On Request-Partitioning-Based Processor Allocation in Two-Dimensional Mesh-based Multicomputers

Sulieman Bani-Ahmad
Department of Information Technology. Al-Balqa Applied University. Jordan
sulieman@bau.edu.jo

Abstract- **Request-Partitioning-Based (RPB) allocation strategies remedy the problem of fragmentation by allowing requests to be partitioned and allocated non-contiguously. Two RPB allocation schemes are proposed in literature; the Adaptive Non-Contiguous Allocation (ANCA) and the Bounded-Gradual-Partitioning (BGP) allocation. In ANCA, the frame requested by the parallel job is divided into two subframes of *equal sizes* at the longest dimension of the request. In BGP, requests are gradually partitioned into one *large* and another *small* subframe of multicomputers. In this paper, ANCA and BGP based allocation strategies are comparatively evaluated through exhaustive simulation-based experiments. Our experimental results also showed that the ANCA scheme can sustain higher system and communication loads compared to BGP in terms of major system performance metrics. We also observed that, in the BGP approach, increasing the partitioning bound value can slightly improve the performance of the parallel system. Comparatively, increasing the partitioning bound in the ANCA approach could significantly improve the performance of the parallel system.**

Keywords: Non-contiguous allocation; 2D-Mesh Multicomputers; Request-Partitioning; ANCA; BGP.

I. INTRODUCTION

Multicomputer systems are cost-effective alternatives of the traditional supercomputers [7]. The interconnection of multicomputers come in different styles called topologies. The two-dimensional (2D) and three-dimensional (3D) mesh-based topologies are probably the most common topologies because they are simple, regular and scalable [1; 5; 6]. Several recent commercial and experimental parallel computers have been built based these two architectures such as the IBM BlueGene/L and the Intel Paragon [1; 5; 6].

Contiguous allocation strategies of mesh-connected multicomputers attempt to locate a contiguous portion of the computing units for the execution of a parallel job [21; 1; 11; 8]. Contiguity of multicomputers helps in minimizing the distance of interprocessor communication path and in avoiding the interprocess interference that creates communication contention.

Another feature of contiguous processor allocation is that all the multicomputers allocated to a parallel job retain the same exact topology as the underlying multicomputer system. Further, the number of multicomputers allocated to a particular parallel job is determined according to the requirement of that parallel job [7]. Thus, in a mesh-connected multicomputer, jobs are allocated to submeshes [7; 1; 11; 8]. A parallel job retains all the multicomputers of the submesh for the entire duration of its life time. Once a parallel job is allocated, it runs till completion (i.e., no time-sharing) [1; 5; 6].

The processor allocator module in a multicomputer system applies allocation strategies or algorithms to identify and assign unallocated multicomputers to parallel jobs [7]. Allocation strategies with better *recognition ability* for available submeshes of unallocated multicomputers can improve the chance of assigning a parallel job into the system and, thus, reduce the job waiting delay [7; 1; 5; 6].

Studies showed that performance improvement cannot be obtained by refining the contiguous allocation strategies [7; 1]. Because of fragmentation problem, the average percent system utilization of a system can significantly degrade [7; 1]. Fragmentation occurs when there are enough unallocated multicomputers in the parallel multicomputer system but the allocator module fails to allocate these multicomputers to the waiting parallel jobs as they are *non-contiguous* or dispersed. This, in turn, limits the performance of contemporary allocation schemes. Consequently, contiguous allocation strategies fail to reduce the effect of fragmentation and hence provide very limited performance.

Request-Partitioning-Based (RPB) processor allocation strategies remedy the problems of fragmentation and low system utilization by allowing requests of parallel jobs to be partitioned and allocated non-contiguously into smaller subframes in case contiguous allocation fails [5; 6; 2; 3; 4]. Notice that small subframes are usually easy to be successfully allocated; the probability of successfully allocating a parallel job is increased. Studies show that RPB allocation strategies can successfully combine the advantages of both contiguous and non-contiguous allocation strategies through preserving some level of contiguity within allocated parallel job. Two RPB allocation schemes are proposed and studied in literature; the Adaptive Non-Contiguous Allocation (ANCA) [7; 4] and the Bounded-Gradual-Partitioning (BGP) allocation [5; 6; 2; 3].

Both RPB schemes try to solve the problem of fragmentation by allowing parallel jobs to be allocated non-contiguously. In ANCA, allocation of jobs is done by splitting the frame requested by the parallel job in hand into two subframes of equal sizes. Splitting is done at the longest dimension of the request [7; 4]. In BGP, however, allocation of jobs is achieved by gradually partitioning the frame requested into one large and another small subframe

of multicomputers [5; 6; 2; 3]. Both ANCA and BGP schemes prevent over-partitioning by placing a limit to the maximum number of non-contiguous *blocks of multicomputers* or subframes that can be assigned to any parallel job. This maximum number is referred to as the partitioning-bound [3; 4].

In this paper we comparatively evaluate both RPB schemes using computer-based simulation. The ANCA allocation strategy tested in this paper is the implementation proposed in [4]. Our experimental results also showed that the ANCA scheme could sustain higher system and communication loads compared to BGP in terms of major system performance metrics. We also observed that, in the BGP approach, increasing the partitioning bound value could slightly improve the performance of the parallel system. Comparatively, increasing the partitioning bound in the ANCA approach could significantly improve the performance of the parallel system.

II. PREVIOUS ALLOCATION STRATEGIES

Contiguous processor allocation strategies focus on finding the requested submesh according to the request of a job in terms of shape (and in orientation in some strategies). Non-contiguous allocation strategies alleviate the constraint of contiguity to achieve higher system utilization [5; 6; 2; 3]. Next we outline several non-contiguous allocation strategies for 2D-mesh multicomputers proposed in the literature.

Hardware advances such as wormhole routing and faster switching techniques have made the communication latency less sensitive to the distance between the communicating nodes [15; 7, 1; 5; 6]. This makes allocating a parallel job to a non-contiguous set of multicomputers plausible. By alleviating the restriction of contiguity, parallel jobs can get allocated and executed early. Several non-contiguous allocation algorithms are proposed in literature. Examples are: the random, the Multiple Buddy System (MBS) [13] and the Paging algorithms [12].

Non-contiguous allocation algorithms can be (i) totally non-contiguous and (ii) partially non-contiguous [4; 6]. In a *totally non-contiguous* allocation, a parallel job can be allocated as long as the number of available processing units is sufficient for its execution. In a *partially non-contiguous* allocation, the processing units allocated to a job retain a certain degree of contiguity.

Partially non-contiguous allocations can successfully provide higher performance than the totally non-contiguous allocations as they that reduce jobs dispersal [4; 12; 14]. In Paging algorithm, there is some degree of contiguity because of the indexing schemes used. Contiguity can also be increased by increasing the index parameter. However, this may produce internal processor fragmentation for large index sizes [12]. In MBS, contiguous allocation is explicitly sought only for requests with sizes of the form 2^{2n}, where n is a positive integer [4].

Request-Partitioning-Based (RPB) processor allocation strategies are partially non-contiguous allocation strategies that remedy the problems of fragmentation and low system utilization by allowing requests of parallel jobs to be partitioned and allocated non-contiguously into smaller subframes in case contiguous allocation fails [5; 6; 2; 3; 4]. Studies show that RPB allocation strategies combine the advantages of both contiguous and non-contiguous allocation strategies through preserving *some* level of contiguity within allocated parallel job [4].

Two RPB allocation schemes are proposed in literature; the Adaptive Non-Contiguous Allocation (ANCA) [7; 4] and the Bounded-Gradual-Partitioning (BGP) allocation [5; 6; 2; 3]. In ANCA, allocation of jobs is done by splitting the frame requested by the parallel job in hand into two subframes of *equal* sizes. Splitting is done at the longest dimension of the request [7; 4]. In BGP, however, allocation of jobs is achieved by gradually partitioning the frame requested into *one large* subframe and another *small* subframe [5; 6; 2; 3]. Both ANCA and BGP schemes prevent over-partitioning by placing a limit to the maximum number of non-contiguous *blocks of multicomputers* or subframes that can be assigned to any parallel job. This maximum number is referred to as the partitioning-bound [3; 4]. Next we describe the RPB allocation schemes in more detailed manner.

A. *The ANCA and Modified ANCA Schemes*

The ANCA algorithm attempts to allocate the job at hand contiguously. If the it fails, it partitions the request into two equi-sized sub-requests. These subframes are then allocated to available locations, if possible; otherwise, each of these sub-requests is recursively further partitioned into two equi-sized sub-requests, and then ANCA tries to map these sub-requests to available locations [4; 7].

In [4] a modified version of the ANCA approach is presented and evaluated, the ANCA-based allocator tries to allocate the parallel job in hand using some given contiguous allocation strategy (e.g., the first-fit or best-fit). This contiguous method is recommended to be selected to be of a good free-submesh recognition capability. If the allocator module fails to allocate the parallel job, it splits the job into two subframes that are as equal as possible in terms of size and topology [4]. This is done by splitting the request at its longest dimension. Given that the request is of dimensions *AxB*. Assume that *A>B*. If the value of the length of the longest dimension of the request is even, then the two subframes will be of exactly-equal sizes *(A/2 x B)*. If the length is odd, the original implementation of the ANCA algorithm [7] tries to allocate two submeshes each of size of *Ceiling(A/2) x B*. This causes having internal fragmentation of size *1 x B*. The original implementation of the ANCA strategy solves this problem of internal fragmentation by *bookkeeping of idle nodes* [7]. In the modified implementation of the ANCA algorithm the request in hand divided into two with the following dimensions: the first is *Ceiling(A/2) x B* and *floor(A/2) x B*. As a result, no need for bookkeeping.

B. *The Gradual-Request-Partitioning-Based (GRP) allocation schemes*

In [2; 3; 6], a family of *adaptive* non-contiguous allocation algorithms for 2D-mesh multicomputers are proposed. These algorithms are all utilizes a contiguous allocation strategy implicitly. These algorithms try to find a contiguous set of

processing units of the same shape and size to the request in hand using the contiguous allocation algorithm. If they fail, the request in hand is divided into two sub-requests after removing one from the longest dimension of the request. That is, for a given request of size αxβ and assuming β>α, the two partition-sizes are αx(β-1) and αx1 after removing one from the longest dimension of the request. The two new sub-requests are then allocated using the contiguous allocation algorithm again. This procedure continues recursively until the request is fulfilled. These algorithms are referred to as PALD-based approaches. PALD stands for ***PA***rtitioning at the ***L***ongest ***D***imension [2; 3].

III. PERFORMANCE EVALUATION

The BGP and ANCA allocation algorithms is implemented in the C language, and later integrated with the ProcSimity simulation tool [20; 17]. 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%. Parallel jobs usually communicate with each other using *all-to-all* communication pattern [19; 10; 13]. We did our experiments using this communication pattern as it produces high message collision is known to be a weak point for non-contiguous allocation algorithms [5; 1]. The processor allocation strategies were tested under the scheduling strategy First-Come-First-Serve. The simulated parallel system is of size 20x20 computing units. The size of the simulated parallel job follows the exponential distribution with an average of 10 units for each dimension.

IV. RESULTS AND OBSERVATIONS

Figure 1 and 2 show how increasing the load of the parallel system affects the Finish Time (FT) of the ten runs of simulator and the Mean Job Response Time (MJRT), respectively. Two observations can be made of figure 1; the first is that increasing the Partitioning Bound (PB) value allows the parallel system to serve parallel jobs earlier and thus increase the system throughput. Consequently, the simulator FT is expected to be less for higher PB values (notice that non-contiguity of request allocated processing nodes does not significantly affect the service time of parallel jobs as they are not severely dispersed as clearly noticed in figure 2).

The other observation on figure 1 is that, considering the same PB values for the ANCA and BGP schemes, the ANCA scheme showed lower FT values. This can be explained as follows: ANCA splits parallel jobs in two subframes of equal sizes. The BGP, however, splits jobs into one relatively large subframe and another small subframe. Notice that the probability of successfully allocating the large subframe of the BGP scheme is less than that of allocating the two smaller subframes produced by the ANCA scheme. Taking into consideration that the allocation of a parallel job is successful if and only if both subframes are allocated, it is expected that the ANCA algorithm is expected to produce lower FT (figure 1) and also lower MJRT values (figure 2) for the same PB value enforced. For the same above reason, it is expected that the ANCA scheme to be superior over the BGP scheme in terms of Average System Utilization (ASU) as shown in figure 3 that proves that ANCA can sustain higher system loads and produce higher system utilization compared to BGP.

Both ANCA and BGP schemes can allocate parallel jobs in a non-contiguous manner. Thus, it is expected that both will disperse parallel jobs. Figure 4 shows how increasing the load of the system affects the mean number of blocks per parallel job (MBPJ). Figure 4 shows that, for the same partitioning bound value, The ANCA scheme usually allocates parallel jobs to more blocks than the BGP approach in average. More MBPJ values imply longer jobs service time. Figure 5 shows that, when the partitioning bound is small, the BGP approach produces lower mean job service time value (MJST) than that produced by the ANCA approach. However, for higher PB values, the performance of the BGP approach degrades$ and the ANCA approach produces less job service time in average.

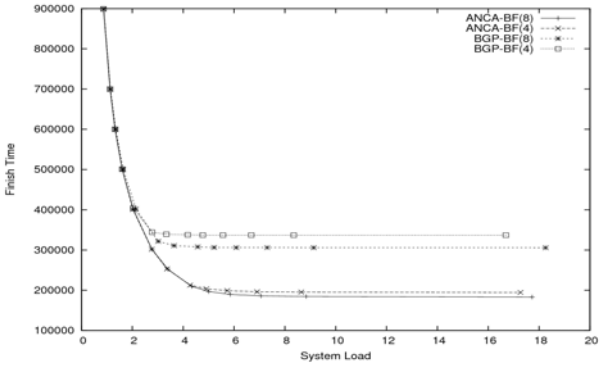

Figure 1: Finish time vs. system load in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

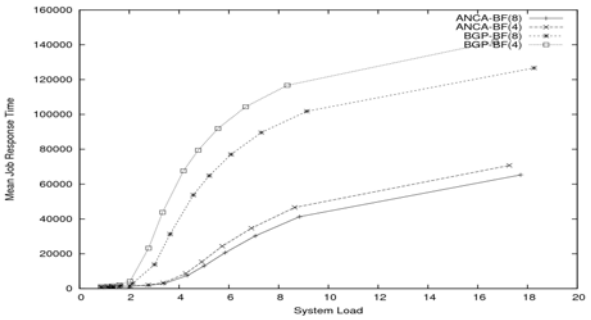

Figure 2: Mean job response time vs. system load in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and BPJ.

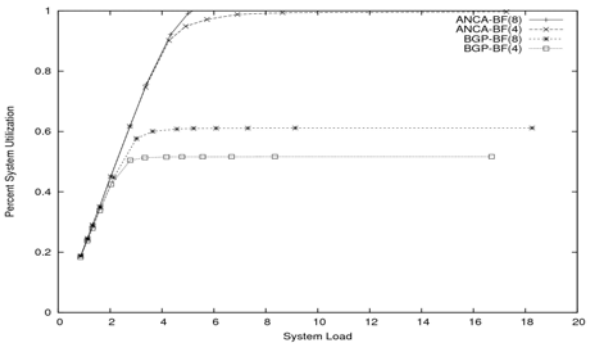

Figure 3: Percent system utilization vs. system load in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

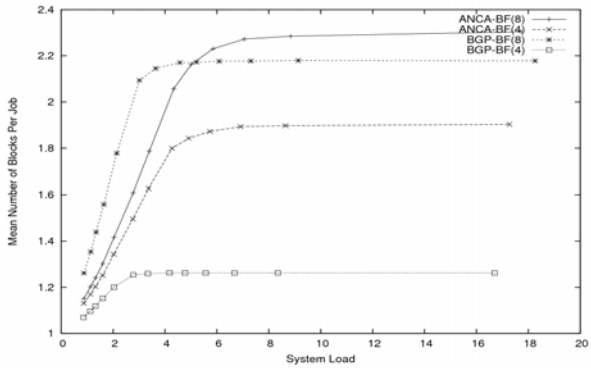

Figure 4: Mean number of blocks per job vs. system load in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

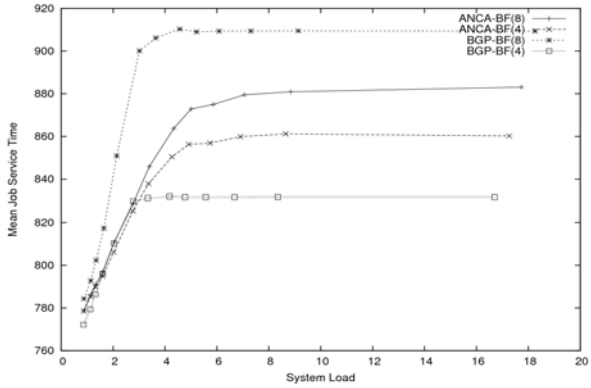

Figure 5: Mean job service time vs. system load in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

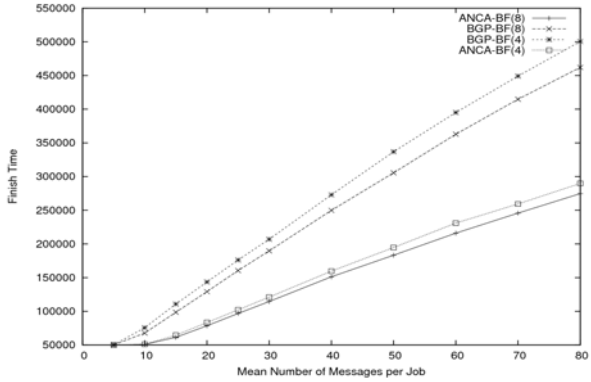

Figure 6: Finish time vs. mean number of messages per job in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

Figure 6 shows how increasing the communication load of the parallel system affects its performance in terms of finish time. In general, increasing the communication load increases FT because the service times of parallel jobs increase. The increase rate is higher for BGP approach.

Figure 7 shows how the communication load affects MBPJ. Similar to what we observed in figure 4, the ANCA approach is expected to allocate jobs to more blocks compared to the BGP approach. The ANCA approach can successfully allocate parallel jobs earlier than the BGP approach. The reason is that allocating the large subframe produced by the BGP allocator is more difficult to allocate the two smaller subframes produced by the ANCA approach. Consequently, the ANCA approach produces lower MJRT (as proven by figure 8). Further, as observed in figure 5, for low PB values, the BGP approach is expected to be superior to the ANCA approach in terms of MJST (as proven by figure 9). However, for higher PB values, the ANCA approach is superior to the BGP approach.

Figures 10 and 11 show how increasing the communication load of the parallel system affects the performance of the interconnection network linking the multicomputers in terms of mean packet blocking time (MPBT) and mean packet latency (MPL). It can be observed from the two figures that, in general, the ANCA approach is superior to the BGP approach in terms of MPBT at high communication loads (figure 10).

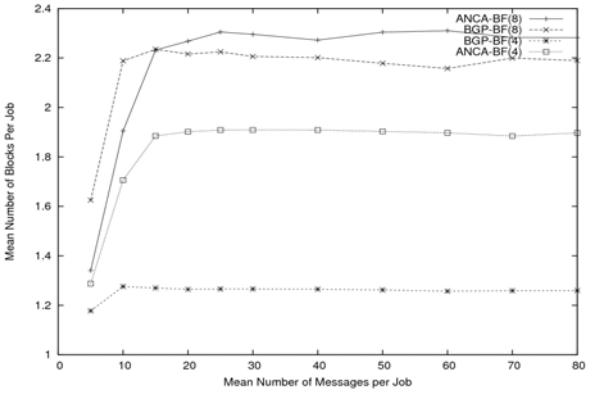

Figure 7: Mean number of blocks per job vs. mean number of messages per job in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

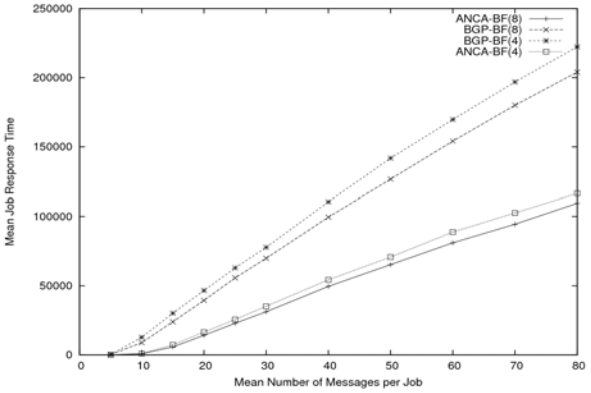

Figure 8: MJRT vs. mean number of messages per job in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

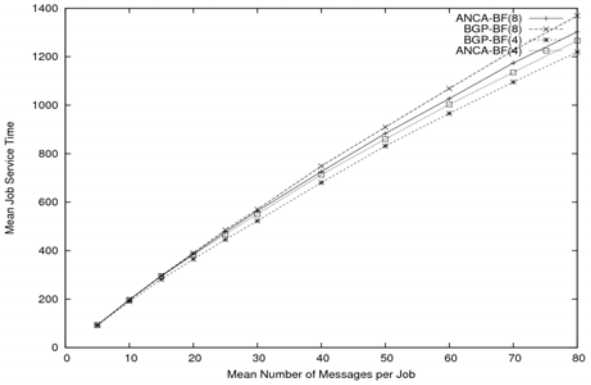

Figure 9: MJST vs. mean number of messages per job in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4 BPJ.

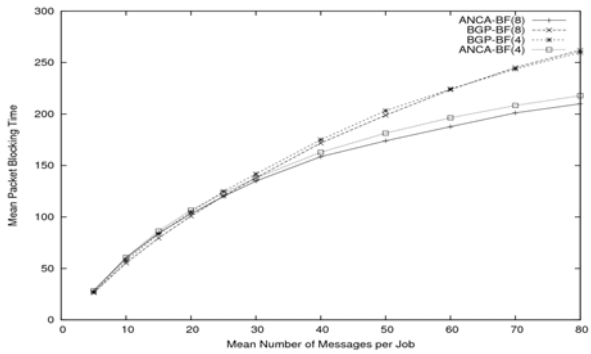

Figure 10: MPBT vs. mean number of messages per job in ANCA-BF and BGP-BF allocation strategies with partitioning bound of 8 and 4BPJ.

Figure 11 shows how FT changes as the PB values increases. Figure 11 shows that, as expected, FT decreases as PB value increases for both ANCA and BGP schemes. However, FT decays faster in the ANCA approach over increasing the PB value. The reason is that the ANCA approach is more successful in allocating parallel jobs early compared to the BGP approach as observed in fig. 1.

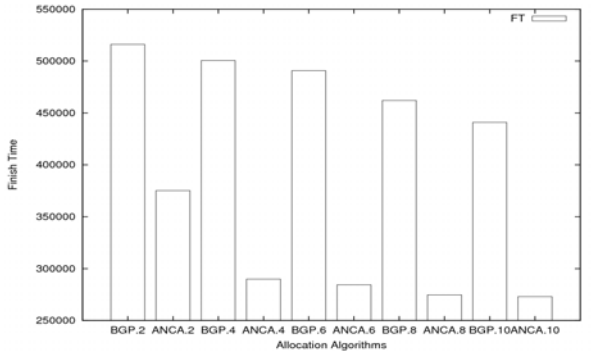

Figure 11: Finish Time vs. partitioning bound in ANCA-BF and BGP-BF allocation strategies.

V. 5. CONCLUSION

In this paper, the ANCA and BGP based allocation strategies are comparatively evaluated through exhaustive simulation-based experiments. Our experimental results shows that the ANCA and BGP allocation schemes are both flexible and tunable as it allows the allocator module to choose an optimal partitioning-bound value while allowing parallel jobs to be allocated early without having them over-partitioned. Our experimental results also showed that the ANCA scheme could sustain higher system and communication loads compared to BGP.

REFERENCES

[1]. Ababneh, I. (2006), "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.

[2]. Bani-Ahmad, S. (2010a), "Intra-job Communication Contention and Request-Partitioning-Based Allocation Strategies in 2D-Mesh Multicomputers". Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP'10). December 18-20, 2010, Dalian, LiaoNing, P. R. China.

[3]. Bani-Ahmad, S. (2010b), "Submesh Allocation in 2D-Mesh Multicomputer: Partitioning at the Longest Dimension of Requests". Proceedings of the Fourth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2010). October 25-30, 2010, Florence, Italy.

[4]. Bani-Ahmad, S. (2011). "Processor Allocation with Reduced Internal and External Fragmentation in 2D Mesh-based Multicomputers". Submitted to Journal of Applied Sciences.

[5]. Bani-Mohammad, S.; Ould-Khaoua , M.; Ababneh, I., and Machenzie, L. (2006), "Non-contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers Based on Sub-meshes Available for Allocation", Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06), vol. 2, IEEE Computer Society Press, USA, 2006, pp. 41-48.

[6]. Bani-Mohammad, S.; Ould-Khaoua, M.; Ababneh, I.; and Machenzie, L. (2007), "A Fast and Efficient Processor Allocation Strategy which Combines a Contiguous and Non-contiguous Processor Allocation Algorithms", Technical Report; TR-2007-229, DCS Technical Report Series, Department of Computing Science, University of Glasgow, January 2007.

[7]. Chang, C. Y. and Mohapatra, P. (1998), "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.

[8]. Chuang, P. J. and Tzeng, N. F. (1994), Allocating precise submesh in mesh-connected systems, IEEE Transactions Parallel and Distributed Systems (Feb. 1994), 211217.

[9]. Ding, J. and Bhuyan, L. N. (1993), An adaptive submesh allocation strategy for two-dimensional mesh connected systems, Proc. Int. Conf. Parallel Process. II (Aug. 1993), 193200.

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

[11]. Li, K. and Cheng, K. H. (1991), "A Two-Dimensional Buddy System for Dynamic Resource Allocation in a Partitionable Mesh Connected System", Journal of Parallel and Distributed Computing, vol. 12, no. 1, Elsevier Science, CA, USA, May 1991, pp. 79-83.

[12]. Liu, T.; W. Huang, K.; Lombardi, F. and Bhuyan, L. N. (1995), A submesh allocation scheme for mesh-connected multiprocessor systems, Proc. Int. Conf. Parallel Process. II (Aug. 1995), 159163.

[13]. Lo, V.; Windisch, K.; Liu, W.; and Nitzberg, B. (1997), "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.

[14]. Min, D. and Mutka, M. W. (1995), Effect of job interactions due to scattered processor allocations in 2-D wormhole networks, in ``Proc. of Int. Conf. on Parallel and Distributed Computing Systems," (Sept. 1995), pp. 262267.

[15]. Ni, L. M. and McKinley, P. K. (1993), "A Survey of Wormhole Routing Techniques in Direct Networks". Computer 26, 2 (Feb. 1993), pp 62-76. DOI= http://dx.doi.org/10.1109/2.191995.

[16]. Niedermeier, R.; K. Reinhardt; and P. Sanders (1997). Towards optimal locality in mesh indexings. In Proc. 11th Intl Symp on Fund. Computation Theory, volume 1279 of LNCS, pages 364-375, 1997.

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

[18]. Suzaki, K.; Tanuma, H.; Hirano, S.; Ichisugi, Y.; Connelly, C.; and Tsukamoto, M. (1996), "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.

[19]. Windisch, K.; Miller, J. V.; and Lo, V. (1995), "ProcSimity: an experimental tool for processor allocation and scheduling in highly parallel systems", Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation (Frontiers'95), IEEE Computer Society Press, Washington, USA, 6-9 Feb 1995, pp. 414-421.

[20]. 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, vol. 51, no. 1, IEEE Computer Society, Washington, USA, January 2002, pp. 46-60.

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