a free sub-mesh and allocates it to the current allocation request if it is found. In ODC-SC, all the running jobs are migrated to the bottom-left corner of the mesh, which aims to produce large areas of adjacent processors in right and top of the mesh. In job migration, the left or bottom edge of the sub-mesh are checked whether they are free or not. Then, shifting the sub-mesh leftwards or downwards until it reaches the left or bottom of the mesh boundary or the right edges or top edges of other allocated sub-meshes [21]. In figure 8, the ODC-SC strategy migrates jobs in case of allocation failure to the bottom left corner in order to free enough sub-mesh for allocation.

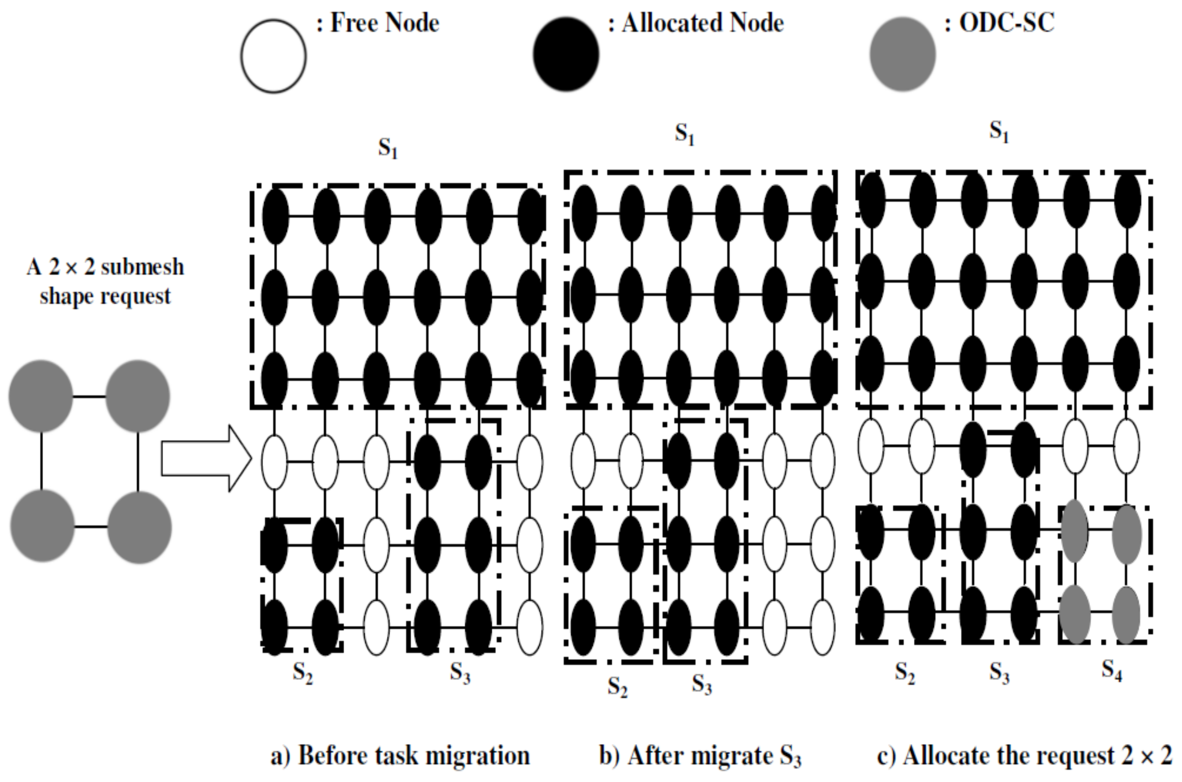

Figure 8: An allocation using Online Dynamic Compaction-Single Corner strategy [21]

- Drawbacks of ODC-SC

It has high number of migrations, and migration is done even if it is not beneficial in allocation.

2.1.7 Online Dynamic Compaction-Four Corners (ODC-FC) strategy.

In this strategy, the system is divided into four equally parts (four quarters) and processors are allocated to an allocation request by any processor allocation strategy. The main advantage for this strategy is that migration occurs in different directions in order to reduce the number of migrations; this is an advantage over the previous migration strategy which is ODC-SC that uses migration on the whole mesh to the left bottom corner which results in a large number of migrations. At the same time, ODC-FC migrates jobs without any restrictions, which means that the probability of unnecessary migrations (migration that does not result in successful allocation) is high.

If the number of requested processors is available and the allocation strategy fails to allocate a job request due to the lack of contiguity, then the job migration process is applied on the running jobs to migrate them into the four corners of the system according to the position of each running job, this migration leads to produce a large area of free adjacent processors in the middle of the system, then the allocation strategy is used again to allocate a free sub-mesh for the job request [21].

Figure 9 shows the migration scheme for ODC-FC where the jobs are migrated to the four corners, where each job is migrated to the corner corresponding to its location.

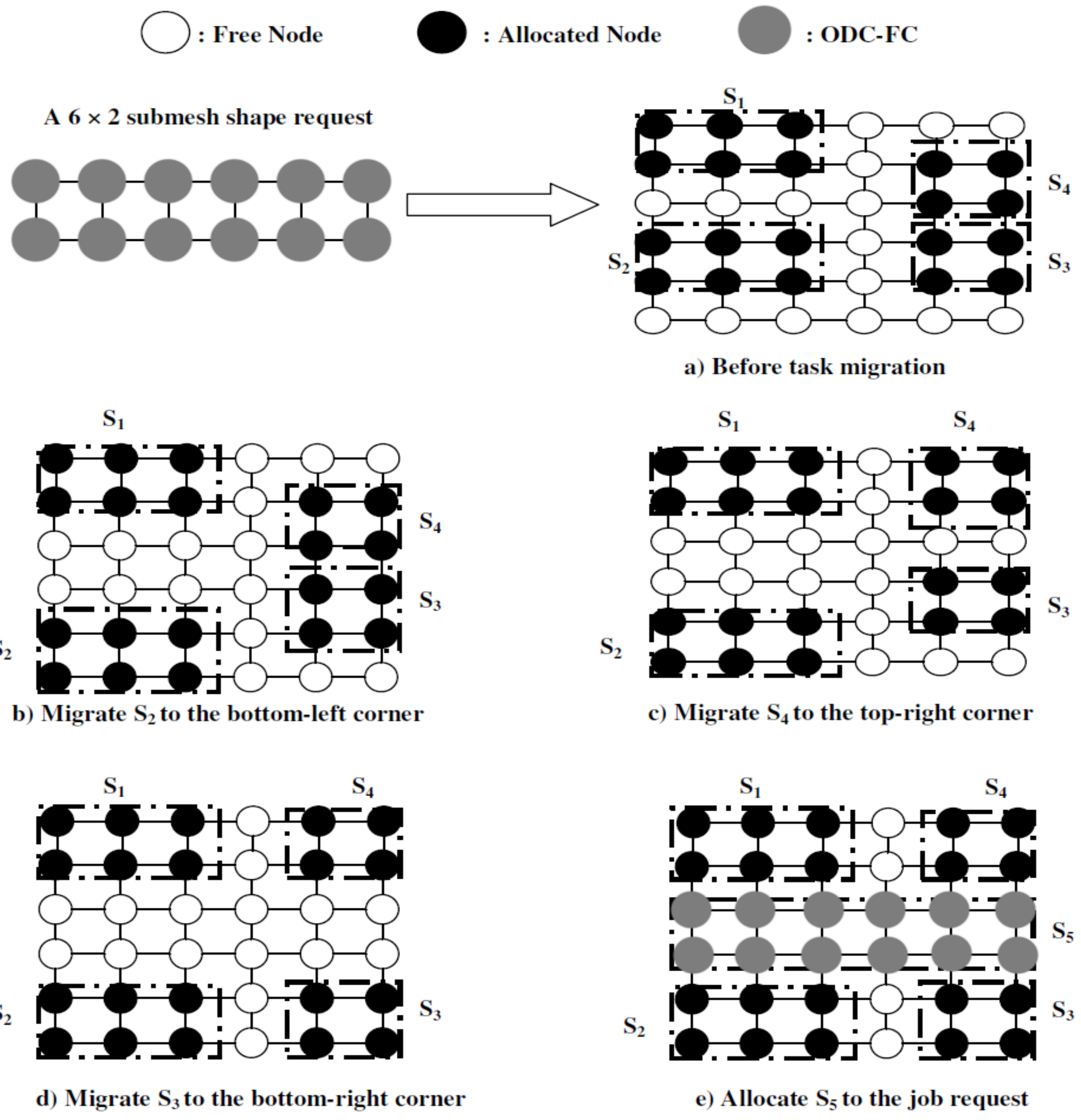
: Free Node
: Allocated Node
: ODC-FC
A 6 × 2 submesh shape request
S1
S2
S3
S4
S5
a) Before task migration
b) Migrate S2 to the bottom-left corner
c) Migrate S4 to the top-right corner
d) Migrate S3 to the bottom-right corner
e) Allocate S5 to the job request

Figure 9: An allocation using Online Dynamic Compaction-Four Corners strategy [21]

- Drawbacks of ODC-FC

ODC-FC has high number of migrations than CODC-FC and CODC-FCQQ, and migration is done even if it is not beneficial in allocation.

2.1.8 Conditional Online Dynamic Compaction-Four Corners (CODC-FC) strategy.

The CODC-FC strategy is a generalization of the ODC-FC strategy. In this strategy, the system is divided into four equal parts (four quarters) and processors are allocated to an allocation request by any processor allocation strategy chosen. If the number of requested processors is available and the allocation strategy fails to allocate a job request due to the lack of contiguity, then the job migration process is applied on the running jobs in order to migrate them to the four corners of the system according to the position of each running job. This migration aims to produce a large area of free adjacent processors in the middle of the system, and then the allocation strategy is used again to allocate a free sub-mesh for the job request [21].

In this strategy, the main advantage is the conditional migration, where the mesh is divided as in ODC-FC [21, 31] and the same allocation strategy is applied, but the migration is carried out if it results in successful allocation, this reduces the number of migrations and as a result the turnaround time is also reduced as opposed to the previous algorithms with migration. This is an advantage in comparison to ODC-SC and ODC-FC.

The disadvantage of CODC-FC is that it applies migration to all quarters at the same time even if single quarter migration can be enough which mean higher number of migrations. Figures 10 and 11 show the outline of CODC-FC allocation and de-allocation strategies respectively. In figure 12, the job with size 3x3 failed to be allocated, so migration is carried out for the four quarters then it is allocated in the freed sub-mesh.

Input : Dimensions of the required sub-mesh of the job to be allocated

Step1: Search (phase 1)

Search for a free sub-mesh using (FF) strategy.

IF a free sub-mesh is found, go to Step 5.

ELSE, check weather migration has been carried out since the last de-allocation.

IF migration has already been carried out, stop.

ELSE

IF the number of free processors in the 2D mesh is greater than or equal to the number of processors needed by the allocation request go to Step 2.

ELSE stop.

Step2: Migration Test

Do the migration using a copy of the data structures (FF on the copy).IF the job migration can result in successful allocation for the current allocation request, go to Step3. ELSE stop.

Step3: Migration

Apply job migration for all allocated jobs to the four corners according to the location of each allocated job. THEN, go to Step4.

Step4: search (phase 2) Search for a free sub-mesh using the FF strategy. Go to Step5.

Step5: Allocation

Allocate the free sub-mesh to the job. Update the lists and all data structures. Stop.

Figure 10: Outline of the allocation procedure for CODC-FC strategy [21, 31]

Input: job_id

Step1:

For all processors in the mesh network

If processor is allocated to the job

De-allocate it

Step2: Update the lists

Figure 11: Outline of the de-allocation procedure for CODC-FC strategy [21, 31]

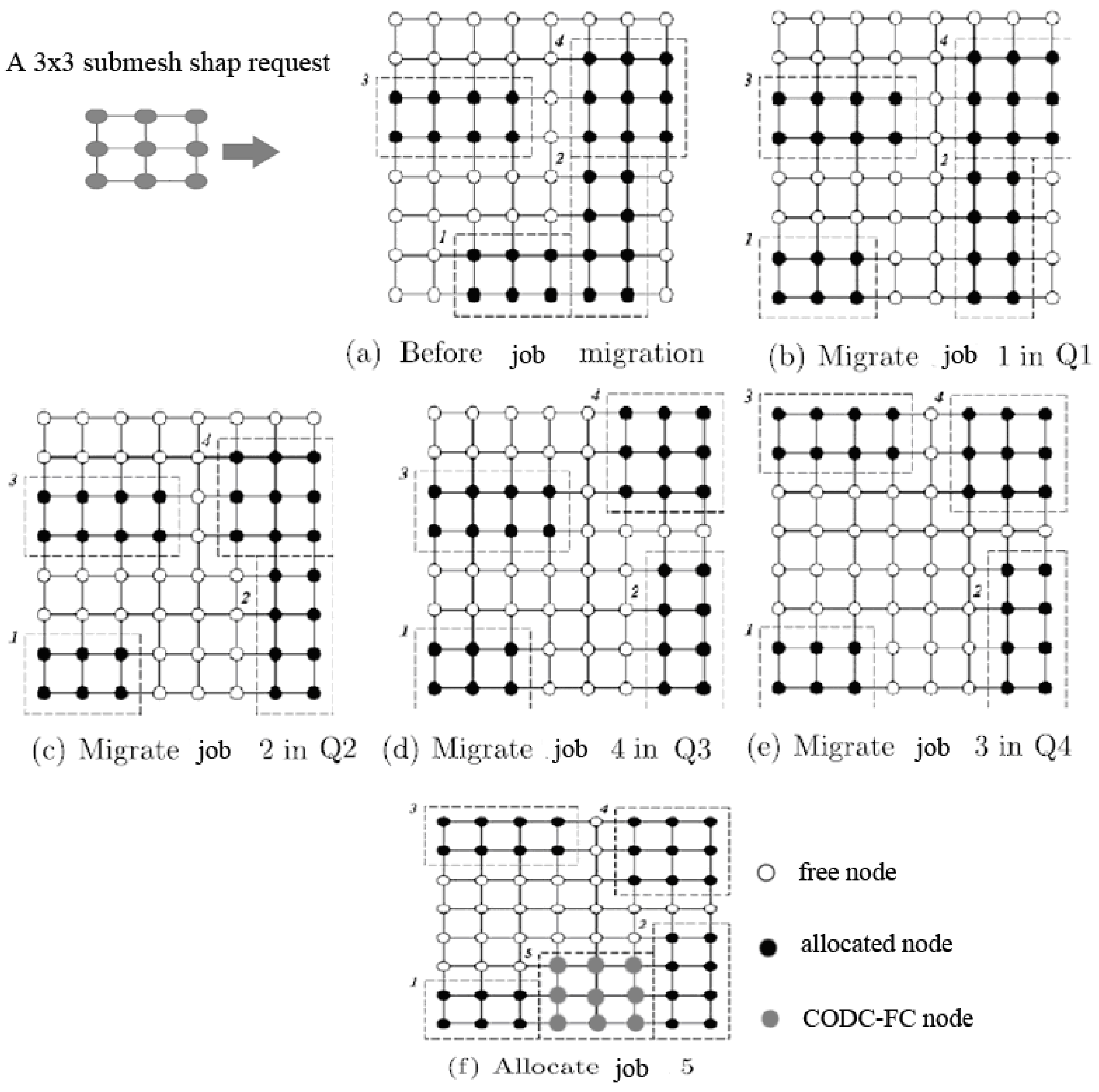

Figure 12: An allocation using Conditional Online Dynamic Compaction-Four Corners strategy [21, 31]

- **Drawbacks of CODC-FC**

In CODC-FC migration is done in all quarters even if an enough free submesh to allocate is freed with fewer migrations.

2.1.9 Conditional Online Dynamic Compaction-Four Corners Quarter by Quarter strategies.

The conditional online Dynamic Compaction-Four Corners Quarter by Quarter is based on ODC-FC except that it applies migration only when it can result in successful allocation for the current allocation request. In this migration strategy, we use the FF strategy as an allocation strategy because of its simplicity and performance is close to that of the BF strategy. The same job migration technique used in ODC-FC has been used in this strategy, but an improvement has been made to this strategy to improve the performance of ODC-FC in terms of average turnaround time and mean system utilization. The aim of this improvement is to reduce the cost of job migration.

In CODC-FC strategy, we keep all jobs that are allocated to the 2D mesh in four ordered lists; each list contains the jobs that belong to the same quarter, which means that the first list contains all jobs that are allocated to the first quarter. The second list contains all jobs that are allocated to the second quarter, and so on. After a job completes its execution and deallocation is carried out, the job is removed from the list that it belongs to. These lists are used to simplify the migration process [31].

In this scheme, the migration process is similar as CODC-FC but migration takes place on quarter-by-quarter bases. When the current allocation request fails, migration within the first quarter is carried out and if this does not lead to successful allocation, we cancel the operation and return the mesh to the previous state. Then we try to migrate within the second quarter, and so on. If migration within a quarter leads to successful allocation, we keep the migration state, this process of migration is called ***virtual migration***; where the migration is applied on a copy of the busy array that contains the location of the current allocated jobs, the virtual migration minimizes the number of migration which results in reduced turnaround time.

The accumulative migration process means that in the worst case scenario CODC-FCQQ will give the same results as CODC-FC. Using virtual migration may cause an increase in time of service but it is balanced in or even decreased as the cumulative migration have a better chance in giving a better result than CODC-FC [31].

Figure 13 shows the process of migration to the four corners on quarter-by-quarter basis, where the migration process starts in the first corner and then the second corner, and so on.

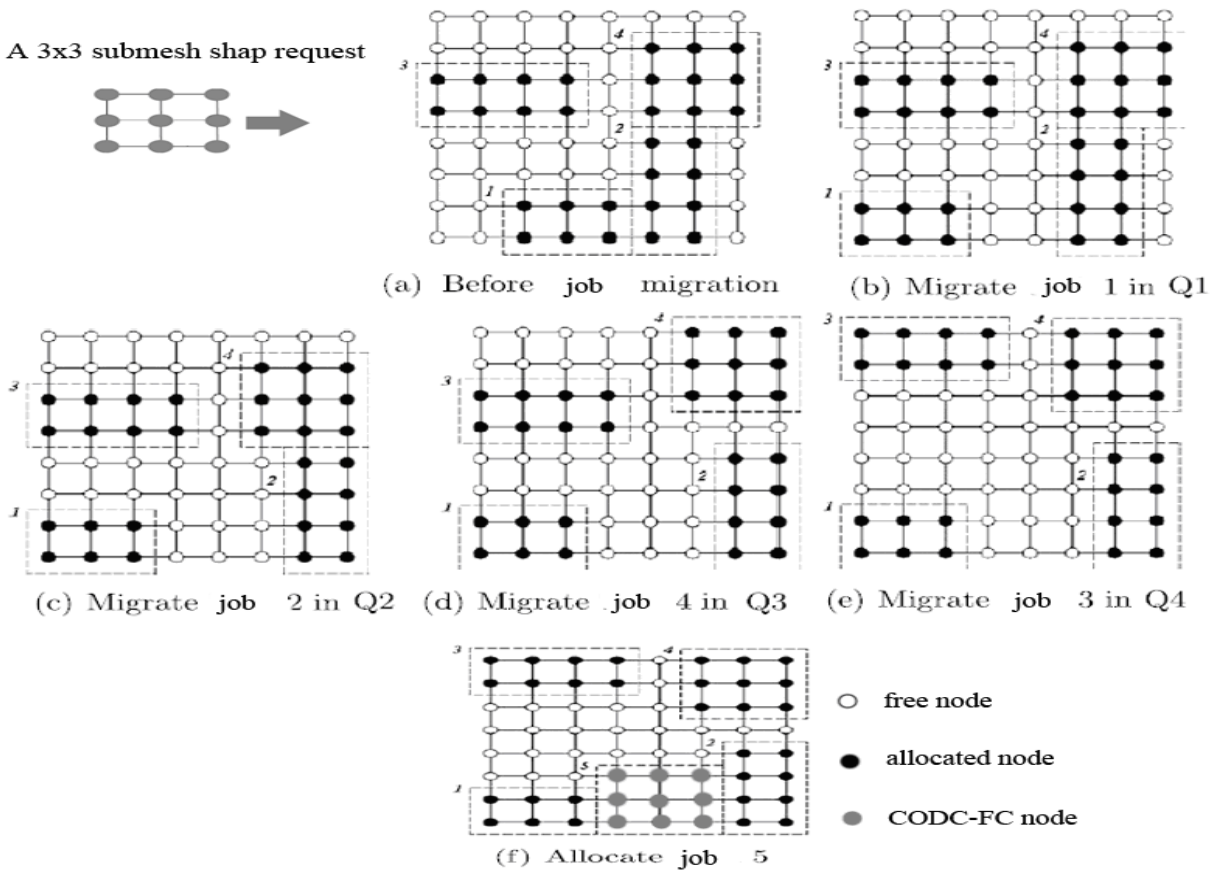

Figure 13: An allocation and migration using Online Dynamic Compaction-Four Corners strategy Quarter By Quarter [31]

- **Drawbacks of CODC-FCQQ**

In CODC-FCQQ migration can fail even if there is enough free nodes because of the shape of the incoming job don’t match the shape of the freed sub-mesh.

2.2 The Simulation Tool (ProcSimity Simulator)

This section briefly describes the ProcSimity simulation tool [25, 32]. ProcSimity is a simulation tool that has been built as a research tool for processor allocation and job scheduling in multicomputer systems [32, 10]. ProcSimity was developed at the University of Oregon [32], and has been supported by Oregon Advanced Computing Institute (OACIS) and National Science Foundation (NSF) [10]. The tool was written in the C programming language and has been extensively used in the simulation of processor allocation and job scheduling in mesh-connected multicomputer systems [8, 10, 29]. This is because it is an open-source tool that includes detailed simulations of important operations regarding multicomputer networks [10, 32].

The general purpose of the ProcSimity is to provide a suitable environment for performance analysis of processor allocation and scheduling for different algorithms. In particular, ProcSimity’s main job is to examine some of the processor allocation problems, as an example of fragmentation and communication overhead problems [2, 5, 7, 8, 10, 29]. ProcSimity has an architecture that consists of a network of processors interconnected through message routers at each node. Adjacent nodes are connected by two-way communication links, and communications such as messages may be passed using by either store-and-forward or wormhole switching. ProcSimity supports the mesh and k-ary n-cube interconnection topologies with dimension-ordered routing [10, 32].The

ProcSimity simulator has a number of preset algorithms regarding allocation and scheduling. Moreover, a new algorithm can be integrated into ProcSimity [10, 32].

2.3 Summary

In table 2.1 a comparison is illustrating the main problems of the allocation and migration strategies mentioned in the previous sections.

Table 2.1 Drawbacks of some allocation and migration strategies

strategy	**Main advantage**	**Main drawback**
First fit	Simplicity	External fragmentation
Best fit	Good performance	External fragmentation
Two-dimensional Buddy	Less time than FF	internal fragmentation
Frame sliding	Faster search	External fragmentation
Adaptive scan	Better performance than FS	External fragmentation
First-Fit Mesh Bifurcation	Less fragmentation than FF	External fragmentation
Online Dynamic Compaction-Single Corner	Improves utilization	Non beneficial migration
Online Dynamic Compaction-Four Corners	Better performance than ODC-SC	Non beneficial migration
Conditional Online Dynamic Compaction-Four Corners	Better performance than ODC-FC	Unnecessary migration
Conditional Online Dynamic Compaction-Four Corners Quarter by Quarter	Better performance than CODC-FC	External fragmentation because of the shape of the incoming job

Chapter Three

Conditional Online Dynamic Compaction - Full Mesh

3.1 Introduction

In most contiguous processor allocation strategies, the main drawback is the fragmentation problem, especially external fragmentation. This happens when an allocation request cannot be served in spite of the presence of enough free processes for the job request, but because of the contiguity condition the available non-contiguous processors cannot be utilized, also the shape of the incoming job doesn't match the shape of the free submesh [21].

To solve this problem, we can rearrange the allocated processors in the mesh to free enough adjacent processors to be able to allocate the requested processors. This process is referred to as the migration process. The main drawback in migration is the cost of migration because jobs must halt execution until the migration process ends and then re-execute the processes either from the previous state, which means that the previous state should be saved. This results in an extra cost.

3.2 Online Dynamic Compaction-Four Corners strategies

The most recent migration strategies depend on using First Fit as an allocation strategy then use migration if allocation fails and there are enough free processors to allocate because of its simplicity and performance is close to that of the Best Fit strategy, so migration is applied on the running jobs to free a sub-mesh to be allocated to an incoming job request. The most recent migration strategies divide the mesh into four quarters of the same size and migrates jobs within the quarter in a specific direction. As mentioned in sections 2.1.7, 2.1.8, and 2.1.9 in chapter 2.

3.2.1 Conditional Online Dynamic Compaction – Full mesh strategy

This strategy is based on the ODC strategies; ODC-SC and CODC-FCQQ strategies. This strategy uses the full mesh as ODC-SC to migrate jobs to the left bottom corner, it uses First Fit as an allocation strategy because of its simplicity and performance and also it uses the virtual migration process that is presented in CODC-FCQQ.

The performance of this strategy can be improved over CODC-FCQQ by switching the orientation of any allocation request that cannot be allocated in the requested orientation. This step has been used in the algorithm in case that the migration was unsuccessful in allocation, then the algorithm rotates the allocation request and then checks if allocation is successful; it almost like the game of Tetris, which in most cases works very well. This causes a big reduction in turnaround time and the number of migrations that is maximized because of the full mesh migration.

Also, the ordering process can be used to improve the performance of the allocation algorithm, which is the process of updating the busy array that contains the current location of the currently being serviced jobs; every time we migrate the job in order to reduce the gaps between the migrated jobs; the gaps are caused by the process of migration; as the current migration for any job is done in relative to the old location of the previous job, this process reduces the gaps and which in turn helps in the next migration, while the other algorithms update busy array when the algorithm finishes migration on the quarter or the full mesh system.

3.3 Performance Evaluation

In this section, the simulation results will be stated as an evaluation of the performance of the algorithm then it is compared to that of the most recent algorithms, CODC-FC and CODC-FCQQ. Since the performance of CODC-FC and CODC-FCQQ is better than that of ODC-FC and ODC-SC in terms of mean system utilization and job turnaround time, only these two strategies will be used in the comparison.

3.3.1 Simulation Results

Extensive simulations have been done in the ProcSimity simulation tool (which is written in C language and often used to simulate parallel systems) to compare the proposed algorithm against the selected algorithms; which are CODC-FC and CODC-FCQQ, all algorithms where implemented in this tool and tested.

The mesh used is a 2D mesh with height (*H*) and width (*W*). We use the scheduling strategy First Come First Served (FCFS) because of its fairness, an exponential distribution has been used to generate inter-arrival time using a unique time unit and also is used to generate the jobs execution times with a mean of one time unit. The time unit used in simulations is specific for the simulator and not a normal time unit such as; seconds, minutes, and hours. The time unit is measured by floating point values, which means that the values generated by the simulator are real numbers. The job distribution is uniformly distributed ranging from 1 to the mesh width and height and this controls the size of incoming jobs.

Table 3.1 shows an example of one of the input files used in the evaluation experiments using a 16x16 mesh, values may vary to simulate different situations and traffic conditions.

Table 3.1: The System Parameters Used in the Simulation Experiments.

Simulator Parameter	**Values**
Dimensions of the Mesh Architecture	16×16
Allocation Strategy	FF
Scheduling Strategy	FCFS
Job Size Distribution	Uniform: Job widths and heights are uniformly distributed over the range from 1 to the mesh side lengths.
Execution Time Distribution	Exponential with a mean of one time unit.
Inter-arrival Time	Exponential with different values for the mean. The values are determined through experimentation with the simulator, ranged from lower values to higher values.
Number of Runs	The number of runs should be enough so that the confidence level is 95% that relative errors are below 5% of the means. The number of runs ranged from dozens to thousands.
Number of Jobs per Run	1000

The last two inputs in the file above is the number of jobs and the number of runs and these two parameters are used to determine the confidence level of the simulation, the simulation experiments have been conducted with a confidence level of 95% with a 5% relative error below the means of the simulation results. There are three evaluation metrics which shows the performance of the algorithms considered in this thesis. These are job turnaround time – which is the time that a job spends in the system (mesh) from arrival to departure-, utilization – which is the percentage of utilized processors over time, and number of migrations- which is the number of movements that the jobs takes in the process of migration-.

The main variable that is independent and imperative in the simulation process is the system load, which is the inverse of the mean inter-arrival time of the jobs. The system load is the value that determines the degree of load on the system which could be low, medium or heavy, this rang is crucial to test the best values for the evaluation metrics tested. The figures listed below show a comparison between the considered algorithms and composed of a system load on *x-axis* and the metric of performance on the *y-axis*.

<u>Turnaround Time</u>

In Figures 14, 15 and 16, the job turnaround time for multiple algorithms against system load shows how fast the algorithm processes a job. The main algorithm in this thesis (CODC-FM) and the one (CODC-FCQQ) that is compared with it uses FF as an allocation strategy and use virtual migration. The results show that the proposed algorithm CODC-FM is superior to all of the algorithms that are included in the figure. Moreover, the algorithm CODC-FCQQ is almost identical in performance to the previous algorithm CODC-FC and they are superior to the previous algorithms whether they use migration or not, while the proposed algorithm CODC-FM performs better than the previous algorithms considered. When comparing CODC-FCQQ with our proposed algorithm CODC-FM In table 3.2, the table demonstrates the values included in the figures below; in which the lower value is considered as an advantage, so we see a value of (25.448994) at (2.1) system load of the CODC-FM algorithm against (57.946968) for the closest algorithm which is CODC-FCQQ at the same system load for an 8X8 mesh In figure 14, also a value of (26.65143) against (42.097273) in favor of CODC-FM for a 22X16 mesh In figure 15, and a value of (13.42729) against (37.244583) in favor of CODC-FM for a 32X32 mesh In figure 16. The formula used to calculate the values in the table is listed below.

Formula 1 : response time = response time + (average response time /no. of runs)

Table 3.2: average turnaround time for the algorithms considered in this thesis compared to other algorithms.

System load	ODC-FC	FF	CODC-FCQQ	CODC-FM
0.1	1.079001	1.030232	1.03208	1.023223
0.3	1.272198	1.11385	1.126135	1.099058
0.5	1.518925	1.240075	1.265202	1.211681
0.7	1.830567	1.430026	1.461612	1.366162
0.9	2.239858	1.733065	1.744113	1.5733
1.1	2.807414	2.24741	2.17395	1.862186
1.3	3.686752	3.307573	2.908342	2.276927
1.5	5.406547	6.212885	4.474857	2.934021
1.7	9.776795	15.678181	8.702282	4.152281
1.9	20.927063	35.124291	19.760804	6.941841
2.1	38.423574	56.880197	37.244583	13.42729

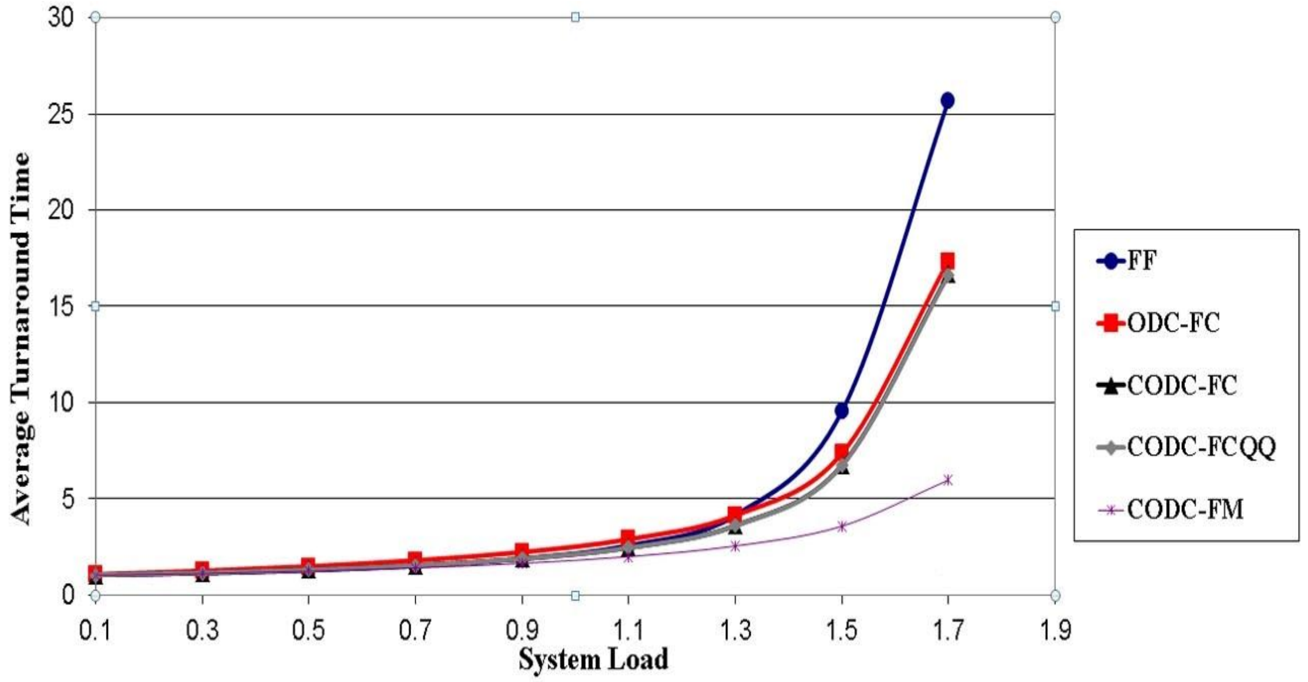

Figure 14: Average Turnaround Time vs System Load for a 8X8 mesh

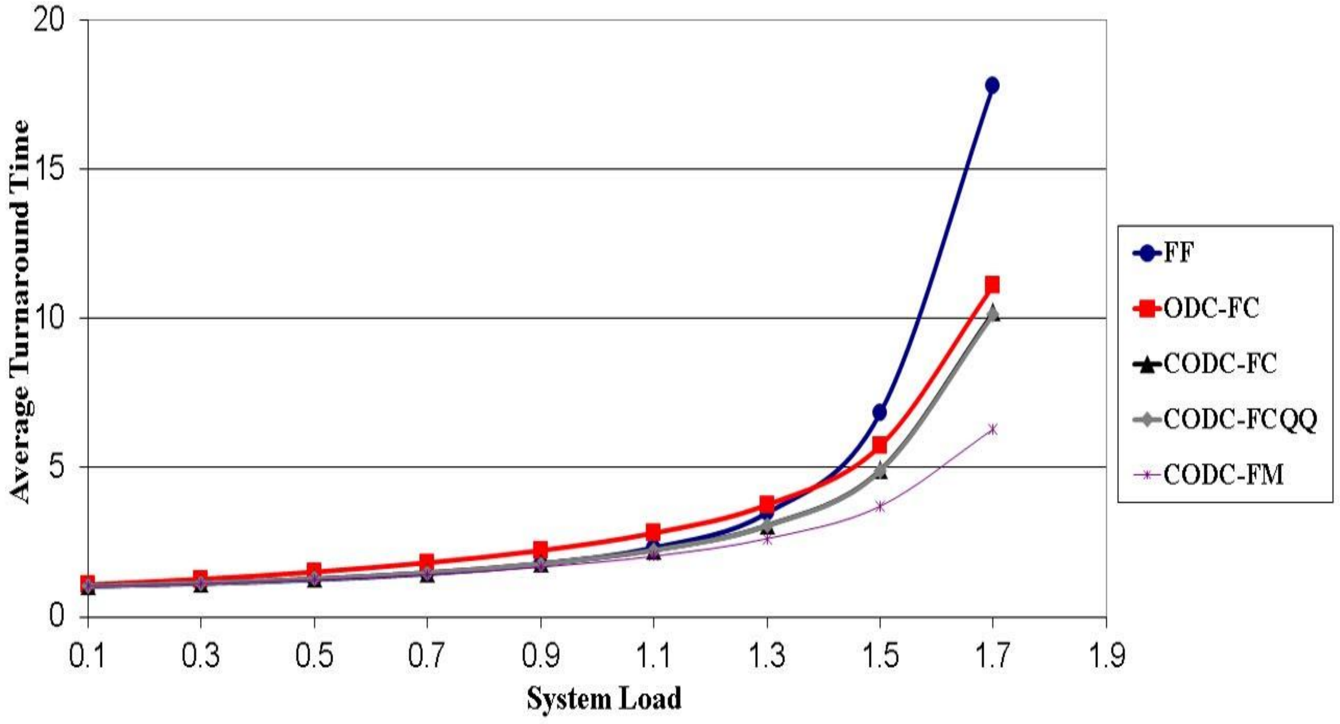

Figure 15: Average Turnaround Time vs System Load for a 22X16 mesh

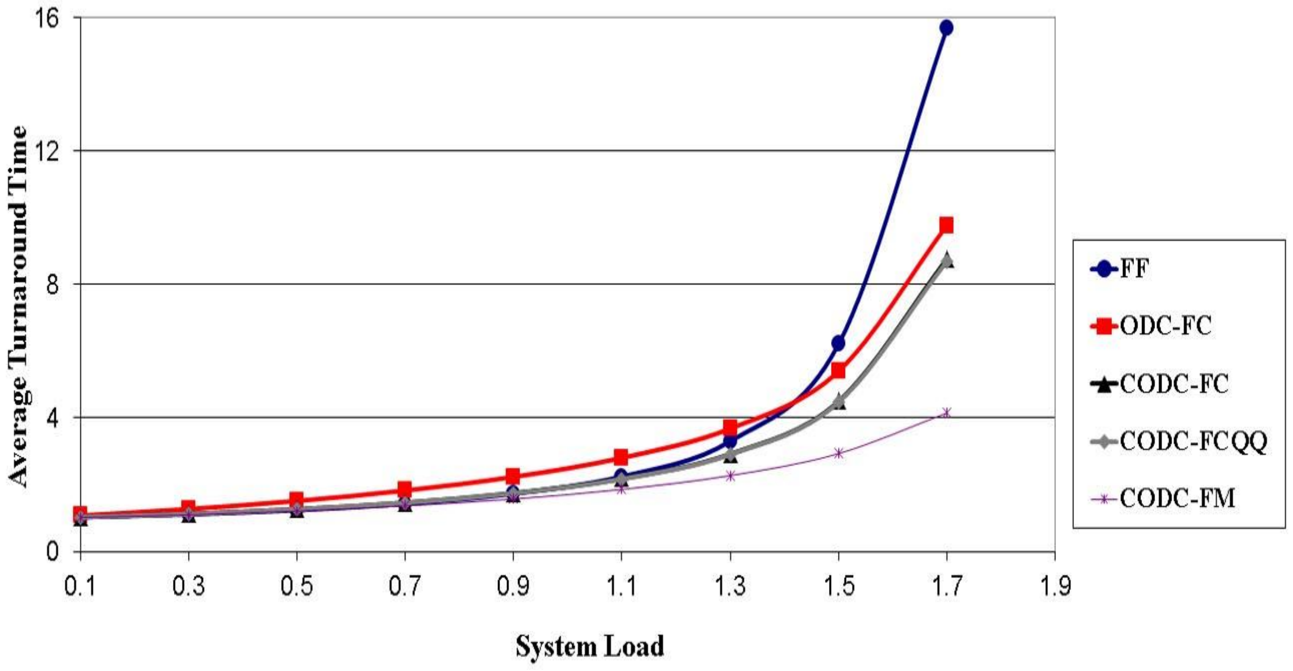

Figure 16: Average Turnaround Time vs System Load for a 32X32 mesh

<u>Utilization</u>

In figures 17, 18 and 19, utilization for multiple algorithms against system load shows how the algorithm utilizes processors. When we compare our proposed algorithm CODC-FM against the previous algorithms (CODC-FCQQ as an example), CODC-FM achieves higher utilization. CODC-FCQQ is better in performance comparing to the previous algorithms because of cumulative migration. This causes fewer migrations which in part maximizes utilization since utilization depends on job service time and the cost of migration. The reason that makes CODC-FM performs better in utilization than CODC-FCQQ is the job orientation switch which reduces job migration since in some cases allocation is done without migration. In figures 17, 18, and 19, we compare the system utilization for different algorithms with different mesh sizes. As an example, when comparing CODC-FCQQ with CODC-FM- the data in table 3.3, which shows some values included in the charts below;

where the higher value is considered as an advantage, therefore we see a value of (0.609426) for CODC-FM against (0.538851) for CODC-FCQQ in favor of CODC-FM for an 8X8 mesh with a (2.1) system load In figure 17, also a value of (0.532494) against (0.501032) in favor of CODC-FM for a 22X16 mesh In figure 18, and a value of (0.538343) against (0.488739) in favor of CODC-FM for a 32X32 mesh In figure 19. The formula used to calculate the values in the table is listed below.

Formula 2: system utilization = system utilization + (run system utilization / no. of runs)

Table 3.3: average system utilization for the algorithms considered in this thesis compared to other algorithms.

System load	ODC-FC	FF	CODC-FC	CODC-FCQQ	CODC-FM
0.1	0.031767	0.031767	0.031767	0.031767	0.031767
0.3	0.095277	0.095277	0.095277	0.095277	0.095277
0.5	0.158745	0.158743	0.158745	0.158745	0.162634
0.7	0.222145	0.222138	0.222145	0.222145	0.222159
0.9	0.28542	0.285394	0.28542	0.28542	0.285481
1.1	0.348447	0.348352	0.348445	0.348445	0.348647
1.3	0.410932	0.410616	0.410934	0.410936	0.411542
1.5	0.471209	0.468314	0.471202	0.47122	0.47387
1.7	0.517484	0.501934	0.517441	0.517546	0.534107
1.9	0.535067	0.509731	0.535025	0.535207	0.583855
2.1	0.538674	0.511529	0.538629	0.538851	0.609426

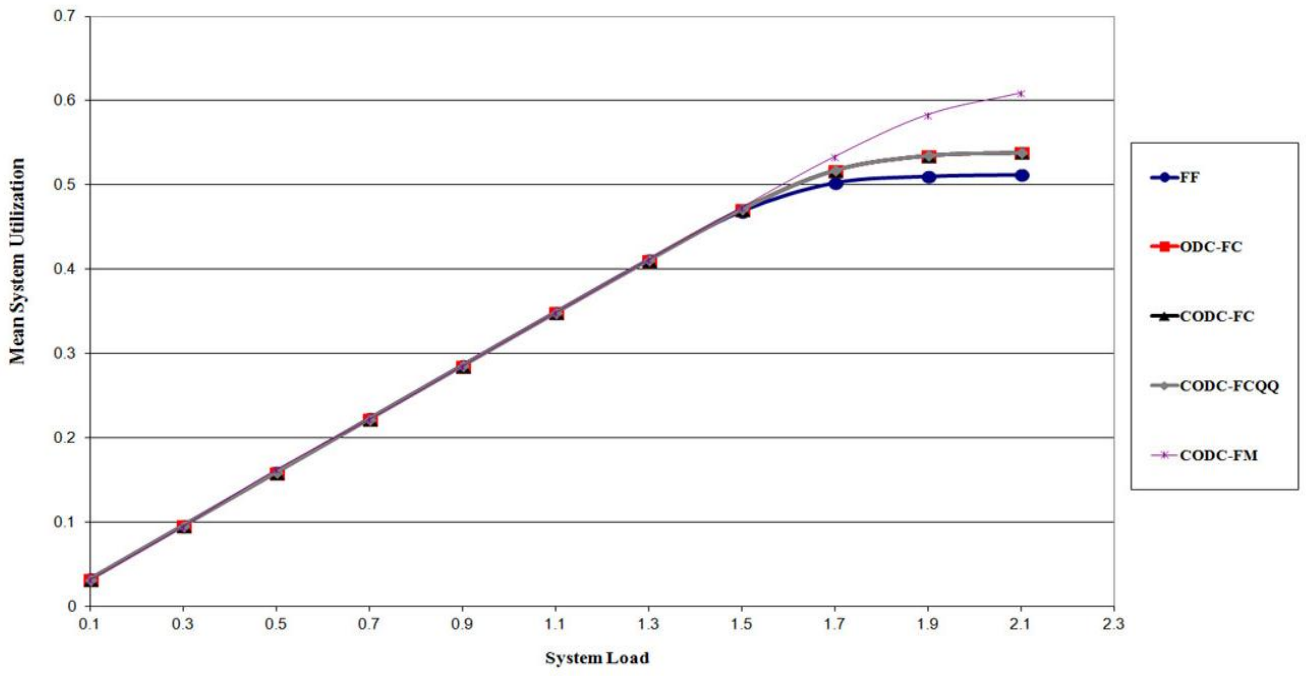

Figure 17: Mean System Utilization vs System Load for a 8X8 mesh

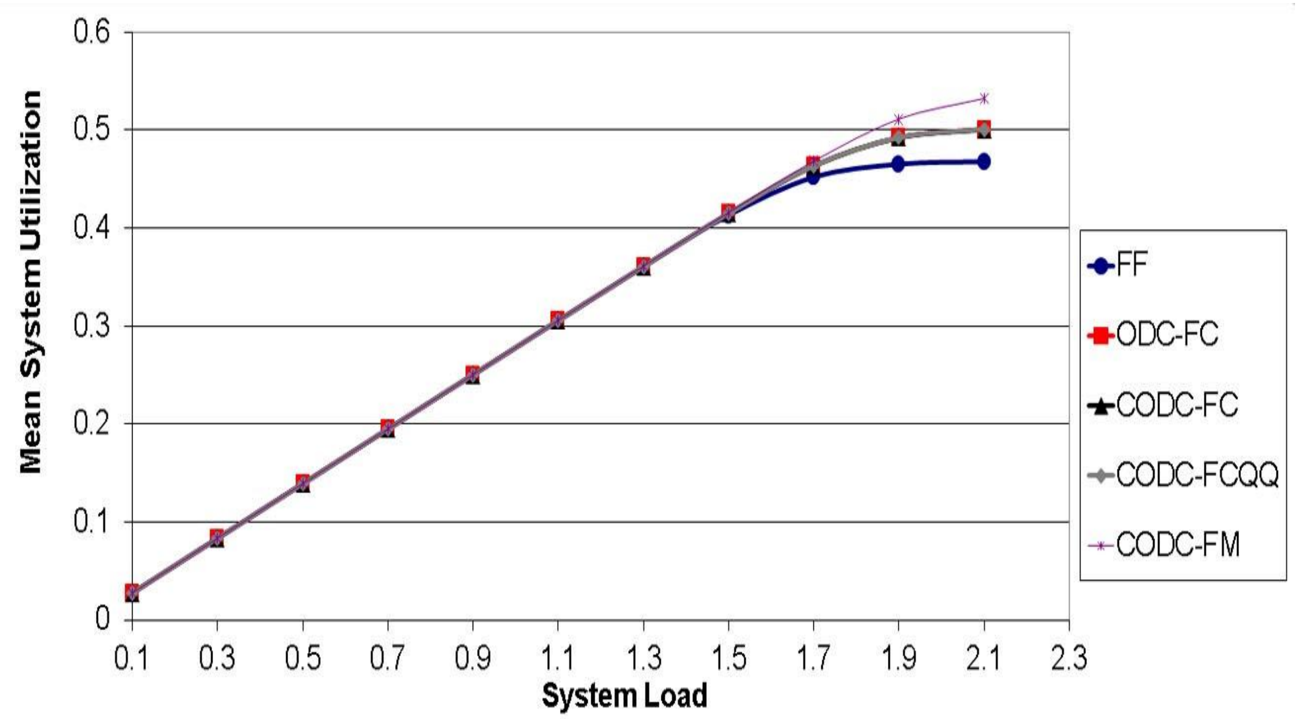

Figure 18: Mean System Utilization vs System Load for a 22X16 mesh

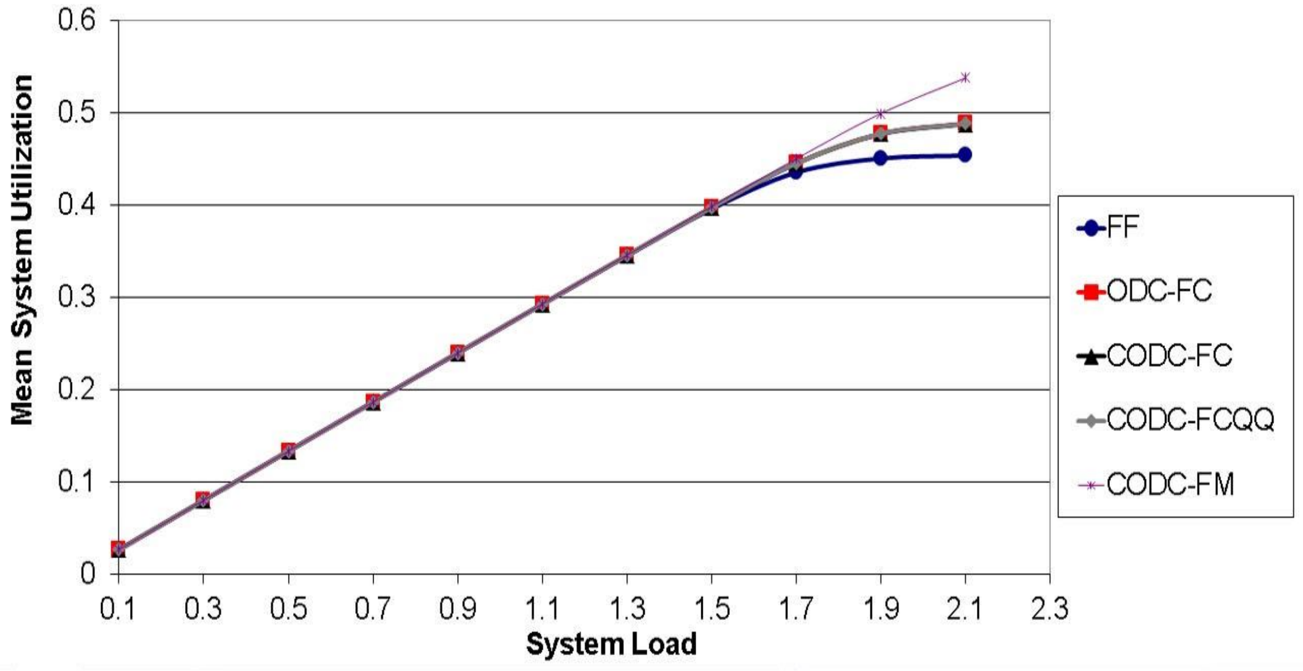

Figure 19: Mean System Utilization vs System Load for a 32X32 mesh

Migration

Figures 20, 21 and 22, Show the average number of migration for multiple algorithms against system load. When we compare against the algorithm in this thesis we see that CODC-FM is only superior to ODC-FC. The reason that CODC-FM has higher in number of migrations is the use of the full mesh system as a migration area, causing higher migrations which in part minimize utilization since utilization depends on job turnaround time and the cost of migration. But since migration is carried out in case of failure of allocation the effect is negligible, also the algorithm in this thesis has a step that should be taken before the migration process which is the orientation switch that may substitute the migration process if it is successful. Also the use of the full mesh in migration can produce a large free sub-mesh that can accommodate more jobs to be serviced, and the ordering process helps with the size of the freed sub-mesh.

In Figures 20, 21, and 22, we compare the number of migrations for the different algorithms. When comparing CODC-FCQQ with CODC-FM as an example, table 3.5 shows some values included in the charts below; the higher value is considered as a disadvantage- we see a value of (111.45) for CODC-FM against a value of (75.21) for CODC-FCQQ for an 8X8 mesh as with a (2.1) system load In figure 20, also a value of (129.3225) against (102.335) in favor of CODC-FCQQ for a 22X16 mesh In figure 21, and a value of (169.56) against (114.27) in favor of CODC-FCQQ for a 32X32 mesh In figure 22. The formula used to calculate the values in the table is listed below.

Formula 3: migration counter = migration counter + migration counter per run / no. of runs

Table 3.4: average number of migrations for the algorithms considered in this thesis compared to other algorithms.

System load	ODC-FC	FF	CODC-FC	CODC-FCQQ	CODC-FM
0.1	9.1125	0	0.5025	0.5025	0.7475
0.3	33.3225	0	3.81	3.7675	5.4225
0.5	63.4775	0	9.375	9.2725	12.7375
0.7	100.035	0	16.95	16.615	22.655
0.9	141.725	0	25.835	25.1925	33.3125
1.1	190.5925	0	36.6575	35.49	45.8275
1.3	244.19	0	48.6225	47.095	59.6225
1.5	300.2875	0	61.405	59.405	74.805
1.7	347.345	0	72.5175	70.0175	90.615
1.9	366.17	0	76.865	74.1925	104.27
2.1	370.2225	0	77.905	75.21	111.45

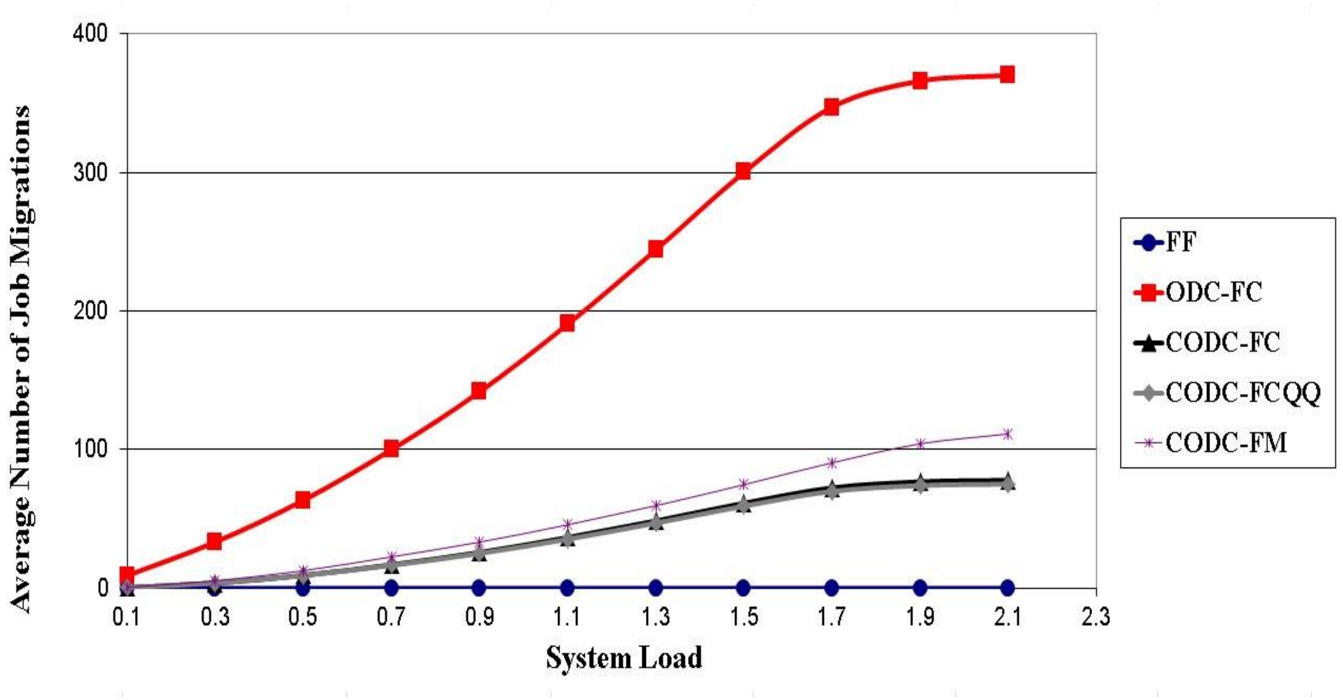

Figure 20: Average Number of Migrations vs System Load for a 8X8 mesh

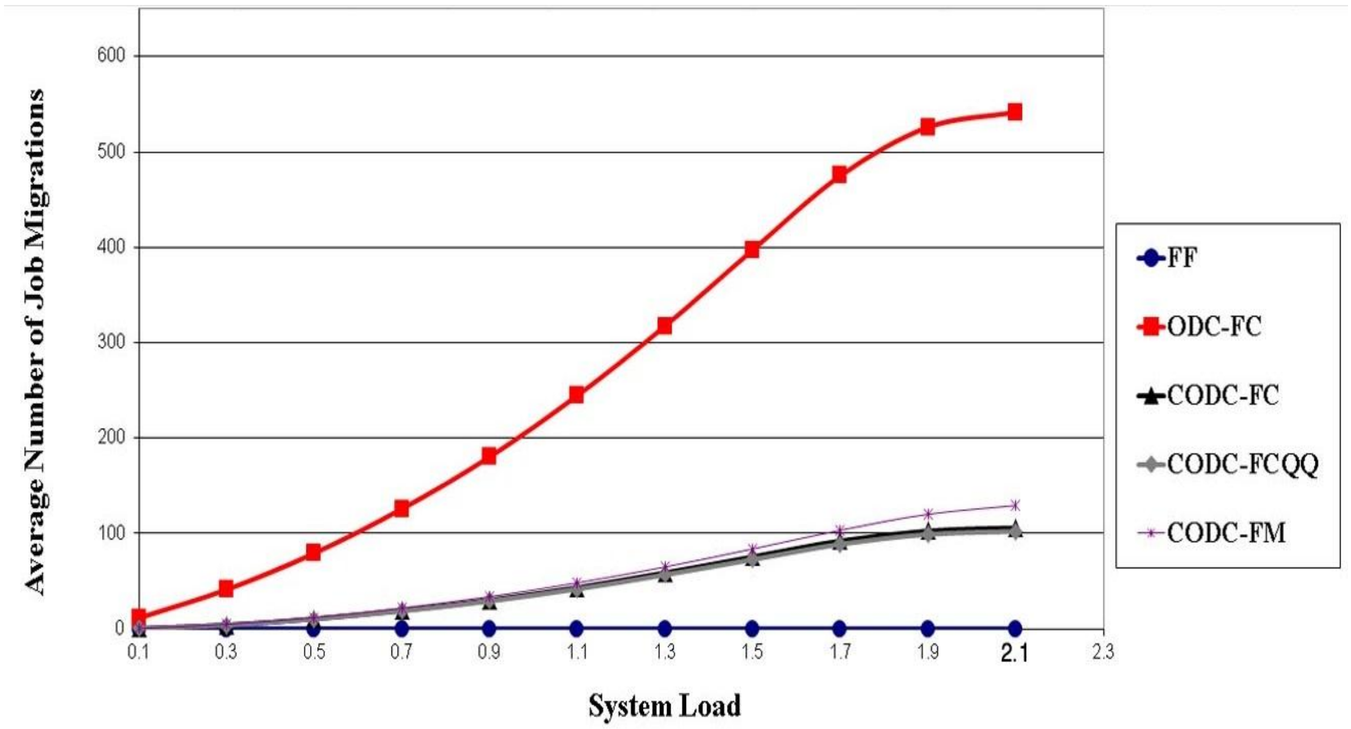

Figure 21: Average Number of Migrations vs System Load for a 22X16 mesh

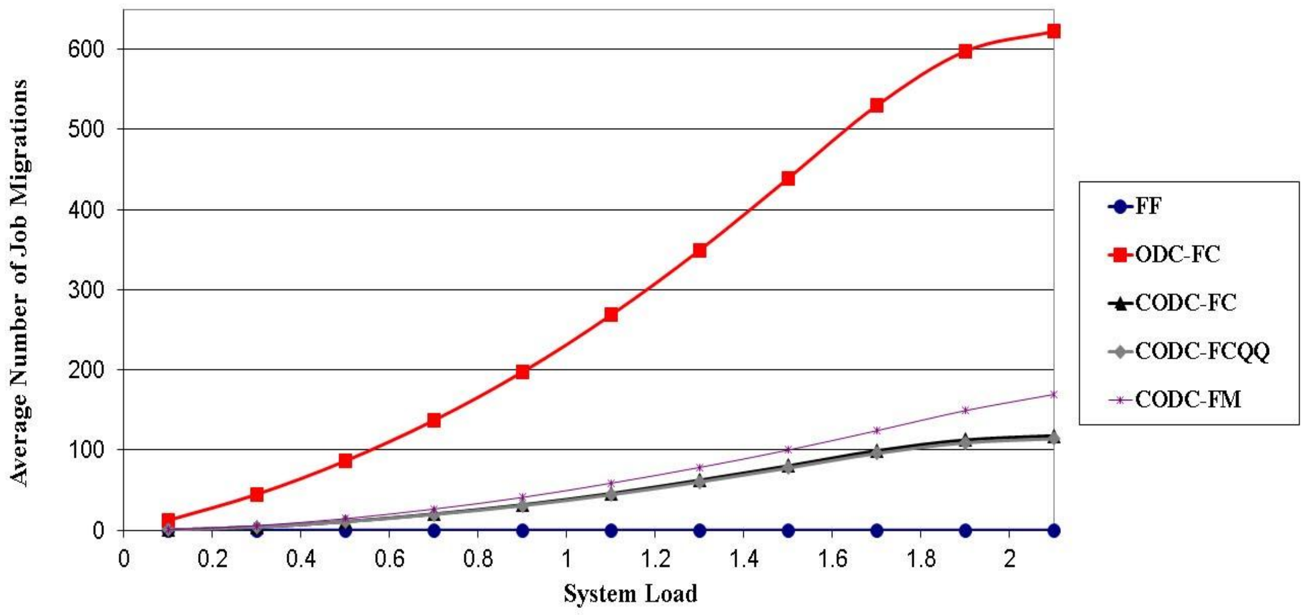

Figure 22: Average Number of Migrations vs System Load for a 32X32 mesh

3.4 Conclusions

While different migration strategies improve the performance of most allocation strategies in different ways, the suggested algorithm Conditional Online Dynamic Compaction-Full Mesh with switch (CODC-FM for short), is an improvement of the previous algorithms as shown by the simulation results, the performance of CODC-FM is better in job turnaround time and mean system utilization, although it has a higher number of migrations than most of the previous algorithms, CODC-FM balance the effect of the high number of migrations with lower job turnaround time since it reduces service time a great deal.

The most promising update in the algorithm which improved the performance is job orientation switch. CODC-FM has the best performance and the best sides of further improvement such as the migration area.

Chapter 4

Conclusions and Future Work

4.1 Summary of the Results

The main purpose of this thesis is to come up with a new and improved migration strategy for 2D mesh-connected multicomputer systems, below are some areas of improvement that the new strategy accomplished.

- The improvements of the previous algorithms shows that improvement can be done in some areas, this is the motivation for a new migrations strategy that has higher performance than the previous ones. Most migration strategies have a close in job turnaround time. In addition, utilization of processors is an important issue. The new migration strategy CODC-FM depends on using the full mesh as a migration area and the most important criteria is that in case of migration failure to allocate a job it switch the orientation (i.e., rotation) of the allocation request, which is expected to improve the system performance.
- The simulation has shown that the performance of CODC-FM is better than that of the previous migration strategies. The proposed algorithm is better than that of the previous algorithms with up to 177% improvement in terms of job turnaround time in high system loads and with an average of 54% overall system loads. Also, the results show an increase in system utilization up to 9.2% in high system loads with an average of 4.6% overall system loads. Moreover, the results show that an

increase in the number of migrations by 30% overall system loads which have little effect on the overall performance of the new algorithm because of the improvement in the different aspects of the algorithm.

4.2 Directions for the Future Work

Many areas of this research work are open for improvement, where the problems that can be investigated are mentioned below.

- It would be interesting to change the way of dividing the mesh system and then evaluate the performance of the allocation strategies in such way.
- The job orientation (i.e., job rotation) is done after migration fails to allocate. Therefore, we can change the first fit algorithm to use job orientation within the algorithm or after the initial allocation fails to allocate the job.

الملخص

تم في هذه الرسالة اقتراح خوارزمية جديدة لترحيل المهام اثناء تنفيذها في النظام بهدف التحسين في أداء النظام مقارنة مع الخوارزميات القديمة المستخدمة لترحيل المهام اثناء التنفيذ، . يتم التركيز في الخوارزمية المقترحة على توسيع منطقة البحث لتشمل الشبكة كاملة مما يزيد من عمليات الترحيل، والهدف من ذلك أن تصبح المنطقة المفرغة أكبر منها في الطرق السابقة. وأيضاً، يتم تنفيذ الترحيل فقط عندما ينتج عنه تخصيص ناجح، وهذا يعني تجنب العمليات الغير مفيدة. بالإضافة الى ذلك، في حال فشل التخصيص للطلب يتم تبديل ابعاد هذا الطلب، وذلك لزيادة احتمل التخصيص الناجح، وهناك ايضاً خاصية أخرى مهمة لطريقة الترحيل المقترحة وهي التقليل من مساحة البحث عن طريق ايقاف عملية البحث عند ايجاد شبكة جزئية كبيرة بما يكفي للتخصيص الناجح مما يجعل الترحيل يتم على اساس تراكمي (تحديث عملية الترحيل بشكل تراكمي بعد كل مهمة).

تشير النتائج في هذه الرسالة الى ان الخوارزمية الجديده المقترحة تحسن من الأداء من ناحية معدل وقت المكوث في النظام (turnaround time) بشكل كبير، كما تحسن من استغلال النظام مقارنة بالخوارزميات السابقة.

تعاني الخوارزمية الجديدة من مشكلة واحدة فقط وهي زيادة عدد مرات الترحيل لأن الترحيل يتم تنفيذه على جميع البرامج قيد التنفيذ في الشبكة كاملة وجميعها ترحل لنفس الإتجاه وهو الزاوية اليسرى السفليه للشبكة، هذه العملية تتسبب بزيادة عدد مرات الحركة التي تنفذ على الوظيفة على الرغم من أن الترحيل يتم على البرامج قيد التنفيذ كل على حده.

References

1. Ababneh, I. (2001), Job scheduling and Contiguous Processor Allocation for Three Dimensional Mesh Multicomputers, AMSE advances in Modeling& Analysis, 6(4), 43-58.

2. Ababneh, I. (2006), An Efficient Free-list submesh Allocation Scheme for Two Dimensional Mesh-connected Multicomputers, Journal of Systems and software, 79(8), 1168-1179.

3. Ababneh, I. (2008), Availability-Based Noncontiguous Processor Allocation Policies for 2D Mesh-connected Multicomputers, Journal of systems and software, 81(7), 1081-1092.

4. Ababneh, I. (2009), On submesh Allocation for 2D Mesh Multicomputers Using the Free List Approach: Global Placement schemes, Journal of performance Evaluation, 6(2), 105-120.

5. Ababneh, I. and Fraij F. (2001), Folding Contiguous and Non-contiguous Space Sharing for Parallel computers, Mu'tah Lil-Buhuth wad-Dirasat Natural and Applied Sciences Series, 16(3), 9-34.

6. Athas, W.C. and Seitz, C.L, (1988), Multicomputers: Message-Passing Concurrent Computers, IEEE Computer, 21(8), 9-24.

7. Bani Mohammad, S., Ouald-Khaoua, M., Ababneh, I., and Machenzie, L. (2006), Noncontiguous 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 Com purer Society Press,41-48.

8. Bani Mohammad, S., Ould-Khaoua, M. and Ababneh, I. (2007), A New Processor Allocation strategy with a High Degree of Contiguity in Mesh-Connected Multicomputers, Journal of Simulation Modeling, Practice and Theory (SIMPRA), Elsevier Science, 15(4), 465-480.

9. Bani-Mohammad, S. (2008), Efficient Processor Allocation Strategies for Mesh-Connected Multicomputers. Ph.D. Thesis, Department of Computing science, University of Glasgow.

10. Bani-Mohammad, S., Ould-Khaoua, M., and Ababneh, I. (2007), An Efficient Non- Contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers, Journal of information Sciences. 177(14), 2867-2883.

11. Blumrich, M., Chen, D., Coteus, P., Gara, A., Giampapa, M., Heidelberger, P., Singh, S., Steinmacher-Burow, B., Takken, T., and Vranas, P. (2003), Design and Analysis of the Blue Gene/L Torus Interconnection Network. IBM Research Report RC23025, IBM Research Division, Thomas J. Watson Research Center.

12. C´esar A. F. De Rose, Hans-Ulrich Heiss, and Philippe A. O. Navaux (2007), Distributed Processor Allocation in Multicomputers, Journal of Parallel Computing. 33(3), 1145-158

13. Chang, C.Y. and Mohapatra, P. (1998), Performance Improvement of Allocation schemes for Mesh-connected Computers, Journal of Parallel and Distributed Computing, 52(1) 40-68.

14. Chiu, G-M., and Chen, S-K. (1999), An Efficient Submesh Allocation Scheme for Two Dimensional Meshes with Little overhead, IEEE Transactions on Parallel and Distributed Systems, 10(5), 471-486.

15. Choo, H., Yoo, S-M., and Youn. H.Y. (2000), Processor Scheduling and Allocation for 3D Torus Multicomputer Systems, IEEE Transactions on Parallel and Distributed Systems, 11(5), 475-484.

16. Chuang. P.-J., and Tzeng, N-F. (1991), An Efficient Submesh Allocation Strategy for Mesh Computer Systems, Proc, 1991 Int'l Conf. Distributed Computer systems, 256-263.

17. Cray, (2004), Cray XT3 datasheet. http://www.cray.com/downloads/Cray_XT3_Datasheet.pdf, last seen (1/10/2015)

18. Ding, J., and Bhuyan, L.-N. (1993), An Adaptive Submesh Allocation Strategy for Two Dimensional Mesh Connected Systems, Proceedings of the 1993 International Conference on Parallel Processing, 193-200.

19. Foster, I. (1995), Designing and Building Parallel Programs, Concepts and Tools for Parallel Software Engineering, Addison-Wesley.

20. Gabrani, G., and Mulkar, T. (2005), A Quad-Tree Based Algorithm for Processor Allocation in 2D Mesh-Connected Multicomputers, Journal of Computer Standards Interfaces, 27(2), 133-147.

21. Goh L.-K., and Veeravalli, B. (2008), Design Performance Evaluation of combined and First-Fit Task Allocation and Migration strategies in Mesh Multiprocessor Systems, Journal of Parallel Computing, 34(9), 508-520.

22. Hamed, M. (2010), Evaluation of Common Scheduling and Contiguous Allocation strategies for Different Parallel Job Size Request Shapes. M.Sc. Thesis, Faculty of Graduate Studies, Jordan University of Science and Technology.(Unpublished)

23. Intel Corporation, (1991), Paragon XPS Product Overview. http://wenku.baidu.com/view/2c07f0264b35eefdc8d333dc.html, last seen (1/10/2015)

24. Kim, G., and Yoon, H. (1998), On Submesh Allocation for Mesh Multicomputers: A Best Fit Allocation and a Virtual Submesh Allocation for Faulty Meshes, IEEE Transactions on Parallel and Distributed Systems, 9(2), 175-185.

25. Kumar V., Grama A., Gupta A., and Karypis G., Introduction to Parallel Computing, Person education Limited,2nd edition, ISBN 0-201-64865-2.

26. Kumar. V, Grama, A., Gupta, A., and Karypis, G. (2003), Introduction to Parallel Computing, The Benyamin/Cummings publishing Company Inc., Redwood City, California.

27. Li, K. and Cheng, K.-H (1991), A Two-Dimensional Buddy System for Dynamic Resource Allocation in a Partition able Mesh connected system, Journal of Parallel and Distributed Computing, 12(1), 79-83.

28. Liu, T., Huang, W.-K., Lombardi, F. and Bhuyan, L. N. (1995), A Submesh Allocation Scheme for Mesh-connected Multiprocessor systems, Proceedings of the International Conference Parallel Processing II, 159-163.

29. Lo. V., Windisch, K., 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.

30. M. Morris Mano, Computer System Architecture, Person Education Limited, 3rd edition, ISBN 0-13-175563-3.

31. Olimat, M. (2012), Job migration in contiguous processor allocation for two-dimensional multicomputer, M.Sc Thesis, department of computer science, Jordan University of science and technology.(Unpublished)

32. ProcSimity V4.3 User's Manual, (1996), University of Oregon.

33. Quinn, J. (1994), Parallel computing, Theory and practice, McGraw-Hill.

34. Seo, K.-H., and Kim, S.-C (2003), Improving System Performance in Contiguous Processor Allocation for Mesh-Connected Parallel systems, The Journal of Systems and software, 67(1), 45-54.

35. Sharma, D. and Pradhan, D.K., (1998), Job Scheduling in Multicomputers, IEEE Transactions on Parallel and Distributed Systems, 9(1), 57-70.

36. Sharma, D., and Pradhan, D.k., (1996), Submesh Allocation in Mesh-Multicomputers Using Busy-List: A Best-Fit Approach with Complete Recognition Capability, Journal of Parallel and Distributed Computing, 36(2), 106-118.

37. Windisch K., J. V. Miller J.V., and Lo V. (1995), 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.

38. Yoo, B.S. and Das, C.-R. (2002), A Fast and Efficient Processor Allocation scheme for Mesh-connected Multicomputers, IEEE Transactions on Parallel & Distributed Systems, 51(1), 40-60.

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

40. Mehdi Modarressi, Marjan Asadinia, Hamid Sarbazi-Azad (2013), Using task migration to improve non-contiguous processor allocation in NoC-based CMPs, Journal of Systems Architecture 59 (2013) 468–481.

41. Jim Ng, Xiaohang Wang, Amit Kumar Singhz, Terrence Mak (2015), DeFrag: Defragmentation for Efficient Runtime Resource Allocation in NoC-based Many-core Systems, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, Guangzhou Institute of Advanced Technology, CAS, China, Department of Computer Science, University of York, UK., Shenzhen Institute of Advanced Technology, CAS, China.