2002

Cache performance of video computation workloads

Stefan Petko

Dhireesha Kudithipudi

Eugene John

Cache Performance of Video Computation Workloads

Stefan Petko, Dhireesha Kudithipudi, and Eugene John
Department of Electrical and Computer Engineering
The University of Texas at San Antonio
San Antonio, TX 78249
{spetko@eng.utsa.edu, ejohn@utsa.edu}

ABSTRACT

Many video computation workloads execute on general-purpose computers including workstations, desktop systems and personal computers. The microprocessors that form the central processing unit of these systems typically expend a significant part of their real estate for on-chip caches and hence it is important to understand cache behavior of video computation workloads. Caches generally improve application performance by capturing temporal and spatial locality in instruction and data accesses. Modern processors run at speeds in the GHz range, and contain 2 or 3 levels of caches. The access time of main memory units in these systems are in the order of 100 cycles. Unless the caches can result in good hit ratios, the memory access times will be prohibitive. In this paper, we study the cache performance of video workloads. The workloads studied include JPEG, MPEG-2 and H263. The impact of cache sizes, block sizes and associativity on cache performance is discussed. It is observed that caches are used very well by these video computation workloads.

I. INTRODUCTION

Media processing is a very powerful and rapidly developing field. The increasing availability of low cost, low power, highly accurate video imagery has resulted in the rapid growth of applications using video technology. The effective use of video requires understanding of video processing, video analysis, video synthesis, video retrieval, video compression and other related computing techniques. Video and image processing are dependent on the computing factors such as encoding and decoding, processing, display and communication. Hence there is a need for high computational performance.

To study and understand the performance issues of image and video processing and computing them is a challenge of its own because of the large number of application domains handled by general purpose processor designers. Though there are numerous benchmarks prevailing in today's market, a clear understanding of the memory and cache performance of media applications is lacking. It is commonly perceived that media applications have streaming data and hence do not make good use of caches. This paper

This research was supported by Texas Higher Education Coordinating Board - ATP grant.

attempts to characterize the memory behavior of video and image processing workloads and study the effectiveness of caches for these workloads. The impact of varying cache specifications like cache size, block size, and associativity on memory performance is studied.

The video and image processing workloads generally access very large data sets and perform streaming data accesses. This typically decreases the data cache performance because it decreases temporal locality resulting in higher cache miss ratios. In contrast with data cache performance the workloads we examined have very low instruction cache miss rates since their code consists of tight loops that are repeatably used.

The rest of the paper is organized as follows. Section 2 describes the workloads used in this study and provides basic simulation statistics. Cache configurations and the simulation tools we used are described in section 3 and 4 respectively. Experimental results are presented and explained in section 5 and the paper is concluded in section 6.

II. WORKLOADS

Table 1 provides information about the video computation workloads that we chose for our experiments. Compression and decompression are important components of video workloads. Hence three image compression applications, JPEG, EPIC, BTPC and two video compression applications MPEG2 and H.263 are included. The implementations of JPEG, MPEG2 and EPIC come from the MediaBench suite [1]. The implementation of BTPC is taken from [8] and we used Telenor's version 1.7 implementation of H.263 [12].

JPEG: JPEG is the acronym for the Joint Photographic Experts Group - an ANSI/ISO/IEC standards organization. JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes. JPEG is lossy, meaning that the decompressed image is not quite the same as the original image. JPEG exploits limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. The four critical steps are extracting an 8x8 pixel block from the picture, calculating the DCT (Discrete Cosine Transform) for each element, the lossy phase where the quantizer rounds off the DCT with respect to the image quality, and compressing the coefficients using Huffman or arithmetic scheme. Cjpeg is used for encoding and djpeg is used for decoding images.

MPEG2: MPEG2 is the acronym for Motion Picture Expert Group. MPEG2 is easily customized in terms of I/O format, image outlet latency, memory optimization, audio decoder etc. The critical steps involved in decoding using MPEG2 are variable length encoding, zigzag scan, quantization, DCT, and motion compensation. Mpeg2 is the most popular standard used for video compressions and is used in digital TV transmission and DVD technology. Mpeg2enc is used for encoding and mpeg2dec is used for decoding video.

H.263: H.263 is designed for very low bit rate video imaging applications. H.263 uses block motion-compensated Discrete Cosine Transform (DCT) structure for encoding. It is based on 2-D DCT with simple motion estimation between frames. H.263 standard is extensively used in video telecommunication applications such as videoconferencing for example. An encoding specification called tmn is used for optimization in H.263. Video encoding is performed by partitioning each picture into macroblocks.

Application	Encode / Decode	Input Format	Total # of Instructions	I+D References	I-Refs %	D-Refs %
JPEG	cjpeg	ppm	40,485,402	47,821,434	85.32	14.67
	djpeg		31,222,812	37,527,113	84.24	15.76
EPIC	epic	pgm	51,498,275	59,078,793	87.29	12.71
	unepic		8,588,720	10,443,701	82.79	17.21
BTPC	cbtpc	ppm	63,978,277	85,699,278	76.37	23.63
	dbtpc		30,417,660	41,879,667	73.17	26.83
MPEG2	mpeg2enc	YUV	150,155,898	194,672,349	80.55	19.45
	mpeg2dec		23,112,916	29,561,252	78.86	21.14
H263	tmn	YUV	202,796,755	236,901,074	85.98	14.02
	tmndec		18,861,125	22,394,177	84.70	15.30

Table 1. Workloads used in this study

EPIC: EPIC is the acronym for Efficient Pyramid Image Coder. It is used for fast decoding. The compression algorithm is based on critically sampled dyadic sub band decomposition and a combined run-length/Huffman entropy coder [1]. It uses algorithms specialized for images, such as wavelet decomposition to perform image compression. Epic is used for compressing and unepic is used for decompressing images.

BTPC: BTPC is the acronym for Binary Tree Predictive Coding. It performs both lossy and lossless compressions and is effective for still images. It is superior to JPEG when performing lossy compressions. The encoder and decoder of BTPC share a common data structure for the pyramid, a lookup-table implementation of the predictor, a compactor data type for the Huffman coders. cbtpc is used for encoding and dbtpc for decoding images.

III. SIMULATION TOOLS

The aforementioned video workloads are executed on the UltraSPARC platform. The Shade tool suite [4] from Sun Microsystems and cachesim5, the cache simulator from Sun are used for our study. Shade is an instruction-set simulator and custom trace generator. Application programs are executed and traced under the control of a user-supplied trace analyzer. Cachesim5 is a Shade analyzer for cache simulation that we used to obtain cache miss statistics for our applications. Applications were compiled with GNU's gcc version 3.1 compiler, executed on the Sun UltraSPARC, and the instruction and data stream analyzed through the cache simulator. The total instructions counts, and percentage of memory references that are instruction references and percentage of memory references which are data references are listed in Table 1.

IV. CACHE CONFIGURATIONS

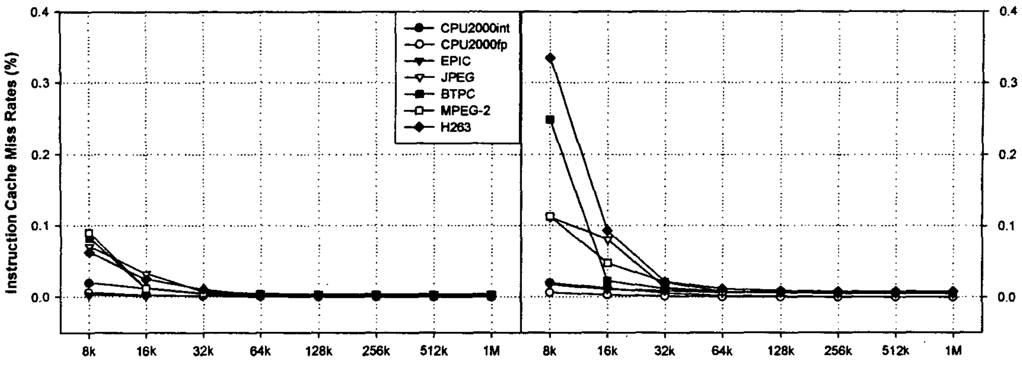
We perform experiments with cache sizes ranging from 8KB to 1MB. Block sizes in the 16 byte-128 byte range are used for studies on the impact of block sizes, however, unless otherwise noted, the block size is 64 bytes in most of the results presented. Caches can be direct-mapped or associative. Direct mapped, 2-way set-associative and 4-way set-associative caches are analyzed in this study.

The sizes and configurations we chose cover the actual cache sizes/configurations in state of the art microprocessors. Level 1 caches in state-of-the-art processors are in the 8KB to 64KB range and level 2 caches are in the 256KB-1MB range. Block sizes on most modern processors are in the 32 byte to 128-byte range. Most of the on chip caches are either direct mapped or 2-way or 4-way set associative.

V. EXPERIMENTAL RESULTS

Figures 1 and 2 show the instruction and data cache miss ratios for all 5 applications used in this study and compare them to the cache miss statistics of the SPEC CPU2000 benchmark suit described in [10]. These results were obtained from [11] and are divided into integer and floating point benchmarks. The SPEC CPU2000 benchmark suite is a collection of 12 integer and 14 floating point programs used to evaluate the performance of a computer's CPU, memory system, and compilers. For Figures 1 and 2, caches have 64-byte block size and are 2-way associative.

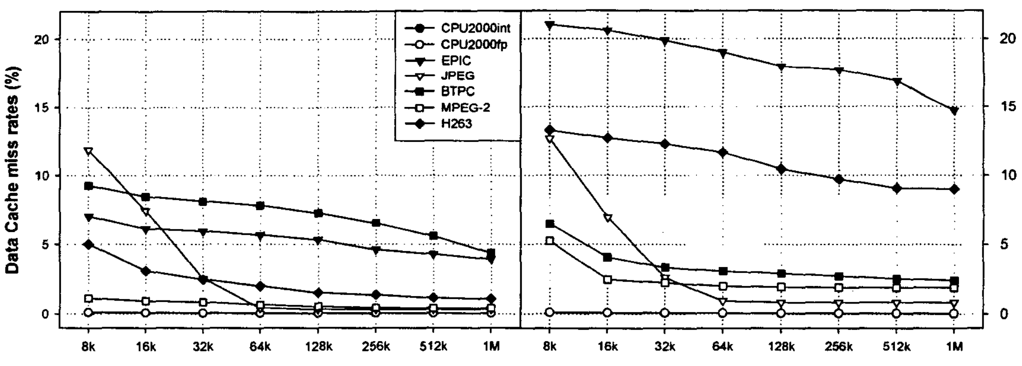
As expected, cache miss ratios decrease with increasing cache sizes. Instruction cache hit rates are well over 99% for caches as small as 64K. Instruction cache hit ratios are excellent for these applications primarily because the code in most instances consists of a few tight loops, which are repeatedly used. The footprint of most of these applications is small and hence the repeated use without evicting the code out of the cache results in extremely good instruction cache performance. Higher miss rates for the decode side of the graph are caused by lower instruction count during decoding for all applications. With cache sizes above 64K the miss rates of the benchmarks used it this study is comparable to miss rates of CPU2000 benchmarks. Part of the reason why CPU2000 miss rates are lower in both I-cache and D-cache results is that the instruction count of CPU2000 benchmarks averaged close to 300 billion as reported in [11]. This greatly reduces the effect of cold start misses.



(a) Encode (b) Decode

Figure 1: I-cache miss rates. Block Size: 64b Associativity: 2-way

Data cache hit ratios also improve with increasing cache sizes. The hit ratios improve to above 90%, but do not reach values like the 99% achieved for instruction caches. High D-cache miss rates for EPIC and BTPC encode and EPIC and H.263 decode primarily result from the data write misses for these applications. We observe that increasing cache size has almost linear affect on cache misses for most applications.



(a) Encode (b) Decode

Figure 2: D-cache miss rates. Block Size: 64b Associativity: 2-way

The impact of cache block size variation is displayed in figure 3. In Figures 3 and 4 cache miss rates of all 5 encoding applications we used is averaged. Increasing cache block size has a different effect on I-cache than the D-cache. From Figure 3 we observe that large block size results in lower I-cache miss rates for small cache sizes, however, for large cache sizes, the difference in block sizes does not make any impact. Block size variation effects on D-cache miss rates are noticeable at all cache sizes we examined. For small D-cache sizes, smaller block size produces low miss rates and increasing block size improves the hit rates for cache sizes greater than 32K. Much greater exploitation of temporal locality in instruction usage than data usage results in a difference in cache performance trends between I-cache and D-cache, when block size is increased.

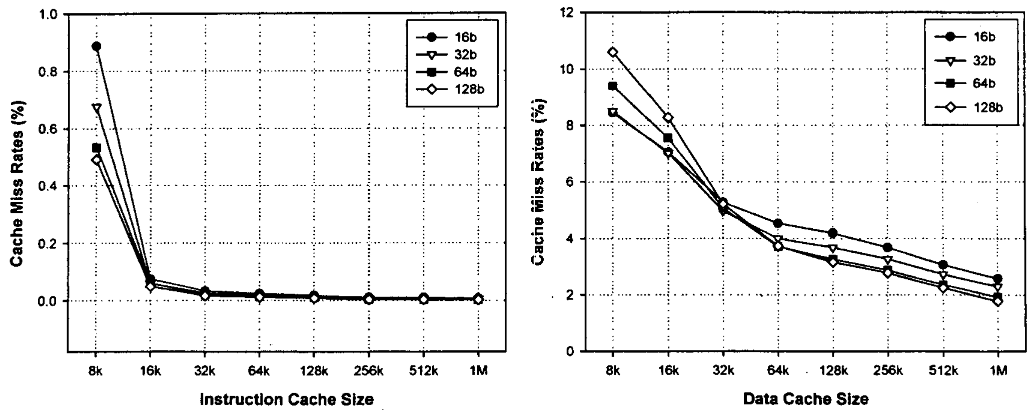


Figure 3: Impact of Block Size on Cache Miss Rates. Associativity: DM

Increased associativity also improves the hit ratio, however, the difference is noticeable only at small cache sizes as shown in Figure 4. Hence it can be concluded that increased associativity matters only for level 1 caches. Otherwise, cache size is the only primary indicator of performance.

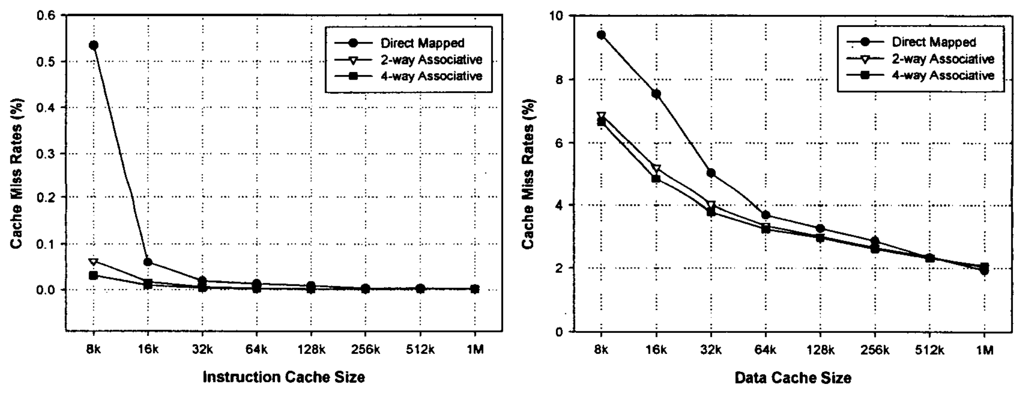


Figure 4: Impact of Associativity on Cache Miss Rates. Block Size: 64b

VI. CONCLUSION

It is commonly perceived that the cache performance of media applications is not good, however our work indicates that caches are used very well by these video computation workloads. Since many of these applications have streaming data, the temporal locality in data usage is perceived to be not good. However, we observe that the temporal locality in data accesses is also good because of the coefficients and constants used in the computation of transforms, filters, etc. Also, in many of these applications, matrix multiplication is very commonly used. In matrix multiplication, the same data elements are repeatedly used, resulting in temporal locality in addition to the spatial locality. Thus the applications are seen to benefit from spatial locality of streaming in data, and temporal locality of reused data coefficients. This study indicates that it is extremely important to have large caches in computation engines that cater to video computation workloads.

REFERENCES

[1] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A tool for evaluating and synthesizing multimedia and communication systems", *Proc. of the 30th Annual International Symposium on Microarchitecture*, Dec 1997, pp. 330-335.

[2] Byeong Kil Lee and L. K. John, "Implications of Programmable general Purpose Processors for Compression/Encryption Applications", *Proceedings of the ASAP conference*, July 2002.

[3] R. F. Cmelik and D. Keppel, 'Shade: A fast instruction set simulator for execution profiling", Sun Microsystems Inc., Technical Report SMLI TR 93-12, 1993.

[4] P.Ranganathan, S.Adve, N.P. Jouppi, "Performance of image and video processing with general-purpose processors and media ISA extensions", *Computer Architecture, 1999. Proc. of the 26th International Symposium on*, pp 124-135,1999.

[5] R. Cucchiara, A.Prati, M.Piccardi, "Data-type dependent cache prefetching for MPEG applications", *Performance, Computing, and Communications Conference, 2002. 21st IEEE International,* pp 115-122, 2002.

[6] Vijay S. Pai, P.Ranganathan, Sarita V. Adve, "The Impact of Instruction-Level Parallelism on Multiprocessor Performance and Simulation Methodology",*High-Performance Computer Architecture, Third International Symposiumon,*pp72-83,1997

[7] David A. Patterson, John L.Hennessy, "Computer Organization and Design: the hardware/software interface", 2nd edition, ISBN 1-55860-428-6,1998.

[8] John. A. Robinson, " Binary Tree predictive Coding". Available: http://www.engr.mun.ca/~john/btpc.html

[9] D. Talla, "Architectural techniques to accelerate multimedia applications on general-purpose processors," Ph.D. thesis, Dept. of Electrical and Computer Engineering, The University of Texas, Austin, Aug. 2001. Available: http://www.ece.utexas.edu/projects/ece/lca/ps/deepu_talla_dissertation.pdf

[10] John L Henning, "Measuring CPU Performance in the new millennium", 2000. Available: http://www.specbench.org/osg/cpu2000/papers/COMPUTER_200007.JLH.pdf

[11] Jason F. Cantin, Mark D. Hill, "Cache Performance for SPEC CPU2000 Benchmarks", 2000. Available at: http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data

[12] Roalt Aalmoes, "Roalt's H.263 Page". Available at: http://www.xs4all.nl/~roalt/h263.html