ELSEVIER

Microprocessors and Microsystems 26 (2002) 63–76

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

Performance analysis of a selectively compressed memory system

Jang-Soo Lee[a,*], Shin-Dug Kim[a], Charles Weems[b]

[a]Department of Computer Science, Yonsei University, 134 Shinchon-Dong, Seodaemoon-Ku, Seoul 120-749, South Korea
[b]Department of Computer Science, University of Massachusetts, Amherst, MA 01003-4610, USA

Received 20 January 2000; revised 1 November 2001; accepted 10 December 2001

Abstract

On-line data compression is a new alternative technique for improving memory system performance, which can increase both the effective memory space and the bandwidth of memory systems. However, decompression time accompanied by accessing compressed data may offset the benefits of compression. In this paper, a selectively compressed memory system (SCMS) based on a combination of selective compression and hiding of decompression overhead is proposed and analyzed. The architecture of an efficient compressed cache and its management policies are presented. Analytical modeling shows that the performance of SCMS is influenced by the compression efficiency, the percentage of references to the compressed data block, and the percentage of references found in the decompression buffer. The decompression buffer plays the most important role in improving the performance of the SCMS. If the decompression buffer can filter more than 70% of the references to the compressed blocks, the SCMS can significantly improve performance over conventional memory systems.

Keywords: On-line data compression; Compression algorithm; Compressed memory system; Cache architecture; Memory performance analysis

1. Introduction

As computing power increases, the memory space requirements of many programs, such as multimedia and 3D graphics applications, have increased dramatically, by as much as 50–100% per year. However, DRAM technology has not kept up with this requirement because its density has increased by just under 60% per year [4]. Furthermore, the performance gap between processor and memory is increasing steadily and has given rise to a phenomenon known as the 'memory wall' problem, which means that main memory access time is the primary obstacle in improving the overall performance of computer systems [11]. In addition, a disk access takes 10^5 times as long as a memory access and thus the time to load data from the disk becomes a significant portion of the overall execution time of a program [4]. To reduce these processor–memory and memory–disk performance gaps, conventional computer systems take advantage of a memory hierarchy. However, a long latency for access to lower level storage systems still occurs due both to relatively slower access and pin-bandwidth limitation [2].

Data compression, which has already been used at the lower level of the memory hierarchy such as the disk or network environment, is a new alternative method for reducing processor–memory and memory–disk performance gaps. Two advantages can be obtained by storing compressed data at each level of the memory hierarchy. First, the effective storage space for each level of the hierarchy can be increased resulting in a reduction of cache misses and page faults. Second, the data transfer time can be reduced. Transferring data in compressed form can improve the effective bandwidth for each level of the memory hierarchy, resulting in a reduced miss penalty and a reduced data load time.

On the other side, the time taken to compress and decompress data incurs significant overhead and this negative effect is enlarged at the higher levels of the memory hierarchy. This overhead may offset the benefits of compression and worsen overall system performance. Thus, there are two fundamental problems to be solved: redesigning the highest level of the memory hierarchy to store the compressed data, and minimizing or hiding the compression/decompression time.

In this research, a selectively compressed memory system (SCMS) is proposed with its cache architecture and memory management policy. The SCMS employs several techniques to reduce decompression overhead and supports a fixed memory allocation method for storing various sizes of compressed data effectively. On-line compression and

* Corresponding author. Tel.: +82-2-2123-2718; fax: +82-2-365-2579.
E-mail addresses: jslee@kurene.yonsei.ac.kr (J.-S. Lee), sdkim@kurene.yonsei.ac.kr (S.-D. Kim), weems@cs.umass.edu (C. Weems).

0141-9331/02/$ - see front matter
PII: S0141-9331(01)00147-8

decompression are performed in hardware, which operates at the processor cycle rate. The performance of the proposed SCMS is evaluated via an analytic model devised in this work, which gives evaluation results that are realistic and reliable without resorting to complicated simulation. The results show that the performance of SCMS is largely affected by three major parameters, i.e. the compression efficiency, the percentage of references that are in the compressed data blocks, and the percentage of references found in the decompression buffer. The decompression buffer is critical in improving the performance of the SCMS. If the decompression buffer handles at least 70% of references to the compressed blocks, the SCMS significantly improves performance with respect to conventional memory systems (CMS). The improvement is, of course, the greatest when compression efficiency is high, and the probability of referencing the compressed data is also high.

Section 2 reviews the on-line data compression method and related work. In Section 3, the characteristics and organization of the proposed SCMS are described, including the cache architecture and management policy. Performance improvement (PI) is evaluated analytically in Section 4. Finally, Section 5 draws our conclusions.

2. Related work

In Ref. [3], a compression technique is used to increase the effective size of memory space and reduce page faults. The compression and decompression are performed in software by the operating system, and compressed pages are stored in a portion of memory space called the 'compression cache'. But, software compression techniques offer performance improvements on a limited set of application programs. Also it shows that performance improvements due to compression are dependent on the compression/decompression time, the compression ratio of data, and the data access pattern of programs.

In Refs. [5,7], the X-Match and X-RL data compression algorithms are presented and implemented by simple hardware, which shows high throughput. The X-RL algorithm as an expanded version of the X-match algorithm that appends run-length encoding, which is sensitive to consecutive zeros. X-RL shows good compression efficiency for memory data. An on-line hardware compression technique is applied to flash memory [8,9], where fixed size blocks must be erased before a new write operation is possible. This feature is desirable in designing a management method to solve the 'fat write' problem in compressed memory.

Ref. [10] presents the Wheeler compression/decompression algorithms and proposes a cache architecture for compressed data. This work applies compression to the higher levels of the memory hierarchy but it does not provide a performance improvement due to the negative effect of decompression overhead and the problem of cache management for fat writes. However, it suggests the possibility that as the processor–memory performance gap increases, the gains due to compression will exceed the losses due to decompression overhead and eventually result in an overall performance improvement.

Research on compressed memory has also been performed in industry. Recently, Abali et al. [1] suggested a memory sub-system called 'memory expansion technology' to increase the effective size of the installed main memory. It incorporates real-time main memory compression/decompression using a parallel processing architecture and a sophisticated memory management architecture to store variable-sized compressed data in the main memory. In particular, it has a large external cache between the processor bus and the compressed main memory, which reduces decompression latency by storing frequently accessed data in uncompressed form. The role of the cache is quite similar to that of the decompression buffer employed in the SCMS.

3. Organization and management of SCMS

In this section, the data compression method is described and the performance of the X-RL algorithm is reported in terms of the compression ratio for various benchmarks. The characteristics and organization of SCMS with its execution flow model are also presented.

3.1. Characteristics of SCMS

The performance of compression/decompression algorithms can be evaluated by their compression ratio and compression/decompression time. Compression ratio is defined as the ratio of compressed data length to the source data length, thus a lower compression ratio corresponds to a higher compression efficiency. As the size of the source data increases and the size of a unit symbol representing the source data decreases, the probability of similarity among symbols increases, which improves the compression ratio. However, compression/decompression times also increase in proportion to the number of symbols, and reducing the symbol size leads to an increase in symbol count due to its characteristics of sequential process.

LZ77 [12], BSTW [5], Wheelers [10], and X-Match/X-RL [8] algorithms have been proposed for on-line data compression and decompression. In our research, the X-RL algorithm, which is superior in both compression ratio and throughput, is chosen. The X-RL algorithm shows excellent performance especially on data containing symbols with frequent runs of consecutive zeros, because run-length encoding is integrated into the basic X-Match algorithm.

3.1.1. Selective compression technique

In the SCMS, compression is performed selectively in accordance with the compression ratio of each block. In