Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 11 (6): 943-952, 2011
ISSN 1812-5654 / DOI: 10.3923/jas.2011.943.952

Processor Allocation with Reduced Internal and External Fragmentation in 2D Mesh-based Multicomputers

S. Bani-Ahmad
Department of Information Technology, Al-Balqa Applied University, Jordan

Abstract: Internal and external fragmentation can significantly limit the performance of mesh-based multicomputer systems. Contiguous allocation strategies fail to reduce the effect of fragmentation and hence provide very limited performance. The Adaptive Non-Contiguous Allocation (ANCA) strategy solves the problem of fragmentation by allowing parallel jobs to be allocated non-contiguously. This is done by splitting the frame requested by the parallel job in hand into two subframes of equal sizes at the longest dimension of the request. This may result in having internal fragmentation problem. To remedy this problem, the ANCA strategy uses book keeping to keep tracking idle nodes. In this study, the ANCA strategy is revisited with a new implementation and with more exhaustive simulation-based evaluation. Further, the idea of preventing over-splitting of parallel requests is studied with the ANCA algorithm. In the proposed implementation, over-partitioning is avoided by placing a limit to maximum number of non-contiguous blocks that can be assigned to any parallel job. This maximum number is referred to as the partitioning-bound. Our experimental results shows that having this partitioning-bound parameter can make the ANCA allocation strategy flexible and tunable as it allows the allocator module to choose an optimal partitioning-bound value while allowing parallel jobs to be allocated early. Our experimental results also showed that the ANCA allocation strategy could sustain higher system and communication loads compared to other non-contiguous allocation strategies, namely; the MBS and Paging allocation strategies.

Key words: Allocation strategy, non-contiguous allocation, contiguous allocation, 2D-mesh multicomputers, request-partitioning

INTRODUCTION

Multicomputer systems are cost-effective alternatives of the traditional supercomputers (Ababneh, 2006; Bani-Mohammad *et al*., 2006; 2007). The interconnection of multicomputers come in different styles called topologies. The two-dimensional and three-dimensional mesh-based topologies are probably the most common topologies. The reason is that mesh architecture is simple, regular and scalable. Several recent commercial and experimental parallel computers have been built based these two architectures such as the IBM BlueGene/L and the Intel Paragon (Bani-Ahmad, 2010a,b; Ababneh, 2006).

Contiguous allocation strategies for mesh-connected multicomputers attempt to locate a contiguous portion of the computing units for the execution of a parallel job (Bani-Ahmad, 2010a,b; Ababneh, 2006; Yoo and Das 2002; Chiu and Chen, 1999; Ismail and Davis, 1995; Li and Cheng, 1991). The goal behind contiguity is (i) to minimize the distance of interprocessor communication path and (ii) to avoid the interprocess interference. In contiguous processor allocation, all the processors allocated to a parallel job retain the same topology as the multicomputer system with the number of processors determined according to the requirement of individual parallel jobs. Thus, in a mesh-connected multicomputer, jobs are allocated to submeshes (Bani-Mohammad *et al*., 2006; 2007; Chiu and Chen, 1999). A parallel job retains all the nodes of the submesh for the entire duration of its life time. Once a parallel job is allocated, it runs till completion (Ababneh, 2006; Bani-Mohammad *et al*., 2006, 2007).

The processor allocator module in a multicomputer is responsible for detecting and assigning free multicomputers to parallel jobs. Allocation strategies with better recognition ability for available submeshes of free multicomputers can improve the chance of assigning a job into the system and, thus, reduce the waiting delay (Ababneh, 2006; Bani-Mohammad *et al*., 2006, 2007).

Studies by Yu and Das (1994); Chang and Mohapatra (1998); Bani-Ahmad (2010a,b) and Ababneh (2006) showed that a significant performance improvement cannot be obtained by refining the contiguous allocation strategies. Because of fragmentation problem, the utilization of a parallel system can significantly degrade (Ababneh, 2006). Fragmentation occurs when there are enough free multicomputers in the parallel system but the allocator module fails to allocate these multicomputers to the waiting parallel jobs. This in turn,

limits the performance of contemporary allocation schemes.

In this study, we evaluate an allocation strategy that tries to alleviate the fragmentation problem by performing non-contiguous. The proposed approach is a modification of the ANCA (Adaptive Non-Contiguous Allocation) strategy proposed in (Chang and Mohapatra, 1998). The ANCA strategy tries to allocate the parallel job in hand to a submesh of the same size and topology that is requested if available. If it fails, the ANCA strategy adaptively splits the parallel job into smaller subframes for allocation of equal sizes. These subframes are allocated simultaneously to non-contiguous locations. Because small subframes are usually easy to be successfully allocated, the probability of successfully allocating a parallel job is increased (Bani-Mohammad *et al*., 2006; 2007; Chang and Mohapatra, 1998).

In our implementation of the modified ANCA approach, the ANCA-based allocator tries to allocate the parallel job in hand using some given contiguous allocation strategy. In our experiments, we use the First Fit (FF) and the Best-Fit (BF) contiguous allocation strategies. Alternatively, any contiguous allocation strategy can be used. If the allocator module fails to allocate the parallel job, it splits the job into two subframes of equal sizes. 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, A, is even, 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 (Chang and Mohapatra, 1998) 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 (Chang and Mohapatra, 1998). This should make this implementation of the ANCA algorithm to be more complicated and both in time and space. In our implementation of the ANCA algorithm we divide the request in hand to 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 in our implementation of the ANCA algorithm.

Rong-Chang *et al*. (2009) show that the GA works quite well in dealing with the off-the-job training course assignment problem. In addition, the influences of variation of population size, the generation number, crossover rate and mutation rate on the solutions are tested. Different penalty values and different penalty functions are also tested and the results show that the combination of penalty function and penalty value can increase top choices and reduce choices outside the preference list.

To prevent over-partitioning of parallel jobs we propose and evaluate the idea of placing a bound on the maximum number of blocks into which a parallel job can be splitted into. This should help having better control over the negative effect of non-contiguity that may significantly increase the time spent in communication between the processes that belong to the same parallel job (Bani-Ahmad, 2010b).

We compare the performance of the proposed strategy along with other non-contiguous allocation strategies proposed earlier for the mesh-connected multicomputers, namely; the MBS (Lo *et al*., 1997) and Paging algorithms (Liu *et al*., 1995). Extensive simulations are conducted to study the tradeoffs of using the ANCA strategy.

In this study, the ANCA processor allocation strategy is revisited with a new implementation that guarantees no internal fragmentation problem. Our experimental evaluation is more exhaustive and critical compared to the experiment conducted in (Chang and Mohapatra, 1998). Further, the idea of preventing over-splitting or over-partitioning is studied with the ANCA algorithm. Over-partitioning is avoided by placing a limit to maximum number of non-contiguous blocks that can be assigned to any parallel job. This maximum number is referred to as the partitioning-bound. Our experimental results shows that having this partitioning-bound parameter can make the ANCA allocation strategy 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.

PREVIOUS ALLOCATION STRATEGIES

Processor allocation strategies have been extensively studied by several researchers. Contiguous allocation strategies focus on finding the requested submesh according to the request of a job. Non-contiguous allocation strategies eliminate the constraint of contiguity. Next we outline several allocation strategies proposed in the literature.

Contiguous allocation strategies

Two Dimensional Buddy (TDB): In the TDB strategy (Li and Cheng, 1991), the system is assumed to be a square with side lengths equal to a power of two. The size of a requested submesh is rounded up to a square with side-lengths as the nearest power of two. Obviously, a square submesh can form a larger square submesh with its

three neighboring buddies. Jobs are allocated to buddies of submeshes. This allocation strategy suffers from internal fragmentation because it only allocates square submeshes whose side lengths are equal to a power of two. In other words, because of rounding up the sides of requests, the allocated submesh can be larger than the actually requested submesh.

Frame Sliding (FS): The FS method (Chuang and Tzang, 1994) was proposed to reduce the fragmentation problem of the TDB allocation by allowing meshes of any arbitrary size to be allocated. Viewing the requested submesh of the job in hand as a frame, the FS algorithm slides the frame across the system to examine for a free submesh to execute the job.

The First-Fit (FF) and the Best-Fit (BF) schemes: The FF and BF algorithms proposed in (Zhu, 1992) guarantee the recognition of a free submesh, provided it exists. The two algorithms work by scanning the entire mesh for possible allocation.

Adaptive-Scan (AS) scheme: The adaptive-scan (Ding and Bhuyan, 1993) changes the orientation of the submesh being searched for if the required submesh in the original orientation is not available. Thus, the AS strategy has better recognition capabilities than that of the BS and FF schemes.

All the above allocation strategies consider only contiguous regions for the execution of a job and are referred to as contiguous allocations. Communication cost is expected to be minimal in contiguous allocations. On the other hand, the restriction that a parallel job has to be allocated to contiguous set multicomputers reduces the chance of successfully allocating the job due to the problem of fragmentation.

Non-contiguous allocation strategies: Hardware advances such as wormhole routing and faster switching techniques have made the communication latency less sensitive to the distance between the communicating nodes (Ni and McKinley, 1993; Lin *et al*., 1993; Chang and Mohapatra, 1998; Ababneh, 2006; Bani-Mohammad *et al*., 2006, 2007). This makes allocating a parallel job to a non-contiguous set of multicomputers plausible. By alleviating the restriction of contiguity, jobs can get allocated executed without waiting if the number of available multicomputers is higher or equal to the required number of multicomputers. Several non-contiguous allocation algorithms are proposed in literature. Examples are: the random, the Multiple Buddy System (MBS) and the Paging algorithms.

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 scheduled 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 (Lo *et al*., 1997).

In the Paging allocation strategy (Liu *et al*., 1995), 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, namely; row-major, shuffled row-major, snake-like, or shuffled snake-like indexing.

TOTALLY AND PARTIALLY NON-CONTIGUOUS STRATEGIES

In general, non-contiguous allocation algorithms can be classified into two categories based on the level of non-contiguity they cause to allocated parallel jobs. These categories are: (i) totally non-contiguous and (ii) partially non-contiguous. 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.

A number of studies have showed that partially non-contiguous allocations that maintain some degree of contiguity can successfully provide higher performance than the non-contiguous allocations that highly disperses parallel jobs (Liu *et al*., 1995; Min and Mutka, 1995). The reason for that is the effect of message contention and communication cost can be lowered if some level of contiguity is secured. This is despite the fact that communication latency nowadays is expected to be less sensitive to the distance of the communication rout traversed by messages. Notice that the contention at common links amongst different messages causes extra delay that result in higher communication costs.

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 (Liu *et al*., 1995). In MBS, contiguous allocation is explicitly sought only for requests with sizes of the form 2^{2n}, where n is a positive integer.

THE ANCA AND BGP APPROACHES

To preserve some level of contiguity within allocated parallel jobs, the ANCA algorithm first attempts to 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 equi-sized sub-requests and then ANCA tries to map these sub-requests to available locations (Chang and Mohapatra, 1998).

An issue with the unbounded ANCA strategy is that it can disperse the allocated sub-meshes more than it is necessary through over partitioning (Bani-Mohammad *et al*., 2007). In this study, we propose placing a bound on the maximum numbers of non-contiguous blocks that can be assigned to a parallel job. Thus, the proposed bounded ANCA approach can reduce the communication overhead.

As have been mentioned earlier, maintaining a good level of contiguity can prove useful in non-contiguous allocation (Bani-Ahmad, 2010a,b). 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 (Lo *et al*., 1997; Bani-Ahmad, 2010a,b). In MBS, contiguous allocation is explicitly sought only for requests with sizes of the form 2^{2n}, where n is a positive integer (Lo *et al*., 1997; Bani-Mohammad *et al*., 2007, Bani-Ahmad, 2010b).

In (Bani-Ahmad, 2010a,b; Bani-Mohammad *et al*., 2007), 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 $\alpha x \beta$ and assuming $\beta > \alpha$, the two partition-sizes are $\alpha x (\beta-1)$ and $\alpha x 1$ 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. The abbreviation PALD stands for PArtitioning at the Longest Dimension (Bani-Ahmad, 2010a,b).

Studies show that geometry of allocated nodes (i.e., relative positions of the nodes) can be another factor for message contention (Chang and Mohapatra, 1998). Usually, parallel applications are optimized for the communication pattern in the underlying architecture being used. Thus, proper mapping of processes onto processors is critical to the communication latency (Sherry and Ni, 1996). Consequently, changing the submesh geometry due to non-contiguous allocation using the MBS, Paging and PALD-based allocation strategies may introduce communication overhead due to the change in the optimized communication pattern (Chang and Mohapatra, 1998). In such a scenario, the ANCA approach that we study in this study is preferred from this point of view over the other non-contiguous allocation strategies. This is because the ANCA approach tries to retain the geometry among the allocated processors as much as possible (Chang and Mohapatra, 1998).

Another advantage of the ANCA approach is that this strategy utilizes physically fragmented processing units in a mesh. This results in allocating parallel jobs early and, thus, to avoid unnecessary waiting. Of coarse, allocating parallel job into non-contiguous set of blocks of processing nodes results in an increase in execution time and the communication latency. However, this increase in execution time and communication cost is less significant than the decrease in queuing (waiting) delay, the total turnaround time of a parallel job can be reduced (Chang and Mohapatra, 1998).

Next, we list the following conclusions regarding the basic advantages of the ANCA approach over the other non-contiguous allocation algorithms (Chang and Mohapatra, 1998). (i) The ANCA algorithm does not choose to non-contiguously allocate a job is contiguous allocation is possible. The ANCA allocation chooses non-contiguous allocation only when fragmentation prevents a waiting parallel job from being allocated. In this case, non-contiguous allocation is chosen to reduce job queuing time. (ii) If the ANCA algorithm has to allocate a parallel job non-contiguously, it tries to maintain some degree of contiguity to avoid an increase over-partitioning. (iii) The geometry of the processing units allocated to a parallel job is retained as much as possible by the ANCA approach to avoid the extra communication overhead caused by disrupting the optimal communication pattern. (iv) ANCA is flexible as the degree of non-contiguity can be controlled through controlling the partitioning-bound value.

The space complexity of our implementation of the ANCA algorithm is the same of the contiguous allocation

strategy being used. It is the same as the first-fit and the best-fit algorithms. Similar to the ANCA implementation in (Chang and Mohapatra, 1998), the time complexity for ANCA in one iteration is slightly worse than the first-fit/best-fit algorithms because of the extra time taken for the marking the candidate subframes. But the performance of the system is expected to be much better in terms of (i) mean response time and (ii) percent system utilization.

PERFORMANCE EVALUATION

Simulation environment: 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 here. The proposed allocation algorithm is implemented in the C language and later integrated with the ProcSimity simulation tool (Windisch *et al*., 1995; ProcSimity, 1997). 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 (Suzaki *et al*., 1996; Grama *et al*., 2003; Lo *et al*., 1997). 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 (Suzaki *et al*., 1996). In this paper we only report the results of our experiments using all-to-all communication pattern.

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.

Simulation output interpretation

Mean Response Time (MRT): The response time is the time from the submission of request until the first real response produced for jobs.

Mean Job Service Time (JST): The service time of a parallel job is the time from the allocation of the job's request until the moment the parallel job finishes execution.

Percent System Utilization (PSU): The average of keeping the processors within a system as busy as possible, this value between 0 and 1.

Mean Packet Blocking Time (MPBT): 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.

Mean Packet Latency (MPL): The average of the time that all packets within job will be sent between processors (From source processor to distention processor).

Mean number of Blocks Per Job (MBPJ): The average number of non-contiguous subframes assigned to the parallel job that were served during simulation.

Simulation results and observations

The impact of system load on the performance of the ANCA-based and the other tested non-contiguous allocation strategies: The ANCA allocation strategy differs from existing non-contiguous allocation strategies (e.g., MBS and Paging) in two major aspects. First, ANCA combines the advantages of both contiguous and non-contiguous allocation schemes. ANCA only allocates parallel jobs non-contiguously when fragmentation occurs. Second, ANCA tries to preserves the geometry of processing units in parallel jobs.

Figure 1 shows how the increase in system load can affect the performance of bounded-ANCA allocation strategy compared to the MBS strategy in terms of the mean number of blocks per job (MBPJ) factor. It is clear from Fig. 1 that in general, the MBPJ increases as the

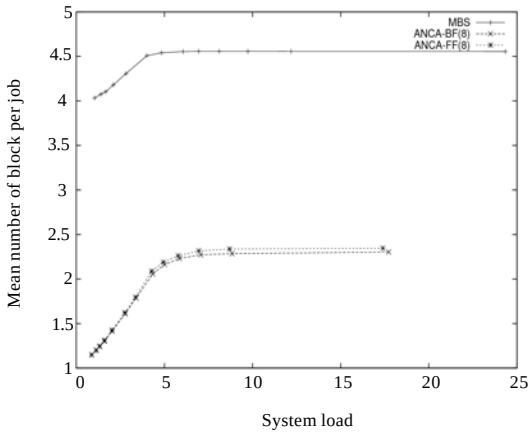

Fig. 1: Mean number of blocks per job vs system load in MBS, ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

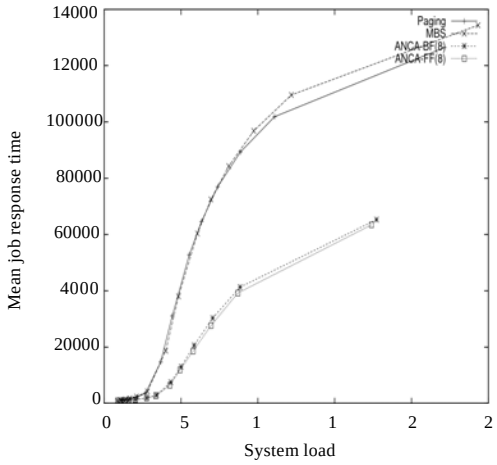

Fig. 2: Mean response time vs system load in MBS, Paging, ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

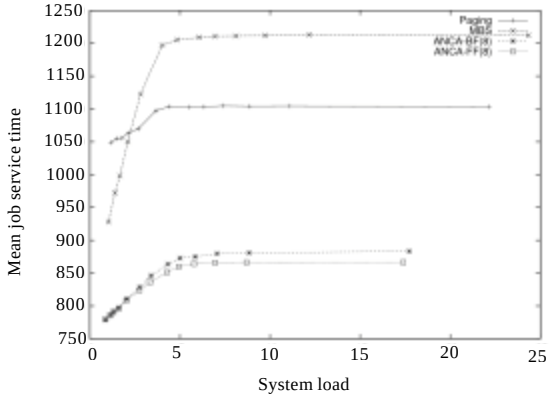

Fig. 3: Mean job service time vs system load in MBS, Paging, ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

system load increases. This is because the probability of finding idle processing units decreases as the system becomes more loaded. And thus the bounded-ANCA has to further disperse the parallel job in order to successfully allocate them. Placing a partitioning-bound can help in keeping the MBPJ factor in control.

Figure 2 shows the effect of increasing the load of the system on the mean response time performance (MJRT) criteria. It is clear from Fig. 2, that the bounded-ANCA algorithm produces significantly less MJRT than the MBS and Paging allocation strategies.

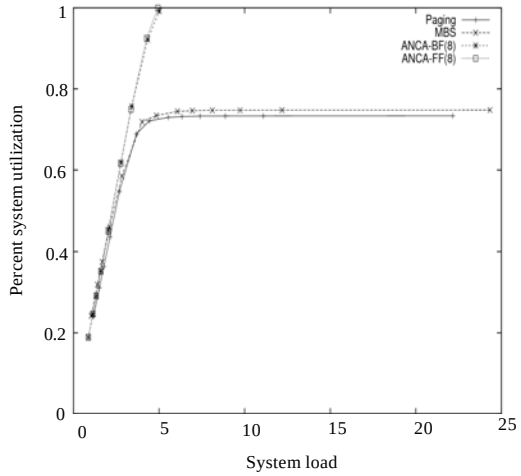

Fig. 4: Percent system utilization vs system load in MBS ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

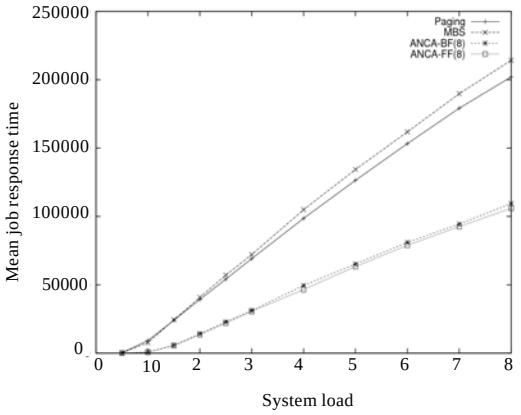

Fig. 5: Mean job response time vs communication load (represented by the number of messages per parallel job) in MBS ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

Figure 3 shows how the Mean Job Service Time (MJST) performance factor changes when the load of the multicomputer system increases. In general, the increase of system load increases the MBPJ factor. This, in turn, increases the communication cost and thus causes the service time of jobs to increase. This increase, however, is lower than the queuing time of the parallel jobs. Consequently, the overall turnaround time of parallel jobs is less even if they get allocated non-contiguously.

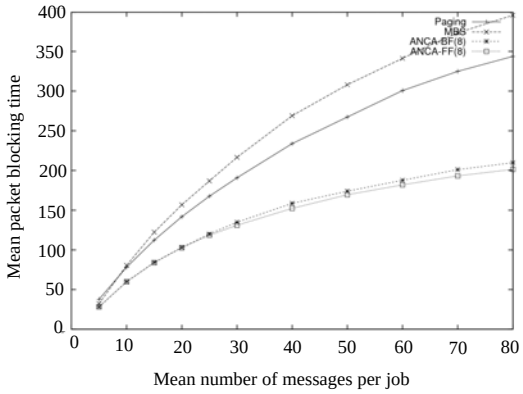

Fig. 6: Mean packet blocking time vs communication load (represented by the number of messages per parallel job) in MBS ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

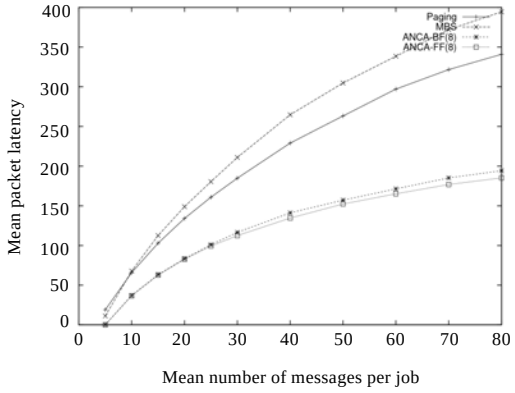

Fig. 7: Mean packet latency vs communication load (represented by the number of messages per parallel job) in MBS ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

Fig. 3 shows that the bounded-ANCA algorithm is more scalable than the MBS and Paging algorithms and can produce higher percent system utilization as shown in Fig. 4.

The effect of communication load on the performance of the ANCA-based and the other tested non-contiguous allocation strategies: Figure 5 shows how the MJRT change as the communication load the system increases. It is clear that the MJRT increase in all tested allocation algorithms. However, the rate of increase in MJRT when

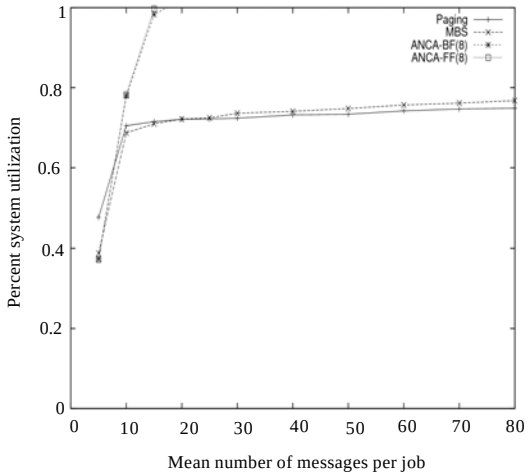

Fig. 8: Percent system utilization vs communication load (represented by the number of messages per parallel job) in MBS ANCA-FF and ANCA-BF allocation strategies with partitioning bound of 8 BPJ under the FCFS scheduling mechanism and all-to-all communication pattern

applying the bounded-ANCA algorithm is considerably less.

Figure 6 shows how the increase in the communication load of the system affects the Mean Packet Blocking Time (MPBT). In general, more messages implies higher communication load. This means that the message contention also gets higher. This in turn increases the MPBT because of the collisions that occur at the common links between the paths of messages. This also increases the Mean Packet Latency (MPL) as shown in Fig. 7. It is clear from Fig. 6 and 7 that the bounded-ANCA algorithm can sustain higher communication loads than the MBS and Paging approaches. Further, the bounded-ANCA approach can better utilize the underlying parallel system under high communication loads as it is clear from Fig. 8.

The effect of partitioning bound on the performance of the ANCA-based allocation strategies: In this subsection we study the effect of the partitioning bound of the bounded-ANCA algorithm on its performance.

Figure 9 shows how the MBPJ performance factor gets affected by increasing the Partitioning-Bound Value (PBV). As expected, raising the PBV factor allows the bounded-ANCA algorithm to further partition parallel jobs when needed. And since Fig. 9- 12 are all performed on a loaded system, raising the PBV factor increases the MBPJ value. This in turn increases the MJRT and the MJST performance factors as illustrated in Fig. 10 and 11, respectively.

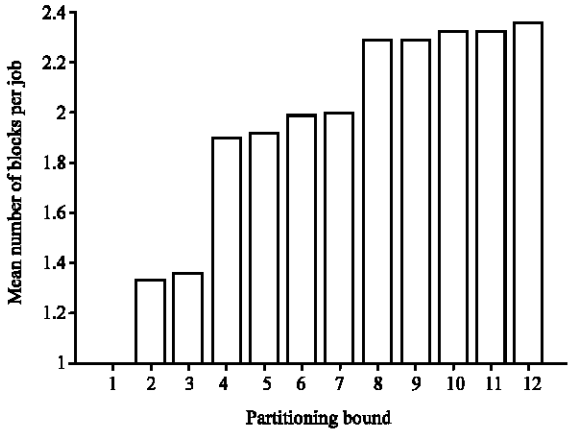

Fig. 9: Mean number of blocks per job vs partitioning bound in MBS ANCA-FF and ANCA-BF allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern

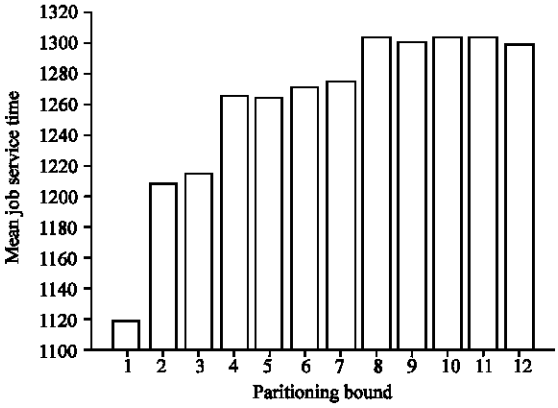

Fig 11: Mean job service time vs partitioning bound in MBS ANCA-FF and ANCA-BF allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern

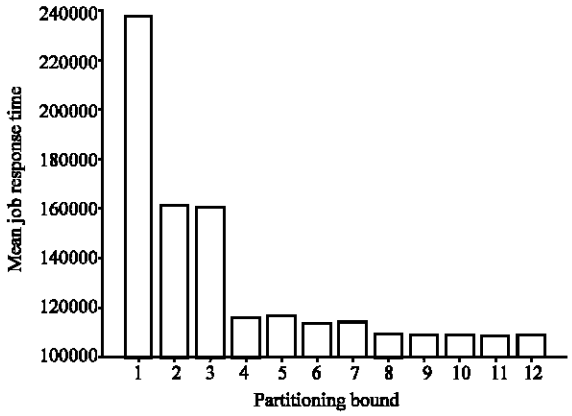

Fig. 10: Mean job response time vs partitioning bound in MBS ANCA-FF and ANCA-BF allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern

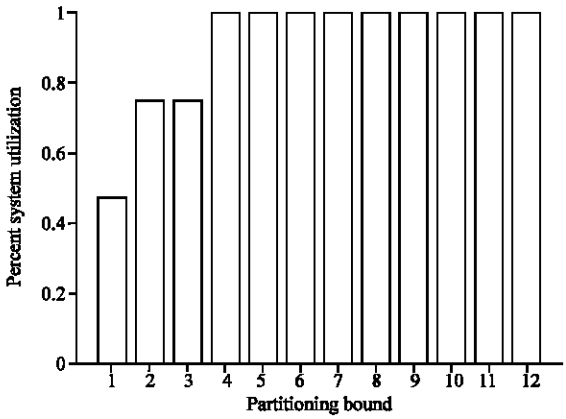

Fig. 12: Percent system utilization vs partitioning bound in MBS ANCA-FF and ANCA-BF allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern

Figure 12 shows how increasing the PBV affect the Percent System Utilization (PSU) performance factor. Figure 12 shows that the PSU of the system reached a maximum values at a PBV of 4. Increasing the PBV further can cause over-partitioning of some jobs with minor or no improvement in the performance of the system in terms of the PSU performance parameter.

THE RPB ALLOCATION ALGORITHMS PROJECT

This project aims at studying current and developing new Request-Partitioning Based (RPB) processor allocation algorithms in 2D- and 3D- mesh-based multicomputers. A request-partitioning-based allocation approach combines the desirable features of both contiguous and non-contiguous allocation strategies for 2D mesh-based parallel computers. A RPB strategy works by partitioning parallel requests at their longest dimension. The goal of this project is to propose and evaluate (through computer-based simulation) the performance of a number of allocation algorithms that are based on this approach.

CONCLUSION

Internal and external fragmentation can significantly limit the performance of mesh-based multicomputer

systems. Contiguous allocation strategies fail to reduce the effect of fragmentation and hence provide very limited performance. The Adaptive Non-Contiguous Allocation (ANCA) strategy solves the problem of fragmentation by allowing parallel jobs to be allocated non-contiguously. In this study, the ANCA strategy is revisited with a new implementation and with more exhaustive simulation-based evaluation. Further, the idea of preventing over-splitting or over-partitioning is studied with the ANCA algorithm. Over-partitioning is avoided by placing a limit to maximum number of non-contiguous blocks that can be assigned to any parallel job. This maximum number is referred to as the partitioning-bound. Our experimental results shows that having this partitioning-bound parameter can make the ANCA allocation strategy 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.

REFERENCES

Ababneh, I., 2006. An efficient free-list submesh allocation scheme for two-dimensional mesh-connected multicomputers. J. Syst. Software, 79: 1168-1179.

Bani-Ahmad, S., 2010a. Intra-job communication contention and request-partitioning-based allocation strategies in 2D-mesh multicomputers. Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming, December 18-20, 2010, Dalian, LiaoNing, P. R. China.

Bani-Ahmad, S., 2010b. Submesh allocation in 2D-mesh multicomputer: Partitioning at the longest dimension of requests. Proceedings of the 4th International Conference on Advanced Engineering Computing and Applications in Sciences, October 25-30, 2010, Florence, Italy.

Bani-Mohammad, S., M. Ould-Khaoua, I. Ababneh and L.M. Mackenzie, 2006. Non-contiguous processor allocation strategy for 2D mesh connected multicomputers based on sub-meshes available for allocation. Parallel Distributed Syst., 2: 1-6.

Bani-Mohammad, S., M. Ould-Khaoua, I. Ababneh and M.L. Mackenzie, 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, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.5481&rep=rep1&type=pdf.

Chang, C.Y. and P. Mohapatra, 1998. Performance improvement of allocation schemes for mesh-connected computers. J. Parallel Distributed Comput., 52: 40-68.

Chiu, G.M. and S.K. Chen, 1999. An efficient submesh allocation scheme for two-dimensional meshes with little overhead. IEEE Trans. Parallel Distributed Syst., 10: 471-486.

Chuang, P.J. and N.F. Tzeng, 1994. Allocating precise submesh in mesh-connected systems. IEEE Trans. Parallel Distributed Syst., 5: 211-217.

Ding, J. and L.N. Bhuyan, 1993. An adaptive submesh allocation strategy for two-dimensional mesh connected systems. Proceedings of International Conference on Parallel Process, Aug. 16-20, Syracuse, New York, USA., pp: 193-200.

Grama, A., G. Karypis, V. Kumar and A. Gupta, 2003. Introduction to Parallel Computing. 2nd Edn., Addison Wesley, Redwood City, California.

Ismail, I.M. and J.A. Davis, 1995. Program-based static allocation policies for highly parallel computers. Proceedings of the 14th Annual International Phoenix Conference on Computers and Communications, March 28-31, Scottsdale, AZ, USA., pp: 61-68.

Li, K. and K.H. Cheng, 1991. A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system. J. Parallel Distributed Comput., 12: 79-83.

Lin, X., P.K. Mckinly and A.H. Esfahanina, 1993. Adaptive multicast wormhole routing in 2D-mesh multicomputers. Proceedings of the 5th International PARLE Conference on Parallel Architectures and Languages Europe, June 14-17, Springer-Verlag, London, UK., pp: 228-241.

Liu, T., W.K. Huang, F. Lombardi and L.N. Bhuyan, 1995. A submesh allocation scheme for mesh-connected multiprocessor systems. Proc. Int. Conf. Parallel Process., 2: 159-163.

Lo, V., K.J. Windisch, W. Liu and B. Nitzberg, 1997. Non-contiguous processor allocation algorithms for mesh-connected multicomputers. IEEE Trans. Parallel Distributed Syst., 8: 712-726.

Min, D. and M.W. Mutka, 1995. Effects of job interactions due to job partitioning and placement. Proceedings of the 8th International Conference on Parallel and Distributed Computing Systems, Sept. 21-23, Orlando, Florida, pp: 262-267.

Ni, L.M. and P.K. McKinley, 1993. A survey of wormhole routing techniques in direct networks. Computer, 26: 62-76.

ProcSimity, 1997. ProcSimity V4.3 Users Manual. University of Oregon, Eugene, Oregon, USA..

Rong-Chang, C., C. Ting-Tsuen and F. Wei-Luen, 2009. Assignment of external off-the-job training courses to employees using genetic algorithm. Inform. Technol. J., 8: 147-155.

Sherry, Q.M. and L.M. Ni, 1996. The effects of network contention on processor allocation strategies. Proceedings of the 10th International Parallel Processing Symposium (IPPS), April 15-19, Honolulu, HI, pp: 268-268.

Suzaki, K., H. Tanuma, S. Hirano, Y. Ichisugi, C. Connelly and M. Tsukamoto, 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, Aug. 18-21, Springer-Verlag, London, UK., pp: 641-650.

Windisch, K., J.V. Miller and V. Lo, 1995. ProcSimity: An experimental tool for processor allocation and scheduling in highly parallel systems. Proceedings of the 5th Symposium on the Frontiers of Massively Parallel Computation, Feb. 6-9, McLean, Virginia, pp: 414-421.

Yoo, B.S. and C.R. Das, 2002. A fast and efficient processor allocation scheme for mesh-connected multicomputers. IEEE Trans. Comput., 51: 46-60.

Yu, C. and C.R. Das, 1994. Limit allocation: An efficient processor management scheme for hypercubes. Int. Conf. Parallel Process., 2: 143-150.

Zhu, Y., 1992. Efficient processor allocation strategies for mesh-connected parallel computers. J. Parallel Distributed Comput., 16: 328-337.