The Effect of Real Workloads and Stochastic Workloads on the Performance of Allocation and Scheduling Algorithms in 2D Mesh Multicomputers

S. Bani-Mohammad, M. Ould-Khaoua, and Lewis M. Mackenzie
Glasgow University, Computing Science, Glasgow G12 8RZ, UK.
{saad, mohamed, lewis}@dcs.gla.ac.uk

I. Ababneh
University of Science and Technology, Computing Science, Irbid, Jordan.
ismael@just.edu.jo

J. D. Ferguson
Strathclyde University, Computer and Information Sciences, Glasgow G1 1XH, UK.
john.ferguson@cis.strath.ac.uk

Abstract

The performance of the existing non-contiguous processor allocation strategies has been traditionally carried out by means of simulation based on a stochastic workload model to generate a stream of incoming jobs. To validate the performance of the existing algorithms, there has been a need to evaluate the algorithms' performance based on a real workload trace. In this paper, we evaluate the performance of several well-known processor allocation and job scheduling strategies based on a real workload trace and compare the results against those obtained from using a stochastic workload. Our results reveal that the conclusions reached on the relative performance merits of the allocation strategies when a real workload trace is used are in general compatible with those obtained when a stochastic workload is used.

1. Introduction

Efficient processor allocation and job scheduling are critical if the full computational power of large-scale multicomputers is to be harnessed effectively [2, 5]. Processor allocation is responsible for selecting the set of processors on which parallel jobs are executed whereas job scheduling is responsible for determining the order in which the jobs are executed [2, 5, 12].

Most allocation strategies employed in a multicomputer are based on contiguous allocation, where the processors allocated to a parallel job are physically contiguous and have the same topology as that of the interconnection network of the multicomputer [2, 5, 19]. Contiguous strategies often result in high external processor fragmentation, as has been shown in [19]. External fragmentation occurs when a sufficient number of free processors are available to satisfy a job request but they are not allocated to it because they are not contiguous.

Several studies have attempted to reduce external fragmentation [4, 14, 17]. One suggested solution is to adopt non-contiguous allocation [4, 12, 17]. In non-contiguous allocation, a job can execute on multiple disjoint smaller sub-networks rather than always waiting until a single sub-network of the requested size and shape is available. Although non-contiguous allocation increases message contention in the network, lifting the contiguity condition is expected to reduce processor fragmentation and increase processor utilization [4, 12, 17]. It is the introduction of wormhole switching [1, 15] that has lead researchers to consider non-contiguous allocation on multicomputer networks with a long communication distances, such as the 2D mesh [1, 4, 12, 17]. This is due to the fact that one of main advantages of wormhole switching over earlier switching techniques, e.g. store-and-forward, is that message latency depends less on the distance the message travels from source to destination.

Most existing research on processor allocation has been carried out in the context of the 2D mesh [2, 4, 5, 8, 14, 17, 19]. The mesh has been one of the most common networks for multicomputers due to its simplicity, scalability, structural regularity, and ease of implementation [2, 5]. It has been used as the underlying network in a number of practical and experimental parallel machines, such as iWARP [3], the IBM BlueGene/L [18], and Delta Touchstone [6].

Most existing allocation strategies employed in a multicomputer have been evaluated by means of

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

simulation using a stochastic workload [2, 4, 5, 8, 12, 13, 14, 17, 19] based mostly on probabilistic distributions. In this workload, the researchers have used exponential distribution to generate inter-arrival times. Job sizes (i.e., the number of processors requested) have been generated using a variety of probabilistic distributions including uniform, uniform increasing, uniform decreasing, and exponential distributions, while jobs execution times have been generated with exponential distribution.

In this paper, the performance of the existing non-contiguous allocation strategies is evaluated using a real workload trace and the results are compared against those obtained through a stochastic workload. A real workload trace is a record of execution of parallel jobs submitted to run on a practical parallel machine in which each job arrives in the system, requests a particular sized partition of the system's processors and executes on the partition for a period of time. Although messages from other jobs may pass through the new partition, the new job holds the processors in this partition exclusively until it finishes running. At this time, the job departs the system and its processors are freed for use by another incoming job [11, 16]. The partitions requested by the jobs typically include job size and the execution time. Moreover, each job in the workload is associated with an arrival time, indicating when it is submitted to the scheduler for consideration. A real workload trace can potentially provide a very high level of realism when used directly in performance evaluation experiments [16]. Our results reveal that the conclusions reached on the relative performance merits of the allocation strategies when a real workload trace is used are compatible with those obtained when a stochastic workload is used.

The rest of the paper is organized as follows. Section 2 provides some preliminaries. Section 3 contains a brief overview of the non-contiguous allocation strategies considered in this study while Section 4 contains a brief overview of the scheduling strategies. Section 5 presents simulation results. Finally, Section 6 concludes this paper.

2. Preliminaries

The target system is a $W \times L$ 2D mesh, where W is the width of the mesh and L is its length. Every processor is denoted by a pair of coordinates (x, y), where $0 \le x < W$ and $0 \le y < L$ [12]. Each processor is connected by bidirectional communication links to its neighbour processors, as depicted in Fig. 1. This figure shows an example of a 4 × 4 2D mesh, where allocated processors are denoted by shaded circles and free processors are denoted by white circles. If a job requests the allocation of sub-mesh of size 2 × 2 contiguous allocation fails because no 2 × 2 sub-mesh of free processors is available, however the 4 free processors can be allocated to the job if allocation is non-contiguous. The following definitions have been adopted from [12].

***Definition 1**: A sub-mesh $S(w,l)$ of width w and length l, where $0 < w \le W$ and $0 < l \le L$ is specified by the coordinates (x, y, x', y'), where (x, y) are the coordinates of the base of the allocated sub-mesh and (x', y') are the coordinates of its end. For example (0, 0, 2, 1) represents the 3 ×2 sub-mesh S in Fig. 1. The base node of the sub-mesh is (0, 0), and its end node is (2, 1). The size of $S(w,l)$ is $w \times l$ processors.*

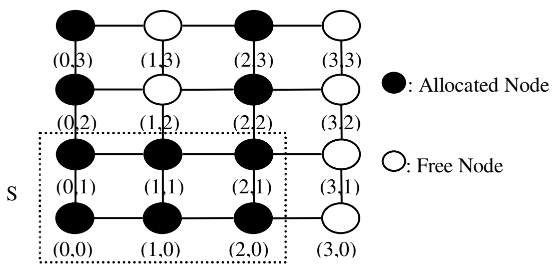

Figure 1. An example of a 4×4 2D mesh

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

***Definition 3**: A free sub-mesh is one whose processors are all not allocated.*

***Definition 4**: A suitable sub-mesh $S(w,l)$ is a free sub-mesh that satisfies the conditions: $w \ge a$ and $l \ge b$ assuming that the allocation of $S(a,b)$ is requested.*

In this study, it is assumed that parallel jobs are selected for allocation and execution using the First-Come-First-Served (FCFS) and Shortest-Service-Demand (SSD) (i.e., shortest execution times) scheduling strategies. FCFS is chosen because it is fair and it is widely used in other similar studies [4, 5, 17, 19], while SSD is adopted because it is expected to reduce performance loss due to FCFS blocking [10].

3. Non-contiguous Allocation Strategies

Advances in switching techniques such as wormhole switching [15], have made communication latency less sensitive to the distance between communicating nodes [4]. This has made allocating a job to non-contiguous processors plausible in longdiameters networks, such the 2D mesh. Non-contiguous allocation allows jobs to be executed without waiting if the number of available processors is sufficient [4, 17]. Below, we describe some non-contiguous allocation strategies that have been proposed for the 2D mesh.

Paging: In the Paging strategy [17], the entire 2D mesh is divided into pages that are sub-meshes with equal sides' length of 2^{size_index}, where $size\_index$ is a positive integer. A page is the allocation unit. The pages are indexed according to several indexing schemes (row-major, shuffled row-major, snake-like, and shuffled snake-like indexing). A paging strategy is denoted as Paging($size\_index$). For example, Paging(2) means that the pages are 4 × 4 sub-mesh. In this paper, we only consider the row-major indexing scheme because using the remaining indexing schemes has only a slight impact on the performance of Paging, as revealed in [17]. In Paging, there is some degree of contiguity because of the indexing schemes used. Contiguity can also be increased by increasing the parameter $size\_index$. However, there is internal processor fragmentation for $size\_index \geq 1$, and it increases with $size\_index$ [17].

Multiple Buddy Strategy (MBS): In MBS [17], the mesh is divided into non-overlapping square sub-meshes with side lengths equal to the powers of two upon initialization. The number of processors, p, requested by an incoming job is factorized into a base-4 representation of the form: $\sum_{i=0}^{\lfloor \log_4 p \rfloor} d_i \times \left(2^i \times 2^i\right)$, where $0 \leq d_i \leq 3$. The request is then considered for allocation according to the factorized number, where d_i blocks of size $2^i \times 2^i$ are required. If a required block is unavailable, MBS searches for a larger block and breaks it down into 4 buddies until it produces blocks of the desired size. If that fails, the requested block is broken into 4 requests for smaller blocks and repeats the allocation process [17]. An issue with MBS is that it may fail to allocate a contiguous sub-mesh although one exists. In fact, contiguous allocation is explicitly sought in MBS only for requests with sizes of the form 2^{2n}, where n is a positive integer.

Greedy Available Busy List (GABL): GABL combines the desirable features of both contiguous and non-contiguous allocation, and partitions requests based on the sub-meshes available for allocation [12]. In GABL [12], when a parallel job is selected for allocation a sub-mesh suitable for the entire job is searched for. If such a sub-mesh is found it is allocated to the job and the allocation is done. Otherwise, the largest free sub-mesh that can fit inside $S(a,b)$ is allocated. Then, the largest free sub-mesh whose side lengths do not exceed the corresponding side lengths of the previously allocated sub-mesh is searched for under the constraint that the number of processors allocated does not exceed $a \times b$. This last step is repeated until $a \times b$ processors are allocated. Allocated sub-meshes are kept in a busy list. When a job departs the sub-meshes it is allocated are removed from the busy list and the number of free processors is updated. Note that allocation always succeeds if the number of free processors is $\geq a \times b$. Moreover, it can be noticed that the methodology used for maintaining contiguity is greedy. GABL tries to allocate large sub-meshes first.

4. Job Scheduling Strategies

The order in which jobs are scheduled can have considerable effect on system performance [10, 13]. In this paper, we consider the FCFS and SSD scheduling strategies. In FCFS, the allocation request that arrives first is considered for allocation first. Allocation attempts stop when they fail for the current FIFO queue head. SSD considers the shortest job to be the one having the shortest processor service demand [10, 13].

The performance of non-contiguous allocation can be significantly affected by the type of the scheduling strategy used. To illustrate this, the performance of the allocation strategies considered in this study has been evaluated under both FCFS and SSD scheduling strategies. The results show that SSD is better than FCFS; therefore, the scheduling and allocation strategies have substantial effect on the performance of the non-contiguous allocation in 2D mesh.

5. Simulation Results

Extensive simulation experiments have been carried out to compare the performance of the allocation strategies. We have implemented the allocation and deallocation algorithms, including the busy list routines, in the C language, and integrated the software into the ProcSimity simulation tool for processor allocation and job scheduling in highly parallel systems. ProcSimity has been used to investigate the processor allocation problems, such as fragmentation and communication overhead problems [11].

In the results shown below, we model a 16 × 22 mesh. This size was selected to closely match the size of the partition which generated the trace. Jobs are served on the FCFS and SSD scheduling strategies. The interconnection network uses wormhole switching. Flits are assumed to take one time unit to move between two adjacent nodes, and t_s time units to be routed through a node. A flit is the smallest unit of data transmission. Message sizes are represented by P_{len}. Processors allocated to a job communicate with each other using the all-to-all communication pattern [8, 15, 17]. In all-to-all communication, each processor allocated to a job sends a packet to all other processors allocated to the same job. This communication pattern is considered because it causes much message collision and it is known as the weak point for non-contiguous allocation [8]. As in [17], the number of messages that are actually generated by a given job is exponentially distributed with a mean num_mes.

Unless specified otherwise, the performance figures shown below are for a 16 × 22 mesh, $t_s = 3$ time units, $P_{len} = 8$ flits and $num\_mes = 5$ packets. The values we use for t_s and P_{len} were recommended in [11]. The main performance parameters used are the average turnaround time of jobs, average service time, average packet latency, average packet blocking time, and mean system utilization. The turnaround time of a job is the time that the job spends in the mesh from arrival to departure. The average service time is the average time it takes for jobs to execute once allocated to processors [11]. The average packet latency is the average time for message packets to reach their destination once they are injected into the network [11]. The average packet blocking time is the average time that message packets spend blocked in network buffers, waiting for access to their next channel [11]. The mean system utilization is the percentage of processors that are utilized over time. An important independent variable in the simulation is the system load, defined as the inverse of the mean inter-arrival time of jobs. Its values for the various simulation points were determined through experimentation with the simulator.

As in [7], to allow for both easy comparison with previous experiments in [12] and for a realistic evaluation of allocation strategies, our performance evaluation includes the use of two different workloads. The first workload is a stochastic workload. In this workload, jobs are assumed to have exponential inter-arrival times, the execution times of jobs are not simulator inputs. They are determined by the simulator and their values depend on t_s, P_{len}, the number of messages sent, message contention, and distances messages traverse. Two distributions are used to generate the lengths and widths of job requests. The first is the uniform distribution over [1, W] for the width of the requested sub-mesh and over [1, L] for its length, where the width and length of a request are generated independently. The second distribution is the exponential distribution, where the width and length of job requests are exponentially distributed with a mean of half the side width and length of the entire mesh. These distributions have often been used in the literature [4, 5, 17, 19]. Each simulation run consists of 1000 completed jobs. Simulation results are averaged over enough independent runs so that the confidence level is 95% and the relative errors do not exceed 5%.

The second workload is a real workload trace, a stream of 10658 real production jobs from the Intel Paragon at the San Diego Supercomputer Centre. The traced job stream is taken only from the 352 nodes [7, 9]. The workload trace had the following statistical characteristics: the mean interarrival time was 1186.7 seconds; the average job size was 34.5 nodes, and with the distribution favouring sizes that are non-powers of two. Our real workload trace uses the arrival times, job execution times and job sizes. As in [7], to challenge allocation strategies, we multiply job arrival times by a constant factor f. When $f < 1$, the interarrival times decrease, resulting in an increased system load.

The notation <allocation strategy>(<scheduling strategy>) represents the strategies in the performance figures. For example, GABL(SSD) refers to the Greedy Available Busy List allocation strategy under the Shortest-Service-Demand scheduling strategy.

In Figs. 2, 3, and 4, the average turnaround time of jobs are plotted against the system load for the all-to-all communication pattern under both FCFS and SSD. It can be seen from these figures that the results from a stochastic workload and a real workload ranked the scheduling and allocation algorithms in the same order from best to worst in terms of performance parameter used in these figures except that the performance of MBS based on a real workload is inferior to that of Paging(0) because the size of the sub-meshes requested by jobs in this workload is non-power of two and the contiguous allocation is explicitly sought in MBS only for requests with sizes of the form 2^{2n}, where n is a positive integer. The results reveal that GABL performs better than all other allocation strategies for both workloads and scheduling strategies considered in this paper. In Fig. 2, for example, the difference in performance in favour for GABL(FCFS) could be as large as 33% to Paging(0)(FCFS), and 68% to MBS(FCFS) under the job arrival rate 0.004 jobs/time unit. Moreover, the results have shown that the effects of the SSD scheduling strategy on the performance of the allocation strategies are better than that of the FCFS scheduling strategy under both workloads.

In Figs. 5, 6, and 7, the average service time of jobs are plotted against the system load for the all-to-all communication pattern under both the FCFS and SSD scheduling strategies. Again, the results from both workloads ranked the scheduling and allocation algorithms in the same order from best to worst in terms of performance parameter used in these figures; GABL performs much better than all other allocation strategies for both workloads considered in this paper. Fig. 6, for example, depicts that when the job arrival rate is 0.0305 jobs/time unit, the average service times of GABL(FCFS) are 67% and 76% of that of Paging(0)(FCFS) and MBS(FCFS), respectively.

Figs. 8, 9, and 10 depict the mean system utilization of the allocation strategies for the all-to-all communication pattern and both scheduling strategies considered in this paper. The simulation results in these figures are presented for a heavy system load. The load is such that the waiting queue is filled very early, allowing each strategy to reach its upper limits of utilization. For both workloads considered in this paper, the non-contiguous allocation strategies achieve a mean system utilization of 72% to 89%. In each of these workloads, the utilization of the three non-contiguous strategies is approximately the same. This is because the allocation strategies, considered in this paper, have the same ability to eliminate both internal and external processor fragmentation. They always succeed to allocate processors to a job when the number of free processors is greater than or equal the allocation request.

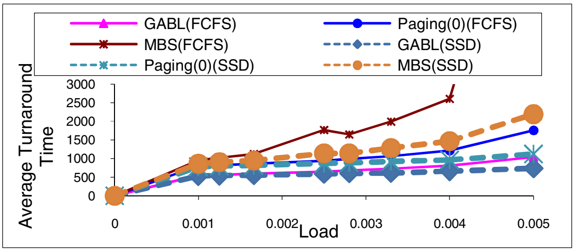

Figure 2. Turnaround time vs. system load for all-to-all communication pattern and a real workload in a 16 × 22 mesh.

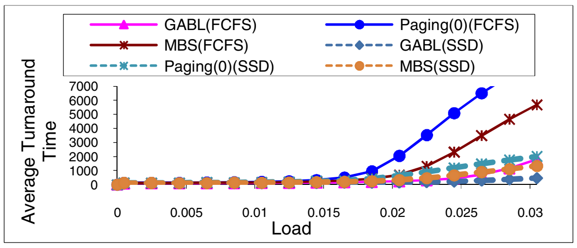

Figure 3. Turnaround time vs. system load for all-to-all communication pattern and a stochastic workload based on uniform side lengths distribution in a 16 × 22 mesh.

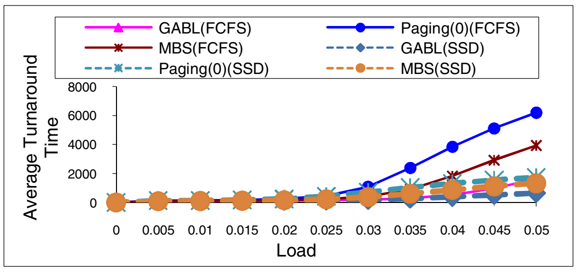

Figure 4. Turnaround time vs. system load for all-to-all communication pattern and a stochastic workload based on exponential side lengths distribution in a 16 × 22 mesh.

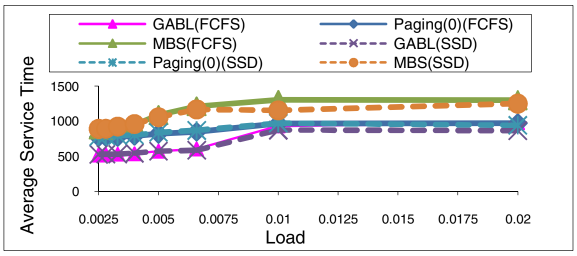

Figure 5. Service time vs. system load for all-to-all communication pattern and a real workload in a 16 × 22 mesh.

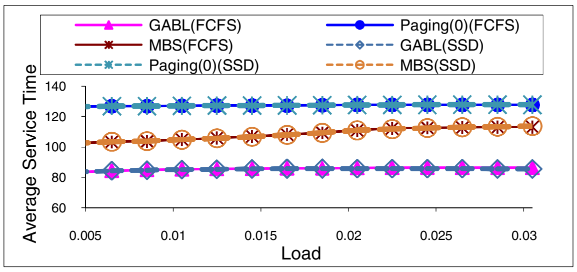

Figure 6. Service time vs. system load for all-to-all communication pattern and a stochastic workload based on uniform side lengths distribution in a 16 × 22 mesh.

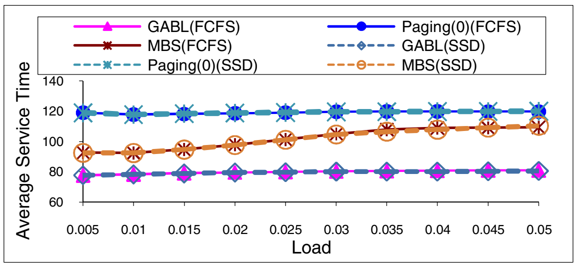

Figure 7. Service time vs. system load for all-to-all communication pattern and a stochastic workload based on exponential side lengths distribution in a 16 × 22 mesh.

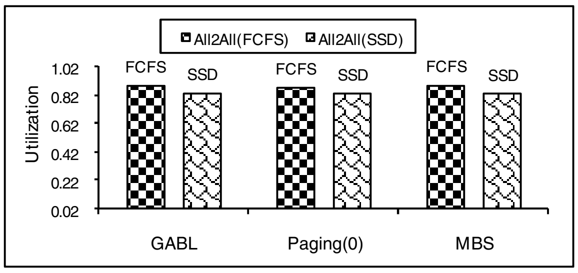

Figure 8. System utilization vs. system load for all-to-all communication pattern and a real workload in a 16 × 22 mesh.

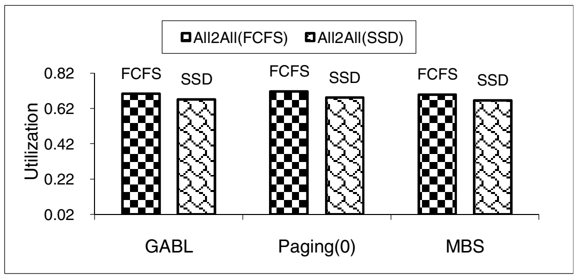

Figure 9. System utilization vs. system load for all-to-all communication pattern and a stochastic workload based on uniform side lengths distribution in a 16 × 22 mesh.

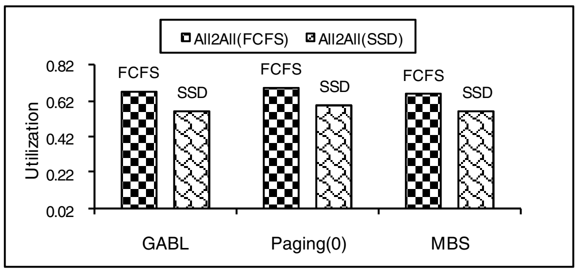

Figure 10. System utilization vs. system load for all-to-all communication pattern and a stochastic workload based on exponential side lengths distribution in a 16 × 22 mesh.

In addition to the performance parameters above, we have measured another two parameters for the non-contiguous allocation strategies. These are the average packet blocking time and the average packet latency computed for all jobs. Packet blocking time is used to measure the contention in the interconnection network. Messages originate from a processor element and their flits traverse the network in pipeline fashion to their destination processor. If the header flit of a packet is routed to a busy channel, that header flit and its trailing flits stop moving and block whichever channels they occupy in the network [17]. This results in packet blocking time, due to contention, which can be measured in the simulation. The non-contiguous allocation introduces potential problems due to message contention because the messages occupy more links, yielding potential communication interference amongst jobs. Therefore, the successful allocation strategy is the strategy that has lower packet latency.

In Figs. 11, 12, and 13, the average packet blocking time is plotted against the system load for the all-to-all communication pattern under FCFS and SSD scheduling strategies. The results of the experiments from a real workload and a stochastic workload ranked the scheduling and allocation algorithms in the same order from best to worst in terms of performance parameter used in these figures; GABL has better packet blocking time than the remaining strategies for all loads. In Fig. 11, for example, the average packet blocking times of GABL(SSD) are 81% and 51% of that of Paging(0)(SSD) and MBS(SSD), respectively, when the job arrival rate is 0.02 jobs/time unit.

In Figs. 14, 15, and 16, the average packet latency is plotted against the system load for the all-to-all communication pattern and the scheduling strategies FCFS and SSD. The results from both workloads ranked the scheduling and allocation algorithms in the same order from best to worst in terms of performance parameter used in these figures; GABL has lower packet latency than all other strategies for both workloads in this study. Fig. 14, for example, depicts that when the job arrival rate is 0.02 jobs/time unit, the average packet latency of GABL(FCFS) is 84% and 54% of that of Paging(0)(FCFS) and MBS(FCFS), respectively. In Fig. 16, the average packet latency of GABL(FCFS) is 66% and 71% of that of Paging(0)(FCFS) and MBS(FCFS), respectively, when the job arrival rate is 0.05 jobs/time units.

To sum up, the above performance results demonstrate that both a real workload and a stochastic workload gave the same ranking of the allocation strategies from best to worst.

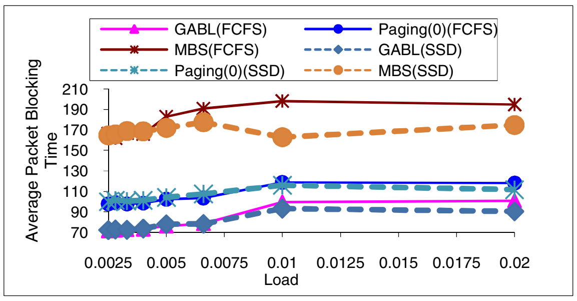

Figure 11. Packet blocking time vs. system load for all-to-all communication pattern and a real workload in a 16 × 22 mesh.

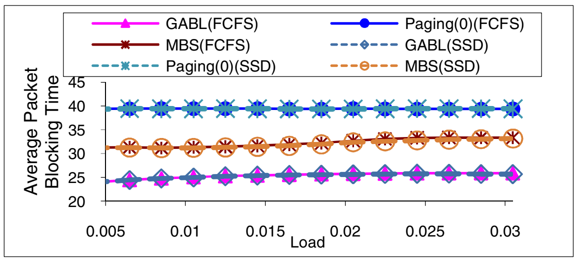

Figure 12. Packet blocking time vs. system load for all-to-all communication pattern and a stochastic workload based on uniform side lengths distribution in a 16 × 22 mesh.

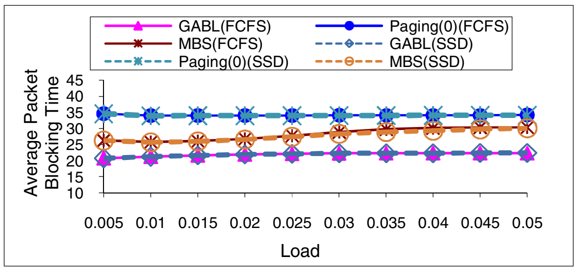

Figure 13. Packet blocking time vs. system load for all-to-all communication pattern and a stochastic workload based on exponential side lengths distribution in a 16 × 22 mesh.

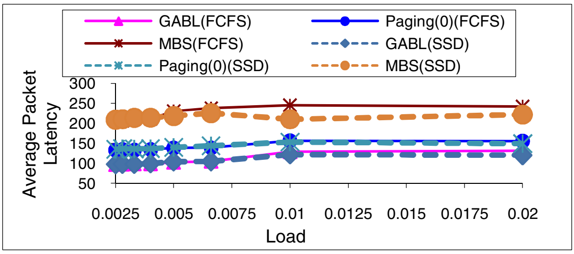

Figure 14. Packet latency vs. system load for all-to-all communication pattern and a real workload in a 16 × 22 mesh.

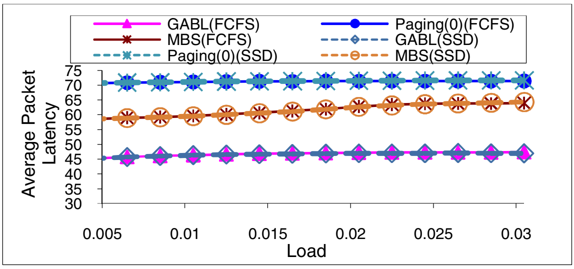

Figure 15. Packet latency vs. system load for all-to-all communication pattern and a stochastic workload based on uniform side lengths distribution in a 16 × 22 mesh.

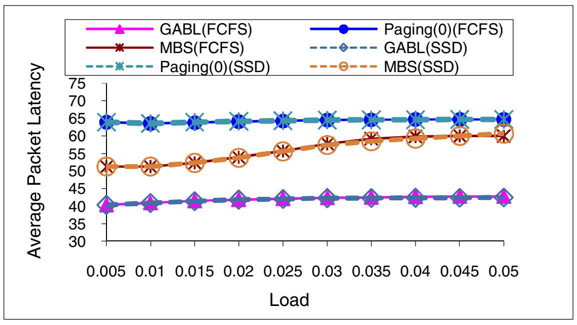

Figure 16. Packet latency vs. system load for all-to-all communication pattern and a stochastic workload based on exponential side lengths distribution in a 16 × 22 mesh.

6. Conclusion and Future Directions

This paper has investigated the impact of real workload traces and stochastic workloads on the performance of the non-contiguous allocation strategies that have been proposed for 2D mesh connected multicomputers. These strategies cover a wide range of choices, including Paging strategy (Paging(0)), Multiple Buddy Strategy (MBS), and Greedy Available Busy List strategy (GABL). GABL differs from the other allocation strategies in maintaining a high degree of contiguity among processors allocated to a job which decreases the number of sub-meshes allocated to a job, hence the distance traversed by messages is decreased, and which in turn decreases the communication overhead. GABL achieves this by using a busy list whose length is often small even when the size of the mesh scales up.

Simulation results have shown that the relative performance of the non-contiguous allocation strategies has not been significantly affected by the choice of a workload. In most experiments a real workload and a stochastic workload ranked the performance of the non-contiguous allocation algorithms in the same order from best to worst in terms of performance parameters used in this study. Moreover, the simulation results have shown that the effects of the SSD scheduling strategy on the performance of the allocation strategies are better than that of the FCFS scheduling strategy in terms of average turnaround time.

As a continuation of this research in the future, it would be interesting to assess the performance of the allocation strategies on other common multicomputer networks, such as torus networks. Another possible direction for future research is to implement the allocation strategies based on other real workload traces from different parallel machines and compare it with our results by means of simulation.

7. Acknowledgment

We would like to thank Prof. Dror Feitelson, School of Computer Science and Engineering, Hebrew University, who provided us workload traces and information regarding the traces. Also, we would like to thank Dr. Eitan Frachtenberg, Hebrew University, who helped us to get in touch with Prof. Feitelson regarding the traces.

8. References

[1] A. Al-Dubai, M. Ould-Khaoua, K. El-Zayyat, I. Ababneh, S. Al-Dobai, "Towards scalable collective communication for multicomputer interconnection networks", *Journal of Information Sciences*, *vol. 163, no. 4*, Elsevier Science, USA, 2004, pp. 293-306.

[2] B.-S.Yoo, C.-R. Das, "A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers", *IEEE Transactions on Parallel & Distributed Systems, vol. 51, no. 1*, IEEE Computer Society, Washington, USA, January 2002, pp. 46-60.

[3] C. Peterson, J. Sutton, P. Wiley, "iWARP: a 100-MPOS, LIW microprocessor for multicomputers", *IEEE Micro*, *vol. 11, no. 3*, IEEE Computer Society, CA, USA, 1991, pp. 26-29, 81-87.

[4] C.-Y. Chang, P. Mohapatra, "Performance improvement of allocation schemes for mesh-connected computers", *Journal of Parallel and Distributed Computing*, *vol. 52, no. 1*, Academic Press, Inc. Orlando, FL, USA, July 1998, pp. 40-68.

[5] I. Ababneh, "An efficient free-list submesh allocation scheme for two-dimensional mesh-connected multicomputers", *Journal of Systems and Software*, *vol. 79, no. 8*, Elsevier Science Inc., New York, NY, USA, August 2006, pp. 1168-1179.

[6] Intel Corporation, *A Touchstone DELTA system description*, 1991.

[7] 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*, IEEE Computer Society, 1997, pp. 120–124.

[8] K. Suzaki, H. Tanuma, S. Hirano, Y. Ichisugi, C. Connelly, and M. Tsukamoto, "Multi-tasking Method on Parallel Computers which Combines a Contiguous and Non-contiguous Processor Partitioning Algorithm", *Proceedings of the Third International Workshop on Applied Parallel Computing, Industrial Computation and Optimization,* Springer-Verlag, UK, 1996, pp. 641-650.

[9] K. Windisch, V. Lo, D. Deitelson, B. Nitzberg, and R. Moore, "A comparison of workload traces from two production parallel machines", *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation (Frontiers'96),* IEEE Computer Society Press, Annapolis, MD, USA, Oct. 1996, pp. 319-326.

[10] P. Krueger, T. Lai, V. A. Radiya, "Job scheduling is more important than processor allocation for hypercube computers", *IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 5*, IEEE Press, Piscataway, NJ, USA, May 1994, pp. 488-497.

[11] ProcSimity V4.3 User's Manual, University of Oregon, 1997.

[12] S. Bani-Mohammad, M. Ould-Khaoua, and I. Ababneh, A new processor allocation strategy with a high degree of contiguity in mesh-connected multicomputers, *Simulation Modelling Practice and Theory*, *vol. 15, no. 4*, Elsevier Science, April 2007, pp. 465-480.

[13] 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, Phiadelphia University, Amman, Jordan, 13-16 May 2007, pp. 934-941.

[14] T. Srinivasan, J. Seshadri, A. Chandrasekhar, J. Jonathan, "A Minimal Fragmentation Algorithm for Task Allocation in Mesh-Connected Multicomputers", *Proceedings of IEEE International Conference on Advances in Intelligent Systems – Theory and Applications – AISTA 2004 in conjunction with IEEE Computer Society, ISBN 2-9599-7768-8,* IEEE Press, Luxembourg, Western Europe, 15-18 Nov 2004.

[15] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction To Parallel Computing*, The Benjamin/Cummings publishing Company, Inc., Redwood City, California, 2003.

[16] V. Lo, J. Mache, and K. Windisch, "A Comparative Study of Real Workload Traces and Synthetic Workload Models for Parallel Job Scheduling", *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'98), ISBN: 3-540-64825-9*, Springer-Verlang, Berlin Heidelberg, 1998, pp. 25-46.

[17] V. Lo, K. Windisch, W. Liu, and B. Nitzberg, "Non-contiguous processor allocation algorithms for mesh-connected multicomputers", *IEEE Transactions on Parallel and Distributed Systems*, *vol. 8, no. 7*, IEEE Press, Piscataway, NJ, USA, July 1997, pp. 712-726.

[18] Y. Aridor, T. Domany, O. Goldshmidt, Y. Kliteynik, J. Moreira, and E. Shmueli, "Open Job Management Architecture for the Blue gene/L Supercomputer", *Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'05),* Springer Berlin / Heidelberg, Cambridge, MA, June 19, 2005, pp. 91-107.

[19] Y. Zhu, "Efficient processor allocation strategies for mesh-connected parallel computers", *Journal of Parallel and Distributed Computing, vol. 16, no. 4*, Elsevier, San Diego, CA, USA, 1992, pp. 328-337.