ELSEVIER

Journal of Systems Architecture 48 (2002) 59–80

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

Design and analysis of static memory management policies for CC-NUMA multiprocessors ☆

Ravishankar Iyer [a], Hujun Wang [a,b,*], Laxmi Narayan Bhuyan [a,c]

[a] Intel Corporation, 15220 N.W. Greenbrier Parkway, Beaverton, OR 97006, USA
[b] Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, USA
[c] Department of Computer Science, University of California, Riverside, CA 92521-0304, USA

Abstract

In this paper, we characterize the performance of three existing memory management techniques, namely, buddy, round-robin, and first-touch policies. With existing memory management schemes, we find several cases where requests from different processors arrive at the same memory simultaneously. To alleviate this problem, we present two improved memory management policies called skew-mapping and prime-mapping policies. By utilizing the properties of skewing and prime, the improved memory management designs considerably improve the application performance of cache coherent non-uniform memory access multiprocessors. We also re-evaluate the performance of a multistage interconnection network using these existing and improved memory management policies. Our results effectively present the performance benefits of different memory management techniques based on the sharing patterns of applications. Applications with a low degree of sharing benefit from the data locality provided by first-touch. However, several applications with significant sharing degrees as well as those with single processor initialization routines benefit highly from the intelligent distribution of data provided by skew-mapping and prime-mapping schemes. Improvements due to the new schemes are found to be as high as 35% in stall time.

Keywords: Memory management; Interconnection networks; Execution-based simulation; Scientific applications; Shared-memory multiprocessor

1. Introduction

Cache coherent non-uniform memory access (CC-NUMA) systems have become extremely popular since they are scalable and provide transparent access to data. With multiple levels of caches, they certainly provide cached data at low latencies. However, once the data access gets beyond the layers of cache, these machines pay a high penalty and their performance deteriorates. Cache misses are made up of local and remote memory accesses. Local memory access latencies are usually a magnitude higher than cache access latencies. The access time to a remote memory in a large system could be several orders of magnitude higher than the cache access time because of the

☆ This research has been supported by NSF grant MIP 9622740.

[*] Corresponding author.
E-mail address: hwang@cs.tamu.edu (H. Wang).

1383-7621/02/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S1383-7621(02)00066-8

time spent in the interconnection network. Even with low miss rates, the bottleneck in the performance of NUMA systems lies in the remote memory access latencies. Effectively, the cache miss latencies depend heavily on the ratio of local and remote accesses. The memory management policy governs the placement of data in shared memory. It specifies which memory accesses would be local and which would be remote. In this paper, our main aim is to present different memory management policies and study their impact on the application performance and interconnection network performance of a CC-NUMA multiprocessor system.

Related work in this area can be divided into two categories. The first category has been the performance evaluation of memory management policies [1–6]. Most of these studies [1–4] focused on distributed shared memory systems without hardware cache coherence. The effect of different policies was studied for CC-NUMA systems in [5,6]. Verghese et al. [5] presented significant data on the OS/hardware support required for dynamic page migration and replication policies. However, their results also indicate that dynamic memory management policies improve the performance of parallel applications (the SPLASH workload) by only 4% over the static schemes. Bhuyan, et al. [6] presented the impact of existing memory management policies and switch design alternatives on the application performance. Hence we concentrate only on static memory management techniques. We propose two improved static schemes: (1) skew-mapping, (2) prime-mapping that significantly improve the performance of several applications over the existing schemes. The skew-mapping scheme is based on skewing pages which are allocated to memories using the round-robin policy. It can be specified as a function that maps logical pages onto memories such that the required pages could be accessed conflict-free. Several general classes of skewing memory data accesses have been investigated and characterized by previous researchers [7–10], but not yet been developed for CC-NUMA memory management. The prime-mapping scheme is based on the allocation of data pages to memories according to a prime number. The use of a prime number for effective distribution of data accesses has been studied in a few papers [11–13]. Lawrie [11] described a memory system designed for parallel array access which is based on the use of a prime number of memories in SIMD computers. Yang [12] presented a prime-mapped cache in the vector processing environment. The memory access logic of improved allocation schemes is similar to those of the existing policies, resulting in no additional delay for memory accesses. Further, the prime-mapped cache for vector computers was evaluated using a set of applications in [13]. However, they have not yet been developed for CC-NUMA multiprocessors.

The second category of related work is the performance evaluation of interconnection networks (IN's). Performance evaluation of IN's has been an active area of research for a long time [14–18]. These studies were conducted with a synthetic workload that is more suitable for a message passing or networking environment. Workload in a CC-NUMA environment is characterized by unsymmetrical bulky messages due to cache coherence, synchronization and memory management. Furthermore, due to the IN advancements, such as the use of virtual channels (VC's), the improvements in system performance have to be judged from the dynamic changes during the execution of applications. The aim of this paper is to re-evaluate the performance of an IN with realistic application data accesses and memory management policies governing the placement of the data blocks. Execution-based evaluations for IN's have been reported [19–21] to test the effectiveness of VC's. While these studies provide useful data, they do not explore different memory management techniques as we do. We consider a multistage interconnection network (MIN) in this paper, similar to the one employed in Butterfly and Cedar multiprocessors [22]. Unlike Cedar, we employ a NUMA organization with one network (like Butterfly) that is used for both forward and backward (reply) messages. We evaluate two different switch design alternatives (*simple wormhole* (*SWH*) *and buffered virtual channel* (*BVC*)) [20] for the MIN.

To evaluate the different memory management techniques in conjunction with different switch architectures, we have significantly modified our CC-NUMA simulator based on Proteus [23]. We