International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 05, 2020
ISSN: 1475-7192

The Non-Contiguous Allocation Strategies Performance for k-ary n-cube Connected Multi Computers Using Smallest Job First Scheduling Strategy

Doreyed Muhammed Ahmed Awaad Al-Kerboly

Abstract--- *The performance of the famous non-contiguous allocation algorithms (Paging (0), Multiple Partner, and Random) proposed for k-ary n-cube connected multi computers have been compared by wide simulation experiments. In the paper, smallest job first scheduling strategy (SJF), and three communication patterns were measured, these are random, one-to-all, and all-to-all. The simulation results display that the performance of the non-contiguous Paging (0) is superior to that of wholly additional non-contiguous allocation algorithms (strategies) in our scenario due to their capability to reduce together (internal and external) fragmentation and reduction the dispute inside the network by persevere a huge contiguity degree through allocated processors.*

Keywords--- *Communication Patterns, k-ary n-cube Connected Multi Computers, Non-Contiguous Allocation, Fragmentation.*

I. INTRODUCTION

A Parallel computing can resolve larger difficulties, save time, and moderate a cost by spending several cheap calculating resources in place of paying for time on a supercomputer. Parallel computer is a processors gathering that mechanism with each other to solve a problematic. Parallel computer is doing several things at the same time; it can similarly solve the problems of restricted memory resources with traditional computers, and also it makes the system reliability superior where any disappointment or error in any of these processors will not loss wholly the system, but it will have been a small effect [4, 6, 11, 24, 25].

The communication of inter-processor can be classified on (shared and distributed) memory according to their memory architecture. In multiprocessors or what's famous as shared memory architecture, wholly processors are connected through shared memory, nevertheless with multicomputer or what's known as distributed memory architectures, every processor can arrival to its own local memory, and it can send or receive messages above the interconnection network. Multicomputer systems have developed very public in resolving large-scale computationally intensive problems in the latest years [4, 6, 11, 24, 25].

Networks of interconnection may be separated into (static and dynamic) networks. Dynamic networks usage switches that can be used to generate paths among processing elements. Dynamic networks are moreover famous as indirect networks [4, 6, 11, 25].

Doreyed Muhammed Ahmed Awaad Al-Kerboly, Mathematics Department, College of Education for Pure Science, University of Anbar, Anbar, Iraq. E-mail: doreyedm@yahoo.com, doreyedm@uoanbar.edu.iq

DOI: 10.37200/IJPR/V24I5/PR2020227
Received: 14 Mar 2020 | Revised: 30 Mar 2020 | Accepted: 06 Apr 2020

In static networks, every node is interconnected to his neighbors straight via wires (i.e. point-to-point connection). Static networks are additionally named direct networks [4, 6, 11, and 25].

Static networks are used so much in great-scale multi computers for the reason that their scalability by increasing channels and nodes depending on the pre-defined structure of network [4, 17, 25].

The k-ary n-cube network is one of the static networks (direct networks). K-ary n-cube is being one of the public networks designed for multi computers because of its properties, for example simplicity of execution besides ability to reduction latency of message over using locality of communication in numerous parallel applications [7, 8, 24]. Message latency includes of dual_ parts, the delay due to message transmission in addition to time a message live in if blocking occurs. These structure must an n-dimensional network structure and at all dimension, there are k-nodes; these nodes are linked to his neighbor nodes via straight channels. The best public examples of k-ary n-cubes are torus networks (n = 2 or 3) with wraparound links [4, 5, 7, 8, 24].

II. MOTIVATION

To the best of our knowledge, to hand has not been in the least study that compares to show of non-contiguous processor allocation for k-ary 3-cube established on synthetic workload models. In this paper have conducted a comparative study of the performance of non- continuous allocation algorithms proposed for k-ary n-cube connected multi-computers under synthetic workloads using extensive simulation. In this study, where three processor allocation algorithms have been considered, these are Random [15], Paging [15], and Multiple Partner [15] Allocation strategies with smallest job first scheduling strategy (SJF).

Preliminaries

The goal system is a k-ary n-cube (Q_k^n) multicomputer, where the network is mentioned to as kn. They have an n-dimensional grid structure and at each dimension, there are k-nodes; these nodes are connected to its neighbor nodes by direct channels. Allocation of processor and scheduling of job are critical to do the whole computational multi computers power by maximizing system utilization and minimizing the communication cost [4, 20, 23, 24].

Allocation of processor is accountable for allocating and choosing the group of processors that implement a parallel job, while scheduling of job is accountable for defining the directive in which jobs are selected for execution [3, 4, 13, 16, 20, 21, 23, 24]. Numerous allocation of processor algorithms has been suggested for k-ary n-cube multi computers. These algorithms may be separated (contiguous and non-contiguous) strategies. Contiguous allocation, a sub- cube of contiguous processors is allocated to a received job. This can disregard the contention between the messages of jobs being executed in the system and thus reduction inter-processor delays of communication. Nevertheless, contiguous allocation strategies are suffering from unlimited processor fragmentation, they may meaningfully reduce performance of system. Processor fragmentation may be separated into (internal and external) fragmentation. Therefore, non-contiguous allocation has been proposed to reduction processor fragmentation [1, 2, 4, 10, 12, 13, 23, 24].

Noncontiguous Allocation Algorithms: These strategies permit jobs to be implemented as soon as the amount of existing processors is enough [24]. Certain of these strategies that have been proposed in the literature are described lower.

K-ary n-cube Random allocation strategy: This strategy is a direct non-contiguous allocation algorithm in which a demand for a specified numeral of processors is satisfied with a numeral of processors chooses randomly. Together (internal and external) fragmentations are reduced, since wholly jobs are given exactly the demanded numeral of processors when existing. Because the condition of contiguity is not obligatory in this random allocation algorithm, the communication interference between jobs is probable to growth. A request for n processors are satisfied with n-randomly choose free processors [10, 15, 24, 27]. K-ary n-cube Paging allocation strategy: This strategy is statically separations the k-ary n-cube into k-ary m-cubes ($0 \leq m \leq n$), where m is the dimension of the demanded cube Qk . A demand for j processors is satisfied by allocating the first $\lceil j \rceil$ free m-dimensional partitions found in a linear scan of the allowed partitions, and this scan keeps some contiguity degree amongst allocated processors. A page is the unit of allocation, and its size, P size is equivalent to2m. The whole of allowable pages is greater or equal to the request, the allowed (free) pages are speed-read since the primary page till the wanted pages expanse is allocated. These algorithm is represented as paging (m) [4, 15, 24, 27]. The numeral of pages demanded by a size job job_size is calculated using the equation:

$$P\ request = \lceil job\_size / P\ size \rceil$$

In Paging allocation strategy, internal and external fragmentations are removed when $m = 0$ (i.e., page size is equal to one), on the other hand form ≥ 1, internal fragmentation occurs as a tradeoff for bigger degree of contiguity [10].

K-ary n-cube Multiple Partner strategy: This strategy, if there are enough processors in k-ary n-cube system, this allocation will succeed. A demand for r processors is factorized as explained in the request factoring procedure under, into the request array, in which Rm contains the numeral of sub-cubes in every dimension (m) that are required to satisfy the demand.

Every sub- cube, Qk fromRm e fruitfully allocated is frequently separated into k-requests for Qk s till they are wholly allocated [1, 2, 10].

Request Factoring Procedure: A job's demand for r processors is factorized into a demand for one or more k-ary m-cubes (base-k representation). In base-k, every one digit m ($0 \leq m \leq n$), represents the numeral of k-ary m-cubes of dimension m that are needed. These sub-cubes demand are located in a request array,Rm where Rm contains the numeral of k-ary m-cubes that the job demanded [10].

III. Simulation Results

In this paper, the conducted wide simulation tests so as to compare the non-contiguous allocation algorithms performance. I used Proc Simity, which is a tool of simulation that is developing “at the Oregon University for studying processor allocation and job scheduling in multi computers. The performance parameters measured are the average job turnaround time besides mean system utilization". The turnaround time of job is the time that the job occupies with the k-ary n-cube system beginning arrival to departure. The utilization of system is the ratio of processors utilized over the time [23, 24]. For low loads, the average turnaround times and mean system utilization are roughly the equivalent for all allocation algorithms measured. This's designed for the purpose that maximum of the processors in system in this cases are existing designed for allocation also allocation extremely likely to succeed.

Synthetic Workload Models

In this paper, the synthetic workload models used are based on models used in preceding research studies [4, 20, 22, 23, 26, and 27], wherever job arrival rate (load) follow an exponential distribution, and is definite as the job inter-arrival time inverse.

Job scheduling scheme is smallest job first. We limit ourselves to smallest job first scheduling (SJF) because smallest job first preserves fairness and also the purpose here is to compare the allocation algorithms [13, 17, 18, 19, 20, 23, 24, 26].

Allocated of processors to a job connect according to certain pattern of communication. Three communication patterns are used in these paper, (all-to-all, one-to-all, and random) communication patterns. With all-to-all communication, every one of the allocated of processors toward a job lead a message to wholly additional allocated of processors to the similar job. In one-to-all pattern of communication, a by chance a selection of processor drives a message to totally new allocated of processors to the similar job. On the other hand, the random communication pattern, a selection of processors randomly sends messages to randomly a selection of a group of processors allocated to the similar job [9, 13, 24].

The jobs execution times rely on the time required for flits to stand routed over node, sizes of packet, numeral of messages sent, contention of message and messages of distances go over [24]. Job remnants in system till a repetition of the pattern of communication is finished. The numeral of these messages is denoted as nummes[13].

The results shown below are for nummes= ni where n = 1 is used in this paper and it means one iteration of the communication pattern (i.e., job leftovers in system till the iteration of communication pattern is finished). This is used for all-to-all and one-to-all communication patterns, on the other hand with random communication pattern the nummes= nf where n = 1 which means the exact number of messages to be sent by all job [13], ts= 3 time units, and Plen= 8 flits.

Results of Synthetic Workload Models

The three figures below from (1 to 3) the mean utilization of system for the allocation strategies is planned against the system load for all-to-all, one-to-all and random communication patterns tested measured. It may be showed in these figures all the non-contiguous allocation strategy in our scenario are roughly equivalent with small different as described below with system utilization.

In figure 1, the values of mean system utilization succeeded in the job arrival rate equal to 0.1 jobs/time unit are around 99.84%, and 99.28%, used for the Paging (0), Multiple Partner, Random strategies, respectively. The Paging (0) algorithm performance is superior than the other algorithms in our scenario with smallest job first scheduling strategy(SJF). This is because the Paging (0) is like others non-contiguous allocation growths the probability of successful allocation for the reason that the allocation is continuously succeed if the amount of processors in the systems is bigger than or equivalent to the allocation request.

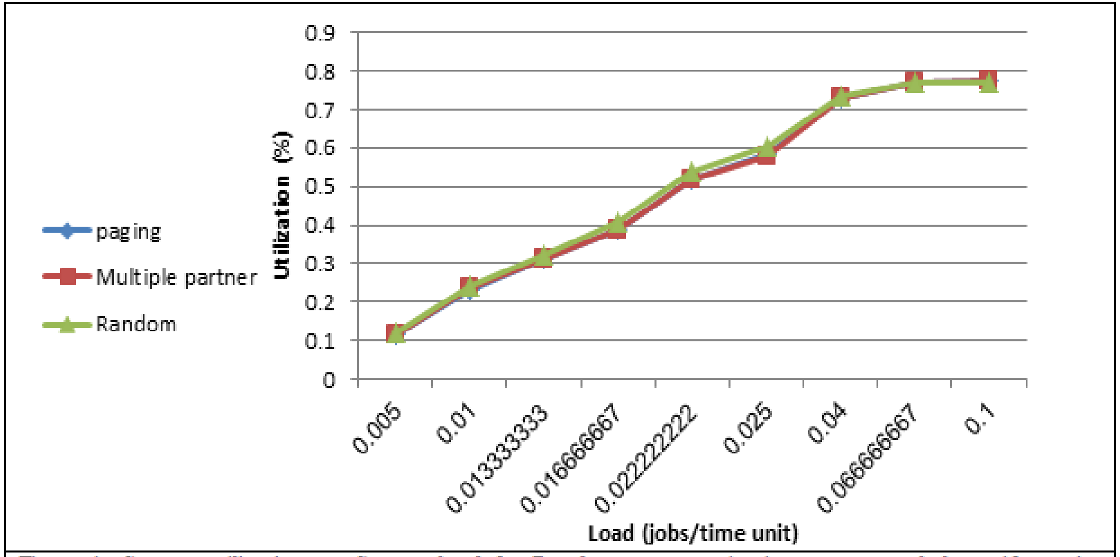

Figure 1: System utilization vs. System load for Random communication pattern and the uniform size distribution with smallest job first scheduling strategy

Figure 1: System Utilization vs System Load for All to All Communication Pattern and the Uniform Size Distribution with Smallest Job First Scheduling Strategy

In figure 2, the values of mean system utilization succeeded in the job arrival rate equal to

0.002 jobs/time unit are around 99.99%, and 99.91%, used for the Multiple Partner, Paging (0), Random strategies, respectively.

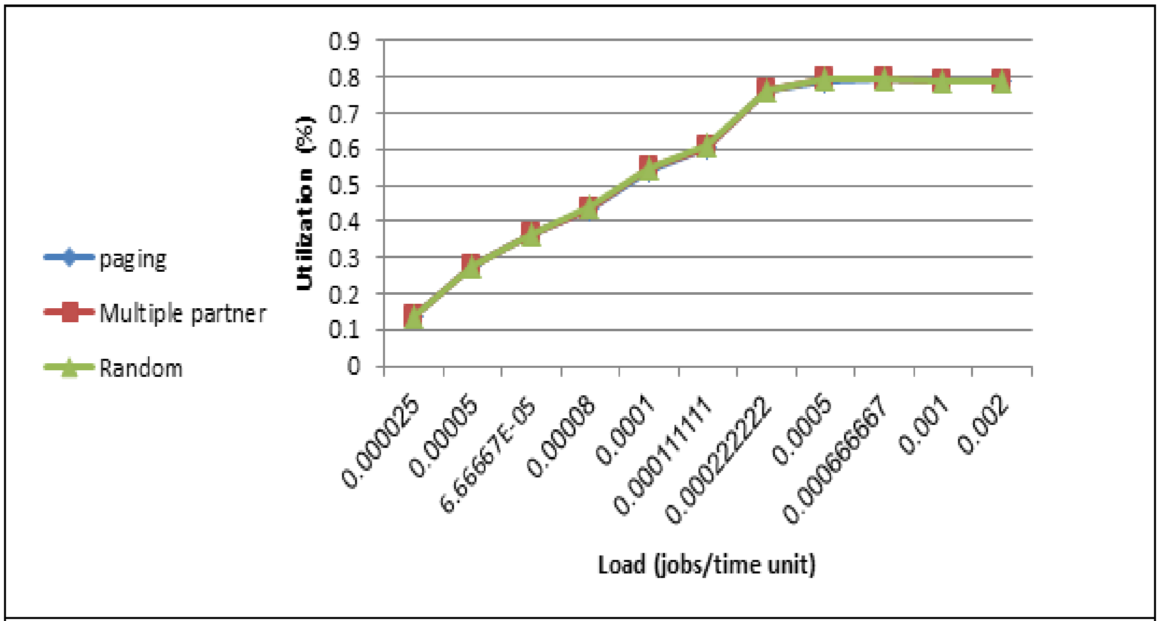

Figure 2: System utilization vs. system load for One to All communication pattern and the uniform size distribution with smallest job first scheduling strategy

Figure 2: System Utilization vs System Load for One to All Communication Pattern and the Uniform Size Distribution with Smallest Job First Scheduling Strategy

DOI: 10.37200/IJPR/V24I5/PR2020227
Received: 14 Mar 2020 | Revised: 30 Mar 2020 | Accepted: 06 Apr 2020

In figure 3, the values of mean system utilization succeeded in the job arrival rate equal to 0.00008 jobs/time unit are around 99.81%, and 97.56%, used for the Random, Multiple Partner, Paging (0) strategies, respectively.

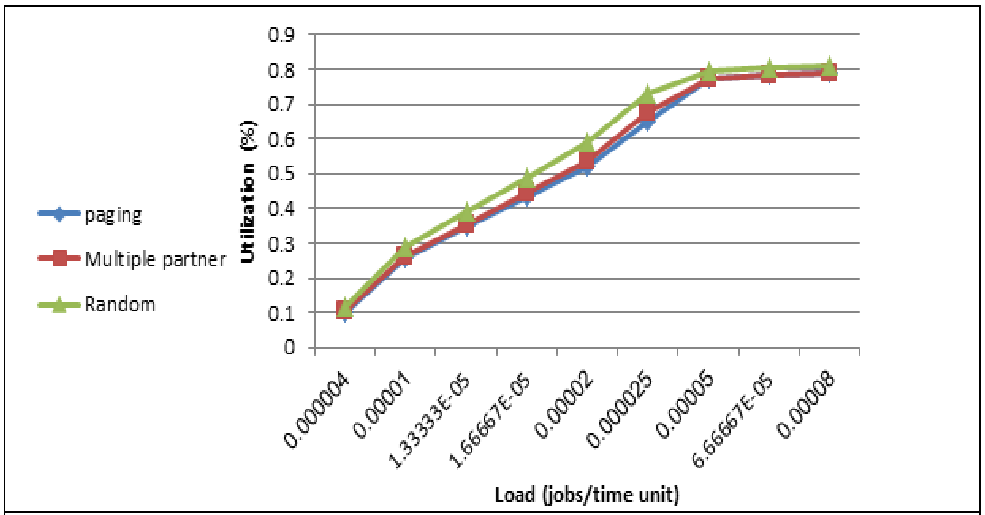

Figure 3: System utilization vs. system load for All to All communication pattern and the uniform size distribution with smallest job first scheduling strategy

Figure 3: System Utilization vs System Load for All to All Communication Pattern and the Uniform Size Distribution with Smallest Job First Scheduling Strategy

With figures 4, 5, 6, the average turnaround times of the allocation algorithms is planned compared to system load. It may be showed in figures that Paging (0) products the top results in exclusively cases with smallest job first scheduling algorithm (SJF) in this scenario. In figure (4), paging (0) average turnaround time is about 99.13%, and 91.98% of that of Multiple Partner, and Random algorithms, correspondingly under the job arrival_ rate of 0.1 jobs/time unit.

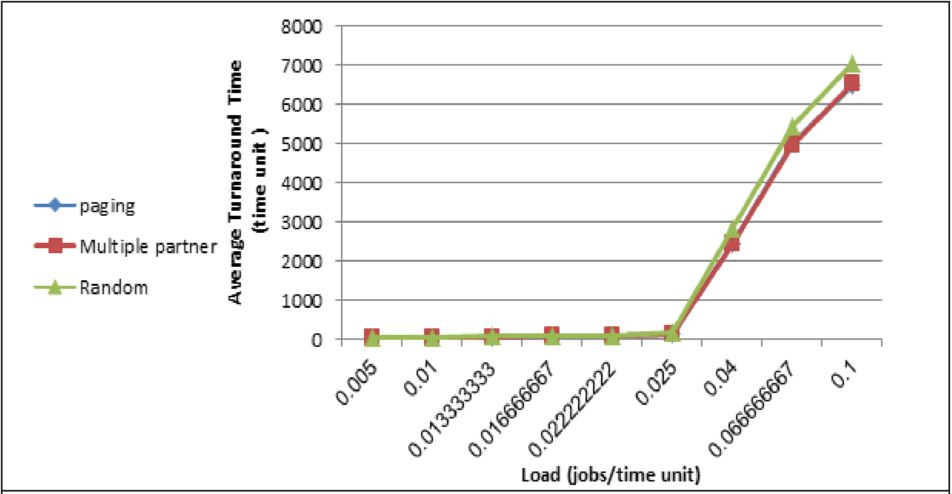

Figure 4: Average turnaround time vs. system load for Random communication pattern and the uniform size distribution with smallest job first scheduling strategy

Figure 4: Average Tumaround time vs System Load for Random Communication Pattern and the Uniform Size Distribution with Smallest Job First Scheduling Strategy

DOI: 10.37200/IJPR/V24I5/PR2020227
Received: 14 Mar 2020 | Revised: 30 Mar 2020 | Accepted: 06 Apr 2020

With figure 5, average turnaround time of paging (0) is around 99.77%, and 98.37% of that of Multiple Partner, and Random strategies, respectively under the job arrival rate of 0.002 jobs/time unit.

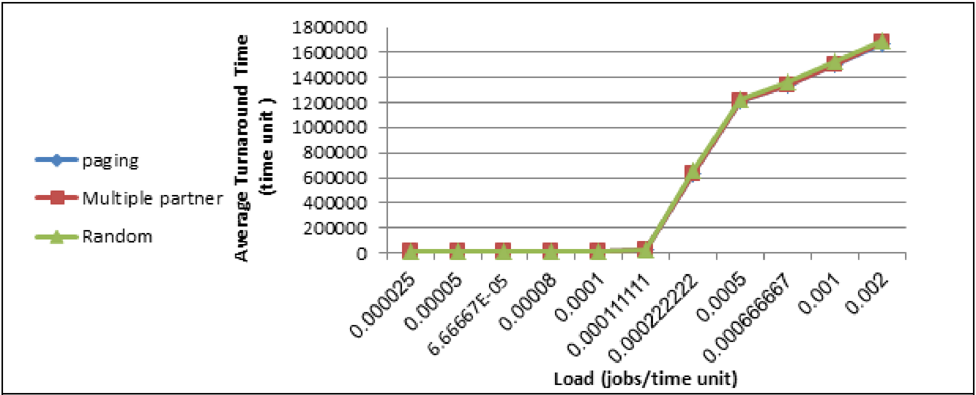

Figure 5: Average turnaround time vs. system load for One to All communication pattern and the uniform size distribution with smallest job first scheduling strategy

Figure 5: Average Tumaround Time vs System Load for One to All Communication Pattern and the Uniform Size Distribution with Smallest Job First Scheduling Strategy

With figure 6, average turnaround time of paging (0) is around 89.35%, and 83.75% of that of Multiple Partner, and Random strategies, respectively with the job arrival rate of 0.00008 jobs/time unit.

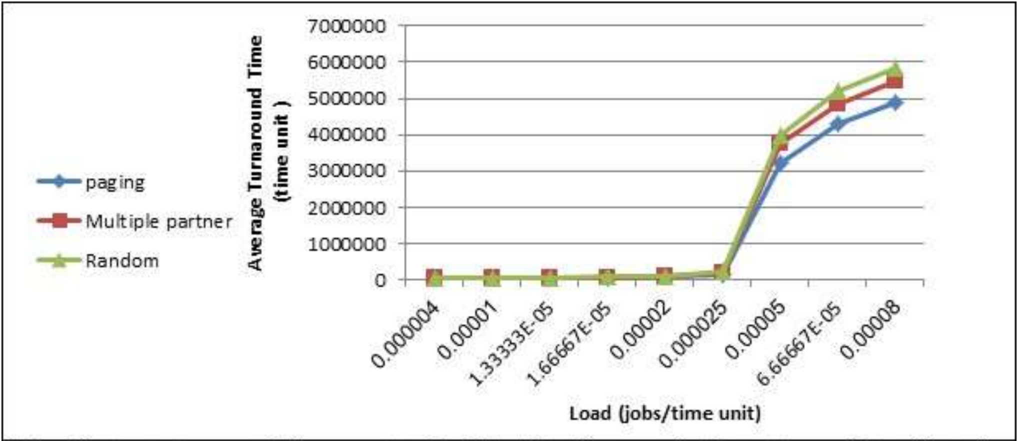

Figure 6: Average tumaround time vs. system load for All to All communication pattem and the uniform size distribution with smallest job first scheduling strategy

Figure 6: Average Tumaround Time vs System Load for All to All Communication Pattern and the Uniform Size Distribution with Smallest Job First Scheduling Strategy

The simulation run involves of 1000 completed jobs. An important self-determining variable in the simulation is the system load, it is formless as the reverse of the mean inter arrival time of jobs. It is sequences of values since little to hefty loads has been determined over testing with the Proc Simity allowing every allocation algorithm to arrive its upper limits of utilization" [4, 22].

IV. CONCLUSIONS

With this paper, I have conducted a comparative study of the performance non-contiguous allocation algorithms

(Paging (0), Multiple Partner, and Random) proposed for k-ary n-cube connected multi computers with smallest job first scheduling strategy (SJF).

I have showed wide simulation tests so as to compare the non-contiguous allocation algorithms performance considered in this paper. This paper deal with Proc Simity that is a simulation instrument which was technologically advanced at the University of Oregon. The average job turnaround time and mean system utilization performance are measured.

In the simulations, three communication patterns were measured, these are one-to-all, all- to-all, and random. In our scenario, the results of simulation show that the Paging (0) with smallest job first scheduling strategy (SJF) performance is superior to that of totally last strategies due to their ability to reduce together internal and external fragmentation and reduction the contention with the network by keeping a big degree of contiguity through allocated processors.

REFERENCES

[1] Abdullah Al-Dhelaan, Processor Allocation & Communication in Networks, Ph.D. Thesis, Department of Computer Science, Oregon State University, March 17, 1989.

[2] Abdullah. Al-Dhelaan and B. Bose. A New Strategy for Processors Allocation In An N- Cube Multiprocessor. *In Proceedings of the International Phoenix Conference on Computers and Communication*, pp. 114-118, March, 1989.

[3] C.-Y. Chang and P. Mohapatra, Performance Improvement of Allocation Schemes for Mesh-Connected Computers, *Journal of Parallel and Distributed Computing,* vol. 52, no. 1, pp. 40-68, 1998.

[4] Doreyed Muhammed Ahmed Awaad Al-kerboly, On the Execution of Different Job Scheduling Strategies for Non-Contiguous Allocation Algorithm in k-ary ncube Connected Multi computers, *Aus Journal* (Version on-line ISSN 0718-7262/Version impreasa ISSN 0718-204x) ,p208-214,2019.

[5] Hamid Mahini, and Hamid Sarbazi-Azad, Resource Placement in Three-Dimensional Tori, *Institute for Research in Fundamental Sciences (IPM),* Tehran, Iran vol. 35, No.10-11, pp. 535-543, 2009.

[6] I. Foster, Designing and Building Parallel Programs, Concepts and Tools for Parallel Software Engineering, Addison-Wesley, 3rd edition,1995.

[7] J. Al-Sadi, K. DAY, and M. Ould-Khaoua, A New Fault-Tolerant Routing Algorithm for K-Ary N-Cube Networks, *International Journal of High Speed Computing,* vol. 12, no. 1, pp. 29-54, 2004.

[8] Jacob Engel, Off-Chip Communications Architectures for High Throughput Network Processors, Ph.D. Thesis, Department of Computer Engineering, College of Engineering and Computer Science, University of Central Florida Orlando, Florida, 2005.

[9] Jos´e Duato, and Sudhakar Yalamanchili, Interconnection Networks: An Engineering Approach, Library of Congress Control Number: 2002104300, ISBN: 1-55860-852-4 Revised Printing, by Elsevier Science (USA), 2003.

[10] Kurt Windisch, Virginia and Bella Bosae, Contiguous and Non-Contiguous Processor Allocation Algorithms for K-Ary N-Cubes, *IEEE Transactions on Parallel and Distributed Systems,* vol.8, pp.712-726, 1995.

[11] M. Morris Mano, Computer System Architecture, Person Education Limited, 3rd edition, ISBN 0-13-175563-3, 1993.

[12] Ming-Syan Chen and Kang G. Shin, Processor Allocation In an N-Cube Multiprocessor Using Grey Code, *IEEE Transactions On Computers,* vol. C-36, no. 12, pp.1396-1407, December 1987.

[13] ProcSimity V 4.3 User's Manual, University of Oregon, 1997.

[14] Qian Ping Gu and Jun Gu, Algorithms and Average Time Bounds of Sorting on a Mesh- Connected Computer, *IEEE Transactions on Parallel and Distributed systems,* vol. 5, no. 3, pp. 308-315, March 1994.

[15] Raed Al Momani, and Ismail Ababneh, Communication Overhead in Non-Contiguous Processor Allocation Policies for 3D Mesh -Connected Multi computers, *The International Arab Jordan of Information Technology,* vol. 9, no. 2, pp. 133-141, March 2012.

[16] S. Attari and A. Isazadeh, Processor Allocation In Mesh Multiprocessors Using a Hybrid Method,

Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06), 0-7695-2736-1/06, pp. 492-496, 2006.

[17] 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 Multi computers *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.

[18] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh and Lewis M. Mackenzie, An Efficient Processor Allocation Strategy that Maintains a High Degree of Contiguity Among Processors in 2D Mesh Connected Multi computers, *2007 ACS/IEEE International Conference on Computer Systems and Applications* (AICCSA 2007), *IEEE computer Society Press, Philadelphia university,* Amman, Jordan, pp. 934-941, 13-16 May 2007.

[19] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh and Lewis M. Mackenzie, A Fast and Efficient Strategy for Sub-Mesh Allocation with Minimal Allocation Overhead in 3D Mesh Connected Multi computers, *Ubiquitous Computing and Communication Journal,* vol. 1, no. 1, pp. 26-36, 2006.

[20] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and Lewis M. Mackhenzie, Comparative Evaluation of Contiguous Allocation Strategies on 3D Mesh Multi computers, *Journal of System and Software,* vol. 82, no. 2, pp. 307-318, 2009.

[21] S. Moghaddam and M. Naghibzadeh, A New Processor Allocation Strategy Using ESS (Expanding Square Strategy), Proceedings of the 14th Euromicro *International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06),* 1066-6192/06, 2006.

[22] Saad Bani Mohammad, Ismail Ababneh, Mohamed Ould-Khaoua, A Comparative Study of Real Workload Traces and Synthetic Workload Models for Non-Contiguous Allocation in 2D Meshes, International Conference on Scalable Computing and Communications; *The Eighth International Conference on Embedded Computing,* pp.633-639, 2009.

[23] Saad Bani-Mohammad, Ismail Ababneh and Mazen Hamdan, Performance Evaluation of Noncontiguous Allocation Algorithms for 2D Mesh Interconnection Networks, *Journal of System and Software,* vol. 84, no.12, pp. 2156-2170, 2011.

[24] Saad O. Bani Mohammad, Efficient Processor Allocation Strategies for Mesh Connected Multi computers, PhD. *Thesis, Department of Computing Science, Faculty of Information and Mathematical Sciences, University of Glasgow,* February 2008.

[25] V. Kumar, A. Grama, A. Gupta, and G. Karypis, Introduction to Parallel Computing, Person Education Limited, 2nd edition, ISBN 0-201-64865-2, 2003.

[26] Virginia Lo, Jens Mache, and Kurt Windisch, A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling, *This research was sponsored by NSF grant MIP-*9108528. *Springer-Verlag Berlin Heidelberg,* pp. 25-46, 1998.

[27] Virginia Lo, Kurt J. Windisch, Wanqian Liu and Bill Nitzberg, Noncontiguous Processor Allocation Algorithms for Mesh-Connected Multi computers, *IEEE Transactions on Parallel and Distributed Systems,* vol. 8, no. 7, pp. 712-726, July 1997.