Simulation Modelling Practice and Theory 80 (2018) 19–31

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Improving system performance in non-contiguous processor allocation for mesh interconnection networks

Saad Bani-Mohammad*, Ismail Ababneh

Department of Computer Science Prince Hussein Bin Abdullah Faculty of Information Technology, Al al-Bayt University, Mafraq, 25113, Jordan

ARTICLE INFO

Article history:
Received 20 June 2016
Revised 9 September 2017
Accepted 2 October 2017

Keywords:
Paging
Multicomputers
Fragmentation
System utilization
Turnaround time
External message interference
Performance comparison
Simulation

ABSTRACT

Contiguous allocation of parallel jobs in multicomputers usually suffers from the degrading effects of fragmentation, because it requires that the allocated processors be contiguous and have the same topology as that of the multicomputer's interconnection network. Fragmentation can be avoided by adopting non-contiguous allocation. However, non-contiguous allocation can increase the distances among processors allocated to a job, which can increase the interference among messages of different jobs and increases message contention and delays. This paper suggests a new non-contiguous processor allocation strategy, referred to as Minimum Interference Paging (MIP), for the 2D mesh network. MIP attempts to reduce the distances among processors allocated using a paging variant that chooses a set of processors with the lowest distance between the first and last allocated processors, where the distance is the number of processors between the first allocated node and last allocated node. Using software simulation, we compared the performance of MIP against that of several well-known non-contiguous allocation strategies: paging (0), Multiple Buddy System (MBS) and Adaptive Non-contiguous Allocation (ANCA). The comparative evaluation was conducted using First-Come-First-Served (FCFS) job scheduling, wormhole routing and six communication patterns. These are the one-to-all, all-to-all, Near Neighbour, Ring, Divide and Conquer Binomial Tree (DQBT), and Random communication patterns. The results show that MIP exhibits superior performance in terms of average turnaround time of jobs.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

In a multicomputer, processor allocation is responsible for selecting the set of processors on which a parallel job is executed. Most strategies that have been suggested for multicomputers are based on *contiguous* allocation, where the processors allocated to a parallel job are physically contiguous and have the same topology as the network topology connecting the processors [4,5,10,13,16,22,26,27]. These strategies result in high external processor fragmentation [2,18–20,24,27]. External fragmentation occurs when there are free processors sufficient in number to satisfy the number requested by the parallel job selected for execution, but they are not allocated to it because the free processors are not contiguous or they do not have the same topology as the network topology connecting these processors.

* Corresponding author.
E-mail addresses: bani@aabu.edu.jo (S. Bani-Mohammad), ismael@aabu.edu.jo (I. Ababneh).

https://doi.org/10.1016/j.simpat.2017.10.001
1569-190X/© 2017 Elsevier B.V. All rights reserved.

Several studies have attempted to reduce such fragmentation [2,7,8,18,20,22,24]. One suggested solution is to adopt *non-contiguous* allocation [2,18,20,24]. In such a strategy, a job can execute on multiple disjoint smaller sub-networks rather than always waiting until a single sub-network of the requested size and shape is available. Although non-contiguous allocation increases message contention in the network, lifting the contiguity condition reduces processor fragmentation and increases processor utilization [18,20,24]. Folding has also been proposed for the 2D mesh [2,7]. Folding permits applications to execute on fewer processors than they have requested, when necessary. This could improve the performance of contiguous and non-contiguous allocation, as demonstrated in [2,7]. The problem with folding is that it requires that jobs be able to execute on a number of processors determined at load time. L-Shaped sub-mesh allocation (LSSA) algorithm proposed in [22] has also been suggested for 2D mesh to reduce external fragmentation. LSSA algorithm is contiguous and is based on L-shaped allocation, which improves the system performance as demonstrated in [22]. The problem with LSSA is that it suffers from external fragmentation as compared to other non-contiguous allocation algorithms proposed in [2,18,20,24].

The interference among messages of running jobs can be high in non-contiguous allocation strategies [2,24]. This is because these strategies can disperse the allocated processors more than it is necessary, which in turn increases the interference among messages and message contention. This degrades the system performance in terms of both average turnaround time of jobs and system utilization.

In this paper, we propose a new non-contiguous allocation strategy, referred to as Minimum Interference Paging (MIP), for the 2D mesh. MIP reduces the distances among the allocated processors, where the distance is the number of processors between the first allocated processor and the last allocated processor. This is achieved by choosing a set of processors with the lowest distance between the first allocated node and the last allocated node. This aims to reduce the interference among messages and improve system performance in terms of both average turnaround time of jobs and system utilization. Our proposed MIP algorithm differs from the LSSA algorithm proposed in [22] in that MIP is non-contiguous algorithm and it is free from external fragmentation while LSSA algorithm is contiguous algorithm and it suffers from external fragmentation. Using detailed simulations, the performance of MIP is compared against the performance of the non-contiguous allocation strategies paging (0) [24], MBS [24], and ANCA [2]. These strategies have been selected because they have been shown to perform well in [2,20,24].

The system model assumed in this paper is a mesh network, which is the underlying interconnection network in a number of practical parallel machines, such as iWARP [1], Delta Touchstone [12], MIT J-Machine [28], Cray T3D [17], and Cray T3E [3]. In this model, the processors are homogenous, the mesh is two-dimensional, and wormhole switching is assumed. As in other simulation studies [2,4–11,18–22,24,27], the results are applicable when the system and application models are relevant. To validate our results, realistic values assumed in previous related works [4,6,9,11,18,19,20,22,24,27] have been adopted.

The rest of the paper is organised as follows. Section 2 contains a brief summary of allocation strategies previously proposed for mesh multicomputers. Section 3 contains the proposed non-contiguous allocation strategy. Section 4 compares the performance of the non-contiguous allocation strategies considered in this research. Section 5 concludes this study.

2. Related work

This section provides a brief overview of some existing non-contiguous allocation strategies that have been suggested for the 2D mesh.

Hardware advances such as wormhole routing and faster switching techniques have made the communication latency less sensitive to the distance between the communicating nodes [2,20,23]. This has made allocating a job to non-contiguous processors plausible. This allows jobs to start execution earlier, provided that the number of free processors is sufficient [2,20].

2.1. Paging allocation strategy

In this strategy [24], the entire 2D mesh is divided into square pages that are sub-meshes with side lengths of 2^{size_index}, where *size_index* is a non-negative integer. A page is the allocation unit. The pages are indexed according to several indexing schemes (row-major, shuffled row-major, snake-like, and shuffled snake-like indexing). Fig. 1 shows the row-major indexing scheme. Busy arrays are used for scanning available pages. If the number of free pages is greater than or equal to the allocation request, the pages are scanned starting with page zero until the needed number of free pages is allocated. A paging strategy is denoted as paging(*size_index*). For example, paging(2) means that the page is a 4×4 sub-mesh. The number of pages a job requests is computed using the equation:

$$\mathrm{Pr}equest = \lceil (a \times b)/Psize \rceil \tag{1}$$

where *Psize* is the size of the page, and *a* and *b* are the side lengths of the requested sub-mesh.

Paging suffers from internal fragmentation when *size_index*> 0. The internal fragmentation of running jobs is computed using:

$$Internal\_Fragmentation = \sum_{jobs} \frac{Lost\_\mathrm{Pr}ocessors}{Allocated\_\mathrm{Pr}ocessors} \tag{2}$$

Download English Version:

https://daneshyari.com/en/article/4962605

Download Persian Version:

https://daneshyari.com/article/4962605

Daneshyari.com