Submesh Allocation in 2D-Mesh Multicomputers: Partitioning at the Longest Dimension of Requests

Sulieman Bani-Ahmad
Department of Information Technology
Al-Balqa Applied University
Al-Salt, Jordan
sulieman@case.edu

***Abstract*-- Two adaptive noncontiguous allocation strategies for 2D-mesh multicomputers are proposed in this paper. The first is first-fit-based and the second is best-fit-based. That is; for a given request, the proposed first-fit-based approach tries to find a free submesh using the well-known first-fit strategy, if it fails, the request at hand is partitioned into two *sub-requests* that are allocated using the first-fit approach. Partitioning is performed at 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 first-fit strategy. This procedure continues recursively until the request is fulfilled. The second approach is also based on *PArtitioning at the Longest Dimension* (PALD) of requests but a best-fit approach is used to allocate requests and sub-requests. The partitioning mechanism aims at (i) lifting the condition of contiguity, and (ii) at the same time maintaining *good* level of contiguity. Removing one from the longest dimension of a request is expected to produce two sub-requests one of which is relatively big and as close as possible to the square-shape and, thus; reducing communication latency caused by non-contiguity. Using extensive simulations, we evaluated the proposed strategies and compared them with previous contiguous and non-contiguous strategies. Simulation outcomes clearly show the proposed PALD-based schemes produce the best *Average Response Time* (ART), the *Average System Utilization* (ASU) and also produce relatively low communication overhead.**

Keywords- Multicomputer; 2D mesh; Non-contiguous Allocation; Request Partitioning.

I. INTRODUCTION

In parallel systems, processors are connected through interconnection network; one of the most widely used architectures is the 2D and 3D mesh-connected architectures. This is because mesh architecture is simple, regular and scalable [4, 14]. Several recent commercial and experimental parallel computers have been built based these architectures such as the IBM BlueGene/L and the Intel Paragon [4].

Processor allocation in 2D-Mesh multicomputer is a major issue as it significantly affects the performance of any parallel system [4]. Processor allocation is concerned with the way for allocation submesh to a job request. Many processor allocation strategies in literature try to allocate a submesh, i.e., a contiguous set of processing units, of the same size and shape of request [1, 3, 4, 5, 7, 12, 21]. This, however, may produce low level of system utilization and cause either internal or external fragmentation or both [2, 18]. Internal fragmentation occurs when the number of processors allocated to a job is more than that it requested [16]. External fragmentation, on the other hand, occurs when enough number of idle processors is available in the system but cannot be assigned to the scheduled job because of the requirement of contiguity [2]. Several studies have attempted to reduce or solve external fragmentation [2, 9, 6, 14, 16, 18], one of the proposed solutions is to use non-contiguous allocation.

In non-contiguous allocation the contiguity condition is relaxed [2]; therefore, a job can execute on multiple disjoint smaller sub-meshes rather than always waiting until a single sub-mesh of the requested size and shape is available [2, 9, 14, 18]. Studies show that non-contiguous allocation of requests may solve the drawbacks of contiguous allocation; non-contiguous allocation strategies produce relatively high system utilization and eliminate fragmentation. However, since communication between processors running the same job can be indirect due to non-contiguity [16], communication latency is usually high. However, the introduction of wormhole routing [17] has lead researchers to consider noncontiguous allocation on multicomputers with a long communication distances, such as the 2D mesh [2, 14, 18]. One of main advantages of wormhole routing over earlier communication schemes, e.g., store-and-forward, is that message latency is less dependent on the distance traversed by the message from source to destination [2, 17]. Thus, non-contiguous allocation has recently received attention of researchers.

Partitioning allocation requests in existing non-contiguous allocation schemes can be performed in multiple ways. For example, allocation requests are subdivided into two equal partitions in [2]. The sub-partitions are recursively subdivided into further smaller sub-requests if allocation fails for any of them. In the study of [18], a promising strategy (MBS) expresses the allocation request as a base-4 number, and bases allocation on this expression.

In this paper, two *adaptive* noncontiguous allocation strategies for 2D-mesh multicomputers are proposed and evaluated through simulation. The first is a first-fit-based

 ISBN: 978-1-61208-101-4

approach that tries to find a contiguous set of processing units of the same shape and size to the request at hand using the well-known first-fit approach. If it fails, the request at 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 first-fit approach again. This procedure continues recursively until the request is fulfilled. This approach is referred to a PALD-FF for ***PA***rtitioning at the ***L***ongest ***D***imension with First-Fit.

The second approach is also PALD-based. However, the best-fit (BF) allocation strategy is used to allocate requests and sub-requests. The used partitioning mechanism aims at (i) lifting the condition of contiguity, and (ii) at the same time maintaining *good* level of contiguity. Removing one from the longest dimension of a request is expected to produce two sub-requests one of which is relatively big and as close as possible to be square-shaped and, thus; reducing communication latency caused by non-contiguity.

Using extensive simulations, we evaluated the proposed strategies and compared them with previous promising strategies. Simulation outcomes clearly show the proposed PALD-based schemes produces the best *Average Response Time* (ART), the *Average System Utilization* (ASU) and produce relatively low communication overhead. The performance of PALD-FF and PALD-BF is compared against the performance of the MBS non-contiguous allocation strategy. This strategy is selected as it has been shown to perform well in [18]. Furthermore, proposed approaches are also compared against the contiguous First-Fit and Best-Fit strategies as this has been used in several previous related studies [2, 3, 18]. The proposed approaches are tested under two job scheduling strategies, namely; first-Come-First-Served (FCFS) and Shortest-Service-Demand-First (SSD). In FCFS, the allocation request that arrived first is scheduled for allocation first. In SSD, the job with the shortest service demand is scheduled first [11]. The FCFS scheduling strategy is chosen as it is fair and it is widely used in other similar studies [2, 3, 4, 6, 14], while the SSD scheduling strategy is used to avoid performance loss due to blocking [11].

II. RELATED WORK

In this section, we provide an overview of some existing contiguous and non-contiguous allocation strategies.

A. Non-Contiguous Allocation Strategies

The ***First Fit (FF)*** strategy is a *contiguous allocation strategy*. This scheme start search at the lowest leftmost node in mesh, and put a virtual grid that's equal size request, and then shifts by one column to the right until first large enough free submesh is found [13]. The ***Best fit (BF)*** is also a contiguous allocation strategy. This scheme is the same as first fit scheme, but it reserves a submesh after consider all large enough free submeshes and chooses the closest requests, i.e., the submesh with minimal leftovers is selected [13]. We use both strategies to search for free submeshes for the partitioned requests as should be shortly illustrated more.

B. Non-Contiguous Allocation Strategies

The introduction of wormhole routing [17] has made communication latency less sensitive to the distance traversed by between communicating entities [2]. This has made allocating a job to non-contiguous processors reasonable, in terms of performance, in networks characterized by a relatively long-diameter, such the 2D mesh. Non-contiguous alleviates the contiguity and thus allowing jobs to be executed without waiting for contiguous set of idle nodes [2, 14].

In the Paging strategy, for instance [18], the entire 2D mesh is virtually sub-divided into pages or sub-meshes of equal sides' length of 2^i where i is a positive integer number that represents the index parameter of the paging approach. The pages are indexed according to several indexing schemes.

In the Multiple Buddy System (MBS) strategy, the mesh of the system at hand is divided into non-overlapping square sub-meshes with side lengths that are powers of 2. The number of processors, p, requested by a job is factorized into a base-4 block. If a required block is unavailable, MBS recursively searches for a larger block and repeatedly breaks it down into *four* buddies until it produces blocks of the desired size. If that fails, the requested block is further broken into four sub-requests until the job is allocated [18].

In the Adaptive Non-Contiguous Allocation (ANCA) strategy work differently. ANCA first attempts to allocate the job at hand contiguously. If the allocation attempt fails, it partitions the request into two equi-sized sub-requests. These sub-frames are then allocated to available locations, if possible; otherwise, each of these sub-requests is recursively further partitioned into two sub-requests, and then ANCA tries to map these sub-requests to available locations [2].

Maintaining a *good* level of contiguity can prove useful in non-contiguous allocation. In Paging, 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 [18]. In MBS, contiguous allocation is explicitly sought only for requests with sizes of the form 2^{2n}, where n is a positive integer.

An issue with the ANCA strategy is that it can disperse the allocated sub-meshes more than it is necessary through *over partitioning*. Over-partitioning may cause skipping over the possibility of identifying and thus allocating larger free sub-meshes for a large part of the request at hand which has been shown to maintain a higher level of contiguity [15]. Thus the communication overhead can be reduced by adaptively and gradually partitioning allocation requests into as large as possible contiguous sub-meshes.

 ISBN: 978-1-61208-101-4

III. THE PROPOSED ALLOCATION STRATEGY

The target system is a W × L two-dimensional mesh, where W and L are the width and the length of the mesh, respectively. Every processor is denoted by a pair of coordinates, namely; x and y, where $0 \le x < W$ and $0 \le y < L$ [14]. Each processor is connected by bidirectional communication links to its neighbor processors.

In this paper, two *adaptive* noncontiguous allocation strategies for 2D-mesh multicomputers are proposed and evaluated. The first is a first-fit-based approach that tries to find a contiguous set of processing units of the same shape and size to the request at hand using the well-known first-fit approach. If it fails, the request at 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 first-fit approach again. This procedure continues recursively until the request is fulfilled. This approach is referred to a PALD-FF for ***PA***rtitioning at the ***L***ongest ***D***imension with First-Fit.

```
Procedure PALD-FF(a, b):
Begin
        JobSize = a × b
        If (number of free processors < JobSize) return failure
        List AllocatedPIDs={}; // the list of PIDs allocate to the current job
        Return PALD-FFAllocate(a, b, AllocatedPIDs);
End

Procedure PALD-FFAllocate (a, b, AllocatedPIDs)
Begin
        S(x, y) = FIND_FF (S(a, b); // FIND_BF for PALD-BF allocation
        If (S(x, y)  != null)
          Add the PIDs of S to the list AllocatedPIDs;
        Else
        {
                If(a>=b)
                        α1= a-1;  β1=b; α2= 1;  β2=b;
                else
                        α1= a;  β1=b-1; α2= a;  β2=1;
                PALD-FFAllocate (α1, β1, AllocatedPIDs);
                PALD-FFAllocate (α2, β2, AllocatedPIDs);
        }
End
```

Figure 1: Pseudo code for the PALD-FF allocation strategy

The second approach is also PALD-based. However, the best-fit (BF) strategy is used to allocate requests and sub-requests. The used partitioning mechanism aims at (i) lifting the condition of contiguity, and (ii) at the same time maintaining *good* level of contiguity. Removing one from the longest dimension of a request is expected to produce two sub-requests one of which is relatively big and as close as possible to be square-shaped and, thus; reducing communication latency caused by non-contiguity.

The proposed PALD-based approach combines the desirable features of both contiguous and non-contiguous allocation. The well-known first-fit (FF) and best-fit (BF) strategies are used here to search for available submeshes. A 13eudo code for the allocation procedure of PALD-FF strategy is shown in Figure 1. The PALD-BF is similar except that the Find_BF() is called instead of Find_FF() to allocate partitions of the parallel job at hand. Notice that allocation always succeeds as long as enough free processors are available in the mesh. The key idea in the proposed PALD approach is to try allocating the largest submeshes possible.

IV. EXPERIMENTAL SETUP AND SIMULATION OUTPUT

The current study is simulation-based with the ProcSimity simulator is to be used. The simulated multicomputer system consists of 256 multicomputers connected through a 2-dimensional mesh network of dimensions W and L W=L=16. The routing mechanism to be used is the wormhole routing [10, 20] with packet size of 8 units and a buffer of size 1 unit and a routing delay of 3 units. The router uses XY routing to direct messages from their source to destination. Message sizes are considered to be of length 8 units. Job size conforms the exponential distribution with mean width and length being W/2 (or L/2). The execution times of jobs conforms the uniform distribution.

To maintain good levels of accuracy, each simulation experiments is repeated 10 times with a total of 1000 jobs are to be simulated in each time. The readings are 95% accurate with a maximum percentage error of 5%. The scheduling mechanisms considered in our experiments are (i) First-Come-First-Serve, or FCFS and (ii) the Shortest Service Demand first, or SSD mechanisms. The simulation outputs are:

(i) Average Response Time (ART): The response time is the time from the submission of request until the first real response produced for jobs. (ii) ***Average system utilization (ASU)***: The average of keeping the processors within a system as busy as possible, this value between 0 and 1. (iii) ***Average Packet Blocking Time (APBT)***: The average amount of time the head of the message is blocked at each station while routing the message over the path from source to destination. (iv) ***Average Packet Latency (APL)***: The average of the time that all packets within job will be sent between processors.

V. EXPERIMENTAL RESULTS AND OBSERVATIONS

In this section, the results from simulations that have been carried out to evaluate the performance of the proposed algorithm are presented and compared against those of MBS, BF and FF. The proposed allocation algorithm is implemented and later integrated with the ProcSimity simulation tool [8, 13]. 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%.

Next we present our experimental results and observations. Parallel jobs usually communicate with each other using one-to-all or all-to-all communication patterns [9, 17, 18]. We did our experiments using both pattern but focused more on the all-to-all communication pattern as it produces message collision than the one-to-all communication pattern and is known to be a weak point for non-contiguous allocation algorithms [9]. The independent variable in the simulation is the system load. The notation *<allocation strategy>(<scheduling strategy>)* is used to

 ISBN: 978-1-61208-101-4

represent the strategies in the performance figures, as in [15]. For example, PALD-FF(FCFS) refers to the PALD-FF processor allocation strategy under the scheduling strategy FCFS.

A. Mean response time criteria

In Figures 2 through 4, the mean job response time of jobs is plotted against the system load for the one-to-all and all-to-all communication patterns under the FCFS and SSD scheduling mechanisms. The figures reveal that PALD-based allocation strategies produce less response times and, thus, perform better than all other strategies. This is more clear under the SSD scheduling mechanism. PALD-FF is substantially superior to the FF and PALD-BF is also superior to BF. For all-to-all communication pattern both tested PALD-based allocation strategies outperformed contiguous allocation strategies.

Figure 5 shows the four allocation strategies compared together in terms of response time. Considering the same system settings figure 5 shows that PALD-based approaches outperform non-PALD-based ones.

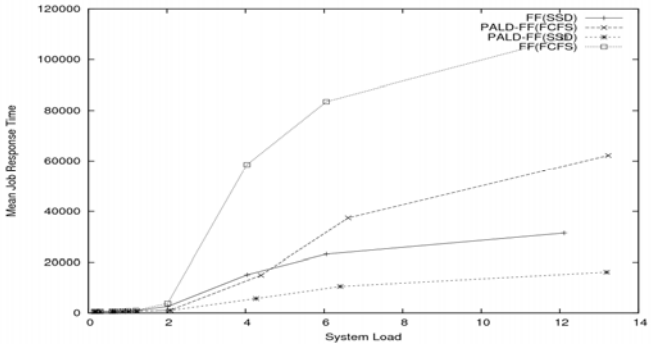

Figure 2: Mean response time in FF and PALD-FF strategies under the FCFS and the SSD scheduling mechanisms and one-to-all communication pattern.

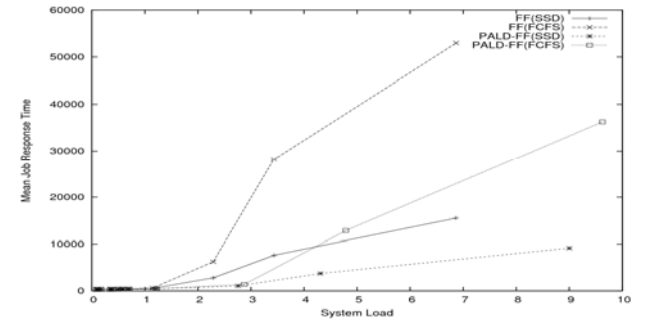

Figure 3: Mean response time in FF and PALD-FF strategies under the FCFS and the SSD scheduling mechanisms and all-to-all communication pattern.

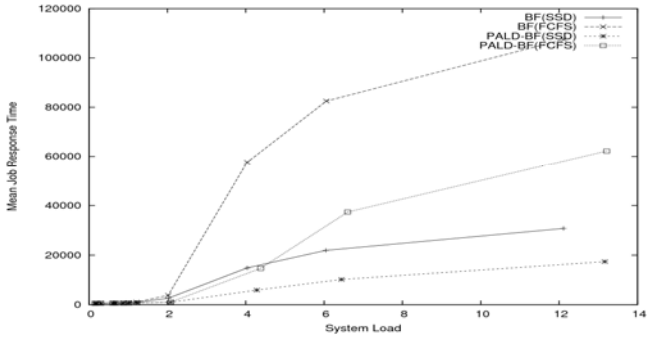

Figure 4: Mean response time in BF and PALD-BF strategies under the FCFS and the SSD scheduling mechanisms and one-to-all communication pattern.

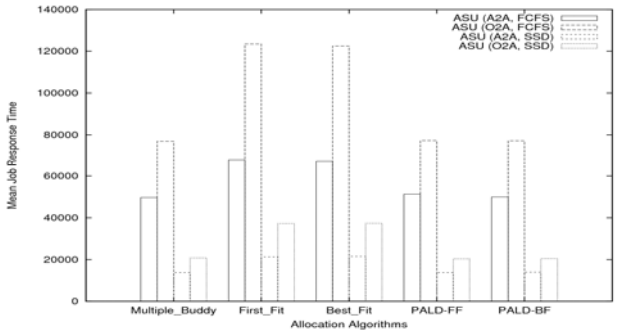

Figure 5: Mean response time in MBS, FF, BF, PALD-FF and PALD-BF strategies under both scheduling mechanisms, both communication patterns.

B. Percent system utilization criteria

Figures 6 through 9 depict the mean system utilization of the tested allocation strategies, namely; FF, BF, PALD-FF, PALD-BF and MBS, for the two communication patterns considered and under the FCFS and SSD scheduling mechanisms. Figures 6 and 7 depict the percent system utilization in FF and PALD-FF allocation strategies under the FCFS and the SSD scheduling mechanisms and one-to-all and all-to-all communication patterns. Similarly, Figures 8 and 9 depict the percent system utilization in BF and PALD-BF allocation strategies under the FCFS and the SSD scheduling mechanisms and both communication patterns.

Figures 6 through 9 reveal that the PALD-based strategies produce higher system utilization and. This is more clear under the SSD scheduling mechanism. PALD-FF and PALD-BF showed around 70% higher system utilization than the FF and BF approaches at the points where the system is heavily loaded, respectively. This observation applied for both communication patterns. This observation can be explained as follows, contiguous allocation produces high external fragmentation, which means that allocation is less likely to succeed. Consequently, system utilization becomes low. The proposed approaches have the ability to eliminate both internal and external processor fragmentation, and thus, produce higher system utilization.

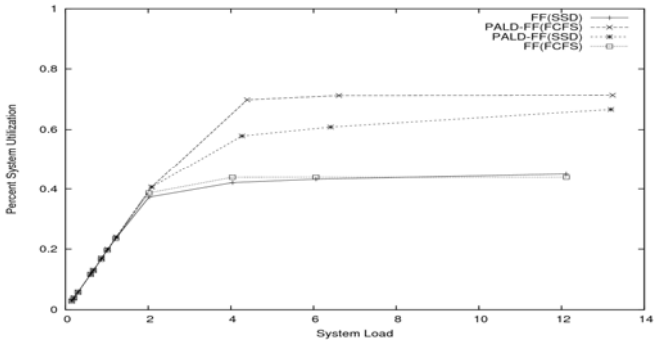

Figure 6: System utilization in FF and PALD-FF strategies under the FCFS and the SSD scheduling mechanisms and one-to-all communication pattern.

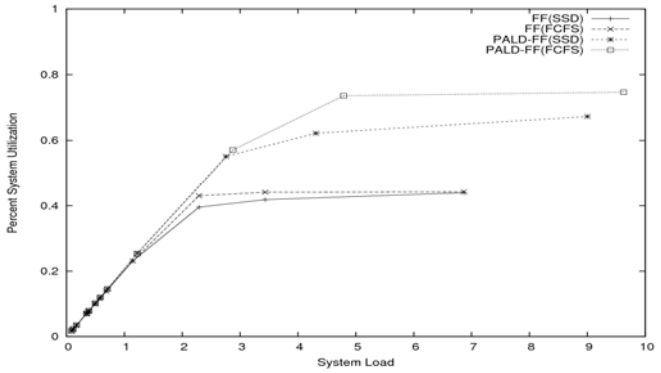

Figure 7: System utilization in FF and PALD-FF strategies under the FCFS and the SSD scheduling mechanisms and all-to-all communication pattern.

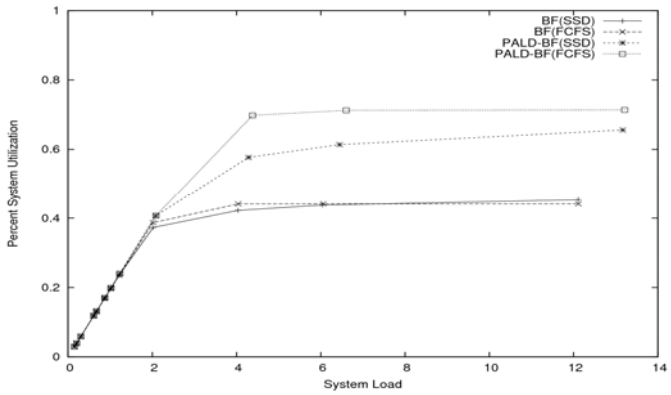

Figure 8: System utilization in BF and PALD-BF strategies under the FCFS and the SSD scheduling mechanisms and one-to-all communication pattern.

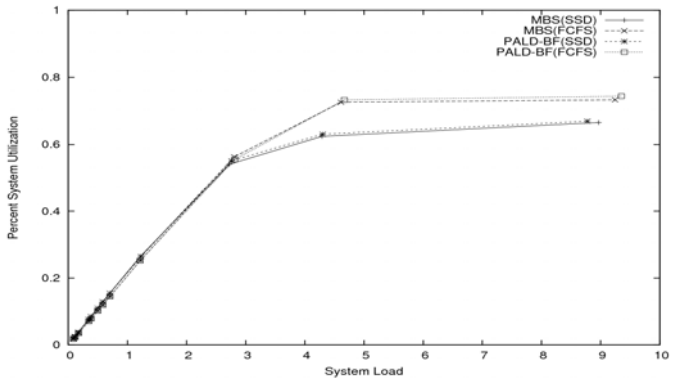

Figure 9: System utilization in MBS and PALD-BF strategies under the FCFS and the SSD scheduling mechanisms and all-to-all communication pattern.

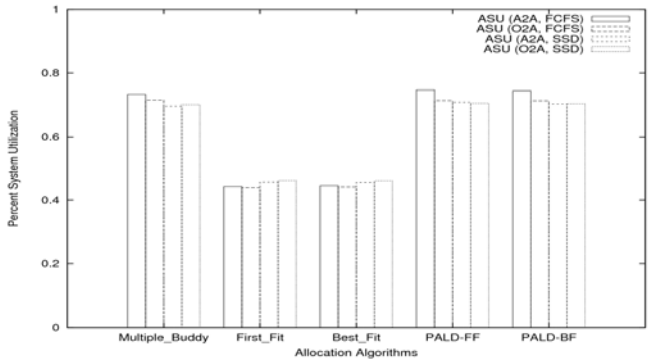

Figure 10: System utilization in MBS, FF, BF, PALD-FF and PALD-BF strategies under both scheduling mechanisms, both communication patterns.

C. Communication Overhead

We have measured other performance criteria for the non-contiguous allocation strategies. These are the mean packet latency (MPL) and the mean packet blocking time (MPBT). Figure 11 shows that the MPL for the tested allocation strategies for all-to-all communication pattern and under the two considered scheduling mechanisms. It can be seen that PALD-FF and PALD-BF strategies have lower MPL values than MBS strategy under the two scheduling strategies FCFS and SSD for the all-to-all communication pattern. This conclusion is compatible with the values of the mean turnaround time shown above.

To summarize, the above performance results demonstrate that PALD-FF and PALD-BF strategies are superior to all other strategies considered in this paper; including the case when contention is heavy (the communication pattern is all-to-all). Figure 12 shows that the MPBT for the tested allocation strategies under the two considered scheduling mechanisms is less than that of MBS strategy.

One concern in PALD-based allocation strategies is that requests may get over-partitioned. This results in allocating dispersed multicomputers to parallel jobs. To test that, we repeated our experiments and allowed for giving a control over the maximum number of blocks allowed to any allocated job (MBPJ). Figure 13 illustrates the observed relationship between MBPJ (the x-axis) and the average system utilization (the y-axis). At an MBPJ value of 14, we found that the system utilization reaches a maximum saturation value of around 0.92. Thus, placing this limit helps in (i) preventing over-partitioning and (ii) keeping the allocation time complexity of PALD allocation strategies to be the same as that of the contiguous allocation strategy used (the FF or BF).

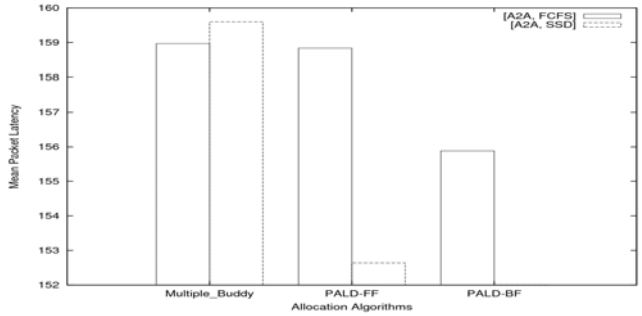

Figure 11: Mean packet latency in MBS, PALD-FF and PALD-BF allocation strategies under the FCFS and the SSD scheduling mechanisms, all-to-all communication patterns.

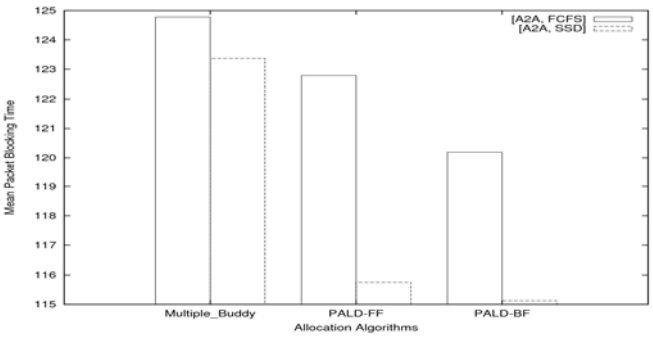

Figure 12: Mean packet blocking time in MBS, PALD-FF and PALD-BF allocation strategies under the FCFS and the SSD scheduling mechanisms, all-to-all communication patterns.

 ISBN: 978-1-61208-101-4

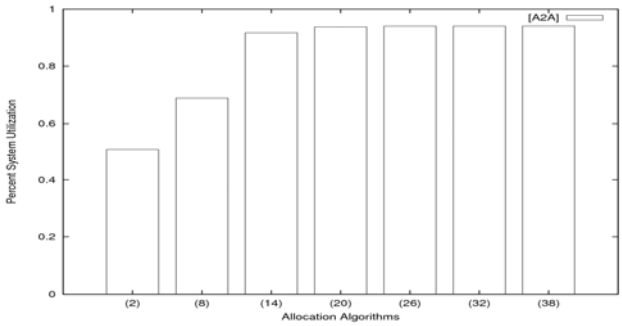

Figure 13: System utilization vs partitioning limit for PALD-BF allocation strategy under the FCFS scheduling mechanism, all-to-all communication patterns.

VI. CONCLUSIONS

Two adaptive noncontiguous allocation strategies are proposed in this paper. The first is first-fit-based and the second is best-fit-based. That is; for a given request, the proposed first-fit-based approach tries to find a free submesh using the well-known first-fit strategy, if it fails, the request at hand is partitioned into two *sub-requests* that are allocated using the first-fit approach. Partitioning is performed at the longest dimension of the request (removing one from the longest dimension of the request at hand). The two new sub-requests are then allocated using the first-fit or the best-fit approaches. This procedure continues recursively until the request is fulfilled. The second approach is also based on ***PA***rtitioning at ***L***ongest ***D***imension (PALD) of requests but a best-fit approach is used to allocate requests and sub-requests.

The partitioning mechanism aims at (i) lifting the condition of contiguity, and (ii) at the same time maintaining *good* level of contiguity. Removing one from the longest dimension of a request is expected to produce two sub-requests one of which is relatively big and as close as possible to the square-shape and, thus; reducing communication latency caused by non-contiguity. Using extensive simulations, we evaluated the proposed strategies and compared them with previous contiguous and non-contiguous strategies. Simulation outcomes clearly show the proposed PALD-based schemes produce the best *Average Response Time* (ART), the *Average System Utilization* (ASU) and produce relatively low communication overhead.

REFERENCES

[1] B. S. Yoo and 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.

[2] 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, Academic Press, Inc. Orlando, FL, USA, July 1998, pp. 40-68.

[3] G.-M. Chiu and S.-K. Chen, "An efficient submesh allocation scheme for two-dimensional meshes with little overhead", IEEE Transactions on Parallel & Distributed Systems, vol. 10, no. 5, IEEE Press, Piscataway, NJ, USA, May 1999, pp. 471-486.

[4] 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.

[5] I. Ismail and J. Davis, "Program-based static allocation policies for highly parallel computers", Proc. IPCCC 95, IEEE Computer Society Press, Scottsdale, AZ, USA, 28-31 Mar 1995, pp. 61-68.

[6] K. H. Seo, "Fragmentation-Efficient Node Allocation Algorithm in 2D Mesh-Connected Systems", Proceedings of the 8th International Symposium on Parallel Architecture, Algorithms and Networks (ISPAN'05), IEEE Computer Society Press, Washington, DC, USA, 7-9 December, 2005, pp. 318-323.

[7] K. Li and K. H. Cheng, "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.

[8] K. Windisch, J. V. Miller, and V. Lo, "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.

[9] 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.

[10] L. M. Ni and P. K. McKinley. 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.

[11] P. Krueger, T. Lai, and 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.

[12] P. J. Chuang and N.-F. Tzeng, "Allocating precise submeshes in mesh connected systems", IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 2, IEEE Press, USA, February 1994, pp. 211-217.

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

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

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

[16] T. Srinivasan, J. Seshadri, A. Chandrasekhar, and 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.

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

[18] 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.

[19] W. Mao, J. Chen, and W. Watson, "Efficient Subtorus Processor Allocation in a Multi-Dimensional Torus", Proceedings of the 8th International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05), IEEE Computer Society, Washington, DC, USA, 30 November -3 December, 2005, pp. 53-60.

[20] X. Lin, P. Mckinly, and A. Esfahanina, 1993. Adaptive Multicast wormhole Routing in 2D-mesh multicomputers. Proceeding of Parallel Architecture and Language conference (PARLE), pp 228-241.

[21] 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, 1992, pp. 328-337.

 ISBN: 978-1-61208-101-4