A Performance Comparison of the Contiguous Allocation Strategies in 3D Mesh Connected Multicomputers

S. Bani-Mohammad[1], M. Ould-Khaoua[1], I. Ababneh[2] and Lewis M. Mackenzie[1]

[1]Department of Computing Science
University of Glasgow, Glasgow G12 8QQ, UK.
{saad, mohamed, lewis}@dcs.gla.ac.uk

[2]Department of Computing Science
Al al-Bayt University, Mafraq, Jordan.
ismail@aabu.edu.jo

Abstract. The performance of contiguous allocation strategies can be significantly affected by the distribution of job execution times. In this paper, the performance of the existing contiguous allocation strategies for 3D mesh multicomputers is re-visited in the context of heavy-tailed distributions (e.g., a Bounded Pareto distribution). The strategies are evaluated and compared using simulation experiments for both First-Come-First-Served (FCFS) and Shortest-Service-Demand (SSD) scheduling strategies under a variety of system loads and system sizes. The results show that the performance of the allocation strategies degrades considerably when job execution times follow a heavy-tailed distribution. Moreover, SSD copes much better than FCFS scheduling strategy in the presence of heavy-tailed job execution times. The results also show that the strategies that depend on a list of allocated sub-meshes for both allocation and deallocation have lower allocation overhead and deliver good system performance in terms of average turnaround time and mean system utilization.

1 Introduction

The mesh has been one of the most common networks for recent multicomputers due to its simplicity, scalability, structural regularity, and ease of implementation [1, 6, 12]. Meshes are suited to a variety of applications including matrix computation, image processing and problems whose task graphs can be embedded naturally into the topology [25].

Efficient processor allocation and job scheduling are critical to harnessing the full computing power of a multicomputer [1, 4, 5, 28]. The goal of job scheduling is to select the next job to be executed while the goal of processor allocation is to select the set of processors on which parallel jobs are executed [1].

In distributed memory multicomputers, jobs are allocated distinct contiguous processor sub-meshes for the duration of their execution [1, 4, 5, 6, 7, 12, 28, 29].

Most existing research studies [1, 4, 6, 11, 12, 29] on contiguous allocation have been carried out mostly in the context of the 2D mesh network. There has been relatively very little work on the 3D version of the mesh. Although the 2D mesh has been used in a number of parallel machines, such as iWARP [2] and Delta Touchstone [8], most practical multicomputers, like the Cray XT3 [3], Cray T3D [19], and the IBM BlueGene/L [14], have used the 3D mesh and torus as the underlying network topology due to its lower diameter and average communication distance [27].

Most existing contiguous allocation strategies for the 3D mesh, mainly the early ones, have time complexities that grow linearly with the size of the mesh [5, 7, 28]. The recent contiguous allocation strategies have time complexities that can be less sensitive to the size of the mesh [20, 23]. They build lists of the busy sub-meshes with the goal of achieving time complexities that depend on the number of allocated sub-meshes instead of the mesh size [20, 23]. Time complexities in $O(m^2)$, where m is the number of allocated sub-meshes in the busy list, were achieved [20, 23]. An advantage of the busy-list approach is that the list of busy sub-meshes is often small even when the mesh size becomes large, which decreases the allocation overhead.

The efficacy of most contiguous allocation strategies has been assessed under the assumption of exponentially distributed execution times [4, 5, 6, 7, 11, 12, 20, 23, 28, 29], which may not reflect all possible practical scenarios. For instance, a number of measurement studies [9, 15, 16, 17, 26] have convincingly shown that the execution times of many computational jobs are characterised by heavy-tailed execution times; that is, there are typically many short jobs, and fewer long jobs. Heavy-tailed distributions capture this variability and behave quite differently from the distributions more commonly used to evaluate the performance of allocation strategies (e.g., the exponential distribution). In particular, when sampling random variables that follow heavy-tailed distributions, the probability of large observations occurring is non-negligible.

In this paper, the performance of the existing contiguous allocation strategies for 3D mesh-connected multicomputers is revisited in the context of heavy-tailed job execution times. Existing strategies were typically evaluated with the assumption of First-Come-First-Served (FCFS) job scheduling. In this paper, a Shortest-Service-Demand (SSD) scheduling strategy is also used because it is expected to reduce performance loss due to blocking. This strategy was found to improve performance significantly [10, 21, 22]. Also in this paper, the performance of allocation strategies is measured in terms of usual performance parameters [4, 5, 6, 7, 20, 21, 22, 23, 24, 28, 29] such as the average turnaround time and mean system utilization. Algorithmic efficiency is measured in terms of the mean measured allocation overhead that allocation and deallocation operations take per job. The results show that the performance of the allocation strategies degrades when the distribution of job execution times is heavy-tailed. As a consequence, an appropriate scheduling strategy should be adopted to deal with heavy-tailed execution times. Our analysis reveals that the SSD scheduling strategy exhibits superior performance than the FCFS scheduling strategy in terms of average turnaround time and mean system utilization.

The rest of the paper is organised as follows. The following section contains relevant preliminaries. Section 3 contains a brief overview of the allocation strategies compared in this study. Section 4 contains a brief overview of the scheduling

strategies considered. Simulation results are presented in Section 5, and Section 6 concludes this paper.

2 Preliminaries

The target system is a $W \times D \times H$ 3D mesh, where W is the width of the cubic mesh, D its depth and H its height. Each processor is denoted by a coordinate triple (x, y, z), where $0 \le x < W$, $0 \le y < D$ and $0 \le z < H$ [24]. A processor is connected by bidirectional communication links to its neighbour processors. The following definitions have been adopted from [4, 24].

Definition 1: A sub-mesh $S(w,d,h)$ *of width* w, *depth* d, *and height* h, *where* $0 < w \le W$, $0 < d \le D$ *and* $0 < h \le H$ *is specified by the coordinates* (x, y, z) *and* (x', y', z'), *where* (x, y, z) *are the coordinates of the base of the sub-mesh and* (x', y', z') *are the coordinates of its end, as shown in Fig. 1.*

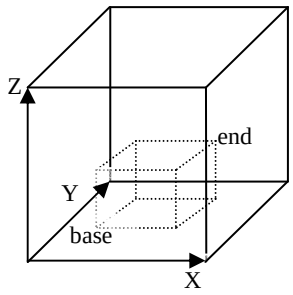

Fig. 1. A sub-mesh inside the 3D mesh.

Definition 2: The size of $S(w,d,h)$ *is* $w \times d \times h$.

Definition 3: An allocated sub-mesh is one whose processors are all allocated to a parallel job.

Definition 4: A free sub-mesh is one whose processors are all unallocated.

Definition 5: The list of all sub-meshes that are currently allocated to jobs and are not available for allocation to other jobs is called the busy list.

Definition 6: A prohibited region is a region consisting of nodes that can not be used as base nodes for the requested sub-mesh.

Definition 7: The Right Border Plane (RBP) of a sub-mesh $S(x_1, y_1, z_1, x_2, y_2, z_2)$ *with respect to a job* $J(\alpha \times \beta \times \gamma)$ *is defined as the collection of nodes with address* $(x_2 + 1, y', z')$ *where* $\max(y_1 - \beta + 1, 0) \le y' \le y_2$ *and* $\max(z_1 - \gamma + 1, 0) \le z' \le z_2$. *A RBP of sub-mesh* S *is a plane located just off the right boundary of* S.

3 Processors Allocation Strategies

Contiguous allocation has been investigated for 2D and 3D mesh-connected multicomputers [1, 4, 5, 6, 7, 11, 12, 28, 29]. The main shortcoming of the very few existing contiguous allocation strategies for the 3D mesh is that they achieve complete sub-mesh recognition capability with high allocation overhead. Below we describe some of the strategies that have been proposed for the 3D mesh.

First Fit (FF) and Best Fit (BF): In these two strategies [7], the free sub-meshes are scanned and FF allocates the first sub-mesh that is large enough to hold the job, whereas BF allocates the smallest suitable sub-mesh. Simulation results have shown that these two strategies have comparable performance in terms of average turnaround time and mean scheduling effectiveness; the performance of FF is close to that of BF, therefore we only consider the FF strategy for comparison in this paper. The strategies FF and BF are not recognition-complete. An allocation request is allocatable only if there is a large enough sub-mesh with the same orientation as the allocation request. Bit arrays are used for the scanning of available processors.

Turning First Fit (TFF) and Turning Best Fit (TBF): The problem of missing an existing possible allocation explained above is solved using TFF and TBF allocation strategies [7]. In these two strategies, turning the allocation request is used to improve the performance of contiguous FF and BF allocation in 3D mesh. The TFF and TBF allocation algorithms support the rotation of the job request. Let (a,b,c) be the width, depth and height of a sub-mesh allocation request. The six permutations (a,b,c), (a,c,b), (b,a,c), (b,c,a), (c,a,b) and (c,b,a) are, in turn, considered for allocation using the allocation strategy. If allocation succeeds for any of these permutations the process stops. For example, assume a free mesh (3, 3, 2) and the job requests (2, 3, 2) and (3, 2, 1) arrive in this order. The second job request cannot be allocated until it is rotated to (1, 3, 2). Simulation results have shown that the TFF strategy can greatly improve performance in terms of average turnaround time and mean scheduling effectiveness. Changing the orientation of allocation requests can alleviate external fragmentation. Moreover, the performance of TFF is almost identical to that of TBF; therefore the TFF strategy is considered for comparison in this paper. In [7], different scheduling strategies, such as First-Come-First-Served (FCFS) and Out-of-Order (OO) have been studied to avoid potential performance loss due to blocking.

The allocation and deallocation times of the algorithms proposed in [7] depend on the number of processors in the mesh system, n. The time complexity of the allocation algorithm is in $O(n^2)$, and the deallocation algorithm has time complexity in $O(n)$.

Busy List (BL) and Turning Busy List (TBL): In these strategies [20, 23], allocation is based on maintaining a busy list of allocated sub-meshes. The list is scanned to determine all prohibited regions. The prohibited region of job $J(\alpha\times\beta\times\gamma)$ with respect to an allocated sub-mesh $S(x_1,y_1,z_1,x_2,y_2,z_2)$ is defined as the sub-mesh represented by the address (x', y', z', x_2, y_2, z_2), where x' = max(x_1-α +1, 0), y' = max (y_1-β +1, 0) and z' = max (z_1-γ+1, 0). The sub-meshes (W-α+1, 0, 0, W-1, D-1, H-1),

(0, D-β+1, 0, W-1, D-1, H-1), and (0, 0, H-γ+1, W-1, D-1, H-1) are automatically not available for accommodating the base node of a free $\alpha \times \beta \times \gamma$ sub-mesh for $J(\alpha \times \beta \times \gamma)$, whether the nodes in these sub-meshes are free or not; otherwise, the sub-mesh would grow out of the corresponding mesh boundary plane (rightmost, deepest and highest planes) of $M(W, D, H)$. These three sub-meshes are called automatic prohibited regions of $J(\alpha \times \beta \times \gamma)$ and must always be excluded during the sub-mesh allocation process. A job $J(\alpha \times \beta \times \gamma)$ is allocatable if there exists at least one node that does not belong to any of the prohibited regions and the three automatic prohibited regions of $J(\alpha \times \beta \times \gamma)$.

All prohibited regions that result from the allocated sub-meshes are subtracted from each RBP of the allocated sub-meshes to determine the nodes that can be used as base nodes for the required sub-mesh size. Simulation results have shown that the performance of the allocation strategy in [20, 23] is at least as good as that of the existing allocation strategies. Moreover, the mean measured allocation time of these strategies is much lower than that of the existing strategies. The results have also revealed that the rotation of the job request improves the performance of the contiguous allocation strategies.

The allocation and deallocation times of the algorithms proposed in [20, 23] depend on the number of elements in the busy list, m. The time complexity of the allocation algorithms is in $O(m^2)$, and the deallocation algorithm has time complexity in $O(m)$. These allocation strategies maintain a busy list of m allocated sub-meshes. Thus, the space complexity of the allocation algorithms is in $O(m)$. This space requirement is small compared to the improvement in performance in terms of allocation overhead, as we will see in the simulation results. Also, this space requirement is small compared to the space requirement of FF, BF, TFF and TBF, which is in $O(n)$. An array is used for storing the allocation states of processors.

The time and space complexities of the allocation and deallocation algorithms considered in this paper are summarized in Table 1. Notice that the strategies that depend on a list of allocated sub-meshes for both allocation and de-allocation can entail smaller time complexity because m does not always depend on the size of the mesh for both allocation and deallocation. For job size distributions typically assumed in simulation studies (e.g., the uniform distribution used in [18]), the number of allocated sub-meshes remains small as the size of the mesh increases.

Table 1. Time and Space Complexity for Allocation and Deallocation Algorithms

Algorithm	Allocation Complexity	Deallocation Complexity	Space Complexity
TBL/BL	$O(m^2)$	$O(m)$	$O(m)$
TFF/FF	$O(n^2)$	$O(n)$	$O(n)$

m : Number of allocated sub-meshes in the busy list.
n : Total number of processors in the mesh.

4 Job Scheduling Strategies

The order in which jobs are scheduled first can have a considerable effect on the performance. In FCFS scheduling strategy, the allocation request that arrived first is considered for allocation first. Allocation attempts stop when they fail for the current FIFO queue head, while in SSD scheduling strategy, the job with the shortest service demand is scheduled first [10, 21, 22]. Any of them can start execution if its allocation request can be satisfied. Job scheduling has substantial effect on the performance of the allocation strategies. In [21, 22], the authors showed that the effect of the SSD scheduling strategy on the performance of the allocation strategies is substantially better than that of the FCFS scheduling strategy.

The performance of contiguous allocation strategies compared can be significantly affected by both a distribution adopted for job execution times and the scheduling strategy. To illustrate this, the performance of allocation strategies in this paper is evaluated in the context of heavy-tailed job execution time under both FCFS and SSD scheduling strategies. SSD scheduling strategy should be adopted to deal with heavy-tailed job execution times and to avoid potential performance loss due to blocking.

5 Simulation Results

Extensive simulation experiments have been carried out to compare the performance of the allocation strategies considered in this paper, with and without change of request orientation. Switching request orientation has been used in [5, 7, 20, 23, 28].

We have implemented the allocation and deallocation algorithms, including the busy list routines, in the C language, and integrated the software into the ProcSimity simulation tool for processor allocation in highly parallel systems [10, 18].

The target mesh is cube with width W, depth D and height H. Jobs are assumed to have exponential inter-arrival times. They are scheduled using First-Come-First-Served (FCFS) and Shortest-Service-Demand (SSD) scheduling strategies. The FCFS scheduling strategy is chosen because it is fair and it is widely used in other similar studies [6, 11, 12, 20, 21, 22, 23, 24], while the SSD scheduling strategy is used to avoid potential performance loss due to blocking [21, 22]. The execution times are modeled by a Bounded Pareto [13] (exhibiting a heavy-tailed property) as follows:

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/q)^{\alpha}} x^{-\alpha-1} (k \le x \le q)$$

where k and q are the lower and upper limit of job execution time, and α is a parameter that reflects the variability of job execution time. In the experiments, these parameters are set to: $k = 15.0$, $q = 4241.0$, and $\alpha = 1.0$ as suggested in [13].

Uniform distribution is used to generate the width, depth and height of job requests. The uniform distribution is used over the range from 1 to the mesh side length, where the width, depth and height of the job requests are generated

independently. This distribution has often been used in the literature [1, 4, 6, 7, 11, 12, 20, 21, 22, 23, 24, 28, 29]. Each simulation run consists of one thousand completed jobs. Simulation results are averaged over enough independent runs so that the confidence level is 95% that relative errors are below 5% of the means. The main performance parameters observed are the average turnaround time of jobs, mean system utilization and average allocation overhead. The turnaround time is the time that a parallel job spends in the mesh from arrival to departure. The utilization is the percentage of processors that are utilized over time. The allocation overhead is the time that the allocation algorithm takes for allocation and deallocation operations per job. The independent variable in the simulation is the system load. The system load is defined as the inverse of the mean inter-arrival time of jobs.

The notation <allocation strategy>(<scheduling strategy>) is used to represent the strategies in the performance figures. For example, TBL(SSD) refers to the Turning Busy List allocation strategy under the scheduling strategy Shortest-Service-Demand.

Figure 2 depicts the average turnaround time of the allocation strategies (TBL, TFF, BL, and FF) for the heavy-tailed and exponential job execution times under FCFS scheduling strategy. The simulation results in this figure are presented for a heavy system load. It can be seen in this figure that the performance of the allocation strategies degrades when the distribution of job execution times is heavy-tailed. For example, the average turnaround time of TBL(FCFS) under exponential job execution time is 49% of the average turnaround time of TBL(FCFS) under heavy-tailed job execution time, therefore, the SSD strategy should be adopted to deal with heavy-tailed job execution times as it avoids performance loss due FCFS blocking.

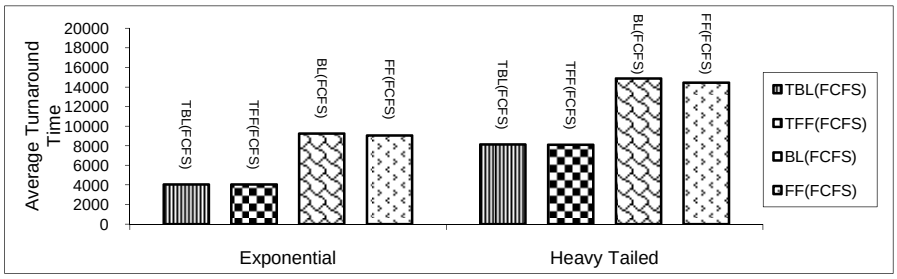

Fig. 2. Average turnaround time in BL, FF, TBL, and TFF under the exponential and heavy-tailed job execution times with FCFS scheduling strategy in an 8 × 8 × 8 mesh.

In Figure 3, the average turnaround time of jobs is plotted against the system load for both scheduling strategies considered in this paper. It can be seen in the figure that the strategies with rotation under SSD strategy (TBL(SSD) and TFF(SSD)) have almost identical performance, and that they are superior to all other strategies. They are followed, in order, by the strategies BL(SSD), FF(SSD), TBL(FCFS), TFF(FCFS), BL(FCFS), and FF(FCFS). When compared to TBL(SSD) and TFF(SSD), BL(SSD) increases the average turnaround times by about 31% and 57% for the loads 0.025 and 0.105 jobs/time unit, respectively. It can also be seen in the figure that the average turnaround times of the strategies that depend on the busy list is very close to that of the strategies that depend on the number of processors in the mesh system. For example, the average turnaround time of TBL(SSD) is very close to

that of TFF(SSD). However, the time complexity of the strategies that depend on the busy list (TBL and BL) is in $O(m^2)$ [20, 23], whereas it is in $O(n^2)$ for the other strategies (TFF and FF) [7]. The time complexity of TBL and BL does not grow with the size of the mesh as in TFF and FF. It can also be seen in the figure that the average turnaround time of the strategies with rotation is substantially superior to the strategies without rotation because it is highly likely that a suitable contiguous sub-mesh is available for allocation to a job when request rotation is allowed. It can also be noticed in the figure that the SSD strategy is much better than the FCFS strategy. This finding demonstrates that the scheduling and allocation strategies both have substantial effect on the performance of allocation strategies in the 3D mesh.

In Figure 4, the mean system utilization of the allocation strategies is plotted against the system loads for the two scheduling strategies considered in this paper. In this figure, TBL(SSD) and TFF(SSD) again have almost identical performance, and they are slightly superior to the other strategies. Also, these results show that switching request orientation improves performance substantially. This is indicated by the largely superior mean system utilization of the strategies that can switch the orientation of allocation requests (TBL(SSD), TBL(FCFS), TFF(SSD), and TFF(FCFS)) when they are compared to the strategies without rotation (BL(SSD), BL(FCFS), FF(SSD), FF(FCFS)). Moreover, the contiguous allocation strategies with rotation under SSD scheduling strategy achieve system utilization of 52%, but the contiguous allocation strategies without rotation can not exceed 42%. Also, higher system utilization is achievable under heavy loads because the waiting queue is filled very early, allowing each allocation strategy to reach its upper limits of utilization.

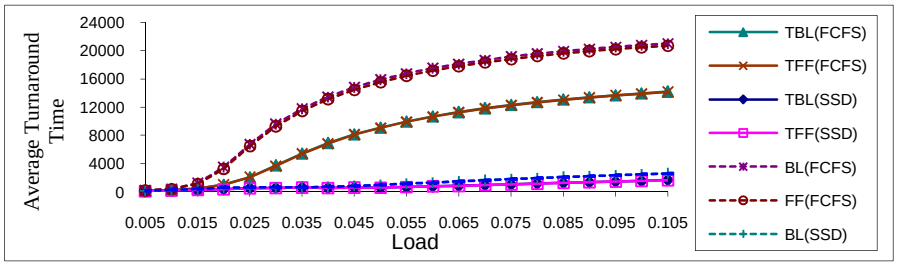

Fig. 3. Average turnaround time vs. system load in BL, FF, TBL, and TFF under the FCFS and SSD scheduling strategies in an 8 × 8 × 8 mesh.

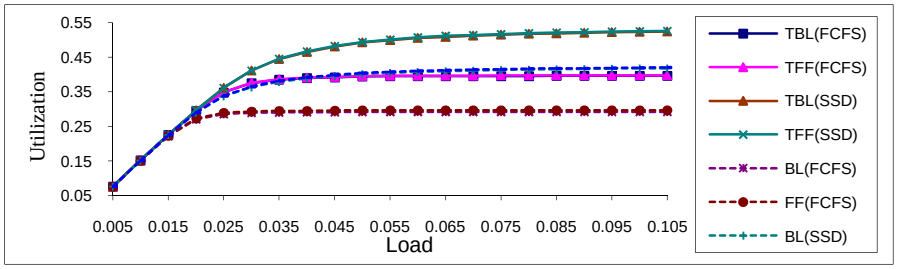

Fig. 4. Mean System utilization in BL, FF, TBL, and TFF under the FCFS and SSD scheduling strategies in an 8 × 8 × 8 mesh.

In Figure 5, the average number of allocated sub-meshes (m) in TBL is plotted against the system load for different mesh sizes under both FCFS and SSD scheduling strategies. It can be seen in the figure that the average number of allocated sub-meshes (m) is much lower than the number of processors in the mesh system (n). It can also be seen in the figure that for larger mesh sizes, the results show that m does not grow with n. It can also be noticed in the figure that the average number of allocated sub-meshes under SSD is higher than that under FCFS. In SSD, the job with the shortest service demand is scheduled first, meaning that allocation and deallocation operations are more numerous resulting in more allocated sub-meshes in the busy list.

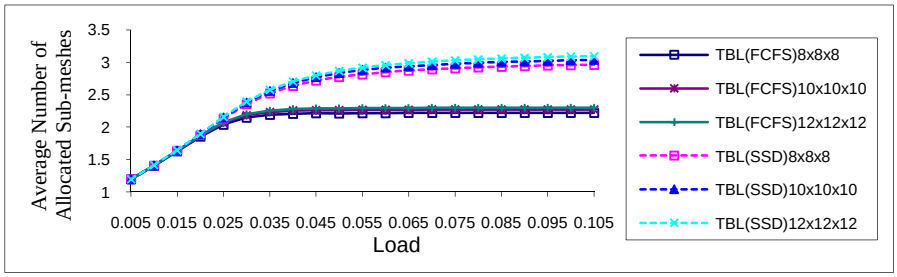

Fig. 5. Average number of allocated sub-meshes (m) in TBL under the FCFS and SSD scheduling strategies in 8 × 8 × 8 mesh, 10 × 10 × 10, and 12 × 12 × 12 mesh.

Figures 6 and 7 show the average allocation and deallocation time (allocation overhead) for the allocation strategies against the job arrival rate in an 8 × 8 × 8 mesh under the two scheduling strategies FCFS and SSD. We observe that the strategies that depend on the busy list (TBL, BL) take much smaller allocation overhead than the strategies that depend on the number of processors in the mesh system (TFF, FF) under both FCFS and SSD scheduling strategies. In Figure 6, for example, the time needed to carry out the allocation and deallocation operations of TBL(FCFS) strategy is 9% of the time taken by these operations in TFF(FCFS) strategy under the arrival rate 0.075 jobs/time unit. It can also be seen in the figures that the time needed for both allocation and deallocation for the strategies with rotation is higher than that of the strategies without rotation because in the worst case, the allocation process for the strategies with rotation, is repeated for all possible permutations (six permutations) of the job request while this process is repeated only one time for the other strategies. Moreover, it can be seen in the figures that the difference in allocation time gets much more significant as the system load increases. Thus, the strategies which depend on the busy list for both allocation and deallocation can be said to be more effective than the strategies that depend on the number of processors in the mesh system.

6 Conclusion and Future Directions

We have compared the performance of contiguous processor allocation strategies proposed for 3D mesh connected multicomputer for a wide range of system load and

system sizes when the distribution of job execution times is heavy-tailed (e.g. Bounded Pareto distribution). These allocation strategies cover a wide range of choices, including traditional First Fit (FF), Turning First Fit (TFF), Busy List (BL) approach that maintains a list of allocated sub-meshes to determine the regions consisting of nodes that cannot be used as base nodes for the requested sub-meshes, and Turning Bust List strategy (TBL), that attempts to maintain a good performance in terms of utilization, turnaround time, and allocation overhead.

In this study, the allocation overhead (i.e., allocation and deallocation time) is taken into account. A new scheduling strategies (SSD) has been used to deal with heavy-tailed job execution times to avoid performance loss due to blocking that results from largest jobs.

Simulation results have shown that the TBL(SSD) strategy is superior overall to all other strategies. It is as effective as the best competitor TFF(SSD) strategy, yet it is substantially more efficient. Moreover, the results have shown that the performance of the allocation strategies that depend on the number of allocated sub-meshes in the busy list (TBL and BL) is at least as good as that of the allocation strategies that depend on the number of processors in the mesh system in terms of average turnaround time and mean system utilization. The results have also shown that, the average allocation and deallocation time of the strategies that depend on the bust list (TBL and BL) is much lower than that of the other strategies that depend on, for both allocation and deallocation, the number of processors in the mesh system (TFF and FF). The results have also revealed that the rotation of the job request can greatly improve the performance of the contiguous allocation strategies. Moreover, the simulation results have shown that the effects of the SSD scheduling strategy on the performance of the allocation strategies is substantially better than that of the FCFS scheduling strategy in terms of performance parameters used in this study.

The busy list strategies (TBL and BL) can be efficient because it is implemented using a busy list approach. This approach can be expected to be efficient in practice because job sizes typically grow with the size of the mesh. The length of the busy list can be expected to be small, even when the size of the mesh scales up.

As a continuation of this research in the future, it would be interesting to implement the allocation strategies based on real workload traces from different parallel machines and compare it with our results obtained by means of simulations.

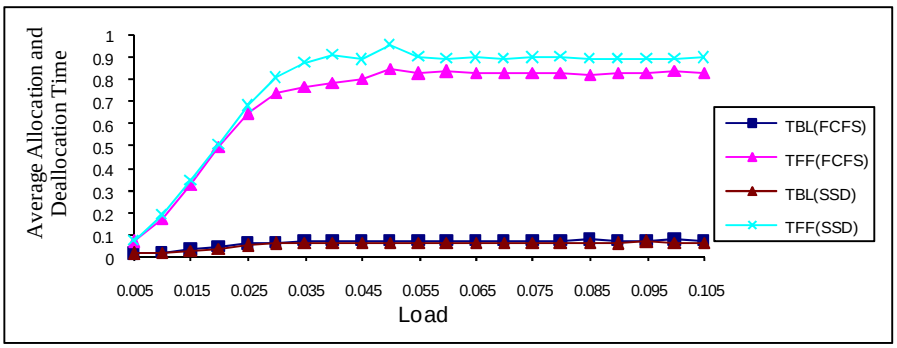

Fig. 6. Average allocation and deallocation times in TBL and TFF under the FCFS and SSD scheduling strategies in an 8 × 8 × 8 mesh.

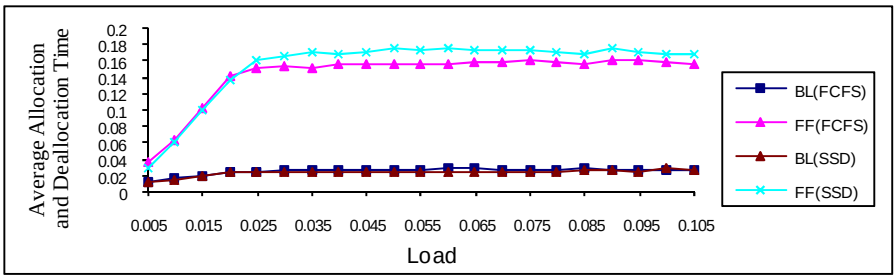

Fig. 7. Average allocation and deallocation times in BL and FF under the FCFS and SSD scheduling strategies in an 8 × 8 × 8 mesh.

References

[1] B.-S.Yoo, C.-R. Das, A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers, *IEEE Transactions on Parallel & Distributed Systems*, **51(1)** (2002) 46-60.

[2] C. Peterson, J. Sutton, P. Wiley, iWARP: a 100-MPOS, LIW microprocessor for multicomputers, *IEEE Micro*, **11(3)** (1991) 26-29, 81-87.

[3] Cray, Cray XT3 Datasheet (2004).

[4] G.-M. Chiu, S.-K. Chen, An efficient submesh allocation scheme for two-dimensional meshes with little overhead, *IEEE Transactions on Parallel & Distributed Systems*, **10(5)** (1999) 471-486.

[5] H.Choo, S.Yoo, H.-Y. Youn, Processor scheduling and allocation for 3D torus multicomputer systems, *IEEE Transactions on Parallel & Distributed Systems*, **11(5)** (2000) 475-484.

[6] I. Ababneh, An Efficient Free-list Submesh Allocation Scheme for two-dimensional mesh-connected multicomputers, *Journal of Systems and Software*, **79(8)** (2006) 1168-1179.

[7] I. Ababneh, Job scheduling and contiguous processor allocation for three-dimensional mesh multicomputers, *AMSE Advances in Modelling & Analysis*, **6(4)** (2001) 43-58.

[8] Intel Corporation, A Touchstone DELTA system description, (1991).

[9] J. Wei, X. Zhou, C-Z. Xu, Robust Processing Rate Allocation for Proportional Slowdown Differentiation on Internet Servers, *IEEE Transactions on Computers*, **54(8)** (2005) 964-977.

[10] 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)*, Washington, DC, USA, IEEE Computer Society Press, (1995) 414-421.

[11] 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, (2005) 318-323.

[12] K.-H. Seo, S.-C. Kim, Improving system performance in contiguous processor allocation for mesh-connected parallel systems, *The Journal of Systems and Software*, **67(1)** (2003) 45-54.

[13] L. He, S. Jarvis, D. Spooner, H. Jiang, D. Dillenberger, and G. Nudd, Allocating Non-Real-Time and Soft Real-Time Jobs in Multiclusters, *IEEE Transactions on Parallel and Distributed Systems*, **17(2)** (2006) 99-112.

[14] M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burow, T. Takken and P. Vranas, Design and Analysis of the BlueGene/L Torus Interconnection Network, *IBM Research Report RC23025*, IBM Research Division, Thomas J. Watson Research Center, Dec. 3, (2003).

[15] M. Harchol-Balter, The Effect of Heavy-Tailed Job Size. Distributions on Computer System Design, *Proceedings of ASA-IMS Conference on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*, Washington, DC, June (1999).

[16] Mark E. Crovella, Lester Lipsky, Long-Lasting Transient Conditions in Simulations with Heavy-Tailed Workloads, *Proceedings of the 1997 Winter Simulation Conference*, 7-10 Dec (1997) 1005-1012.

[17] Mor Harchol-Balter, Mark E. Crovella, Cristina D. Murta, On Choosing a Task Assignment Policy for a Distributed Server System, *Journal of Parallel and Distributed Computing*, **59(2)** (1999) 204-228.

[18] ProcSimity V4.3 User's Manual, University of Oregon, (1997).

[19] R.E. Kessler, J.L Swarszmeier, Cray T3D: a new dimension for Cray research, *Proc. CompCon*, (1993) 176-182.

[20] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh and Lewis M. Mackhenzie, An Efficient Turning Busy List Sub-mesh Allocation Strategy for 3D Mesh Connected Multicomputers, *Proceedings of the 7th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking & Broadcasting, (PGNET 2006)*, Liverpool John Moores University, UK, 26-27 June (2006) 37-43.

[21] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh and Lewis M. Mackhenzie, An Efficient Processor Allocation Strategy that Maintains a High Degree of Contiguity among Processors in 2D Mesh Connected Multicomputers, *2007 ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2007)*, IEEE Computer Society Press, Amman, Jordan, 13-16 May (2007).

[22] 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, University of Glasgow, January (2007).

[23] S. Bani-Mohammad, M. Ould-Khaoua, I. Ababneh, and L. Machenzie, A Fast and Efficient Strategy for Sub-mesh Allocation with Minimal Allocation Overhead in 3D Mesh Connected Multicomputers, *Ubiquitous Computing and Communication Journal*, ISSN 1992-8424, **1** (2006).

[24] 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)*, Minneapolis, Minnesota, USA, IEEE Computer Society Press, **2** (2006) 41-48.

[25] V. Varavithya, Multicasting in wormhole routed multicomputers, Ph.D. Thesis, Department of Electrical and Computer Engineering, Iowa State University, (1998).

[26] Vahid Tabatabaee, Ananta Tiwari, Jeffrey K. Hollingsworth, Parallel Parameter Tuning for Applications with Performance Variability, *SC'05*, Seattle WA, (2005).

[27] W. Athas, C. Seitz, Multicomputers: message-passing concurrent computers, *IEEE Computer*, **21(8)** (1988) 9-24.

[28] W. Qiao, L. Ni, Efficient processor allocation for 3D tori, *Technical Report*, Michigan State University, East Lansing, MI, 48824-1027, (1994).

[29] Y. Zhu, Efficient processor allocation strategies for mesh-connected parallel computers, *Journal of Parallel and Distributed Computing*, **16(4)** (1992) 328-337.