

An MPEG-4 Performance Study for non-SIMD, General Purpose Architectures

Sally A. McKee

Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853

Zhen Fang

School of Computing
University of Utah
Salt Lake City, UT 84112

Mateo Valero

Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya
E-08071 Barcelona, Spain

Abstract

MPEG-4 is an important international standard with wide applicability. This paper focuses on MPEG-4's main profile, video, whose approach allows more efficiency in coding and more flexibility in managing heterogeneous media objects than previous MPEG standards. This study presents evidence to support the assertion that for non-SIMD architectures and computational models, most memory-system optimizations will have little effect on MPEG-4 performance. This paper makes two contributions. First, it serves as an independent confirmation that for current, general-purpose architectures, MPEG-4 video is computation bound (just like most other media processing applications). Second, our findings should prove useful to other researchers and practitioners considering how to (or how not to) optimize MPEG-4 performance.

1 Introduction

ISO/IEC 14496 [9], generally known as *MPEG-4* [14, 10], is an important international standard whose immediate applications range from digital television and internet streaming video to mobile multimedia and games. MPEG-4 defines four main layers or *profiles*: control, still image, video and audio. We focus on the main profile, video, whose approach allows more efficiency in coding and more flexibility in managing heterogeneous media objects than its predecessors.

Two features distinguish MPEG-4 from MPEG-1 and MPEG-2: interactivity and streaming. Scene construction can be overlapped with image download. The decomposition of media data into objects provides two main advantages. First, it maximizes the efficiency of compression and encryption by introducing the potential to extract uncorrelated media content from the same source (and to apply the best media processing strat-

egy for different types of content). Second, it allows a single protocol to manage a broad range of heterogeneous media content. Uncorrelated objects are coded, encrypted, and transmitted separately. At the reception site, powerful transformations (including zooming, rotation, or translation of image objects) may be performed over each object to recompose the audiovisual scene.

This new media standard thus enables interesting new functionalities, but poses difficult challenges with respect to computational capabilities. The “conventional wisdom” holds that the large working sets of streaming multimedia applications do not make effective use of large caches [6, 18, 19], which leaves such applications hungry for bus bandwidth. ¹ Kuroda and Nishitani [15] show that the large data volumes for MPEG-2 motion compensation cause cache misses and main memory bandwidth to be severe performance problems. They hold that the memory/bus bottleneck is a prohibitive obstacle for MPEG on general-purpose architectures. Note that there is little difference between MPEG-2 and MPEG-4 in motion compensation. It follows that traditional cache- and bus-based memory hierarchies must be insufficient for this new application domain.

Many have found media processing to be computationally intensive on modern platforms [24, 12, 18, 3, 11], but most of this work focuses on MPEG-1 or MPEG-2 *kernels*. With its new, real-time streaming feature, MPEG-4 poses a potential nightmare for a traditional memory hierarchy with shared memory buses. Media ISA extensions (*e.g.*, Intel's® MMX and SSE, AMD's® 3DNOW!, Sun's VIS™, and Motorola's AltiVec™) can improve media-processing performance. ² SIMD-tuned compilers to exploit these exist, but getting the best performance often requires that programmers insert calls to assembly code libraries, use

¹The literature can be confusing on these points — authors who recognize the high data locality of multimedia kernels still assume that cache and bandwidth utilization will be problematic.

²Jennings and Conte provide a succinct overview of many of these [11].

special data structures, or hand-instrument source or assembly code. In fact, the “lagging compilers” lead Conte *et al.* [2] to assert that adding multimedia extensions to a general-purpose processor may not be an appropriate solution for multimedia workloads, while others maintain that these ISA extensions will make general-purpose processors more efficient and affordable than custom media accelerators [6]. Appropriate or not, most commercial multimedia applications now run on platforms that support these extensions, and thus have been tuned to take advantage of them. These applications can also be of interest in other contexts [26]. For instance, we study how MPEG-4’s real-time streaming feature affects memory performance on high-performance graphics machines. The original motivation was to assess the opportunity for improving performance via memory-system optimizations, but we find that the code behaves contrary to our expectations.

This paper is intended to help overcome certain misconceptions about MPEG-4 behavior with respect to main-memory utilization (particularly for high-performance platforms lacking SIMD ISA extensions). While perhaps not prevalent in the literature, we have often heard these misconceptions often in informal discussions on MPEG-4, and had held them ourselves prior to conducting this study.

We find the following assumptions to be false:

- MPEG-4 is a memory-streaming application.
- MPEG-4’s performance is limited by bus-bandwidth.
- MPEG-4’s performance is limited by latency.
- MPEG-4’s performance is adversely affected by larger image sizes.
- MPEG-4’s performance is adversely affected by a greater number of images or layers.

We target the Computer Architecture research community to help others avoid proposing architectural enhancements not needed for multimedia. We focus on a single profile, and we study a single implementation of MPEG-4. We do not experiment with MPEG-4 audio here, but our experience suggests it will present no problem to cache performance: MP3 audio applications [20], GSM long-term frequency vocoders [1], and similar codes are cache-friendly, since they also work at the frame level (one dimension, in this case), and since filtering and convolution operations (common in audio) have high temporal and spatial data locality.

This work is not intended to be exhaustive, but we believe our preliminary evidence sufficiently compelling to support the assertion that for non-SIMD architectures and computational models, most memory-system optimizations will fail to deliver significant improvement on

MPEG-4 performance. The contributions of this paper are thus twofold. First, it serves as independent confirmation that for current, general-purpose architectures, MPEG-4 video is computation bound, just like most other media-processing applications. MPEG-4’s streaming functionality does not affect this property of multimedia codes. Second, our findings should prove useful to other researchers and practitioners who may consider how to (or how not to) optimize MPEG-4 performance.

2 Background

This section first provides a brief tutorial on the video profile of the MPEG-4 standard, and then surveys the conclusions drawn from related MPEG performance characterization and architectural studies.

2.1 Overview

We provide an introduction to the MPEG-4 visual profile, but the interested reader may find much more detailed information elsewhere [10]. For reference, Jennings and Conte [11] provide a concise overview of MPEG-2 operation. The MPEG-4 standard’s object-based approach hinges on the central concept of the *visual object*, or *VO*. A visual object corresponds to a particular 2-D object in the scene, and is characterized by temporal and spatial information in the form of shape, motion, and texture. VOs are sampled in time, and each can be encoded in scalable (multi-layer) or non-scalable (single layer) form.

Each time sample of a video object constitutes a *video object plane*, or *VOP*, containing motion parameters, shape information, and texture data. VOPs are encoded using 16×16 or 8×8 macroblocks, where a macroblock contains a section of the luminance component and the spatially subsampled chrominance components. Texture is coded separately by a discrete cosine transform (DCT) scheme. Arbitrary shapes are coded using a context-based arithmetic encoding scheme and are compressed via a bitmap-based method. Motion estimation and compensation are used in video compression to exploit temporal redundancies between frames. Whereas MPEG-1 and MPEG-2’s motion estimation uses block-based techniques, MPEG-4’s motion estimation has been adapted to VOPs.

Figure 1 illustrates how motion estimation is used in coding visual object planes, with arrows indicating interframe dependences. An *Intra VOP*, or *I-VOP*, is encoded independently and contains a complete image that is compressed for spatial redundancy only. A forward *Predicted VOP*, or *P-VOP*, is built from the nearest previously coded VOP. A *Bidirectional VOP*, or *B-VOP*, is interpolated based on I-VOPs and P-VOPs. Pulling

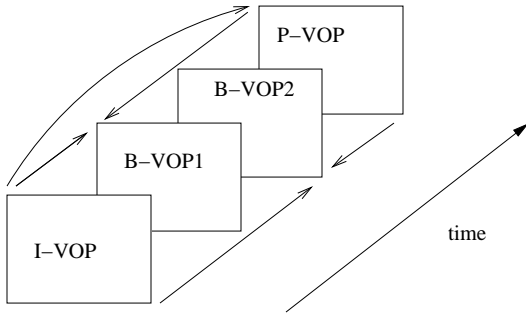


Figure 1. Three Modes of VOP Coding in MPEG-4

data from both past and future frames offers excellent opportunities for compression, but introduces data dependences. The decoder reads a stream of bits looking for the unique bit patterns called *startcodes* that mark the divisions between different sections of data in the hierarchical structure described above. The decoder must follow the operation-order dependences decided by the encoder. The VOPs are thus processed in the non-temporal order (I-VOP, P-VOP, B-VOP1, B-VOP2, ...). In other words, when the display order is I, B1, B2, P, the encoding and decoding orders are both I, P, B1, B2. This out-of-order decoding (with respect to temporality) increases the performance and storage requirements for real-time playback.

2.2 Related Work

Since we approach MPEG-4 from a memory-performance perspective, we briefly survey related work in the same vein. Space limitations prevent us from providing a comprehensive survey of background work — rather, we strive to convey the flavor of prior work examining media application performance and contributing to our former misconceptions. We wish to stress that the claims and findings of those cited here are not necessarily incorrect in other contexts (*e.g.*, with older workloads and processor models, or with SIMD ISA extensions), but instead that it may be inappropriate to infer that these conclusions also apply to MPEG-4 on general-purpose platforms.

Many researchers, the authors included, have claimed that streaming access patterns and large working sets in multimedia applications make ineffective use of caches [6, 18, 19, 25].

Prefetching is one method commonly advocated to attack the DRAM latency problem for media applications [16, 6, 17, 26]. For example, in their IA32 SIMD Streaming Extensions, Raman [17] *et al.* maintain that software prefetching will be important in supporting future streaming media applications. The rationale behind this is that these applications, especially MPEG-

4, are memory-latency bound. Zucker *et al.* [26] make this assumption in their trace-based study of hardware and software prefetching for MPEG benchmarks on the PA-RISC architecture. We assume their results are for traces generated by MPEG-2 applications, but this is not explicitly stated. This need for prefetching contradicts our findings for MPEG-4. In other contexts, *e.g.*, systems that exploit SIMD multimedia instruction sets, prefetching can be profitable [19, 21]. Extrapolating to assume prefetching works well on general-purpose computers lacking such ISA extensions or on applications not linked with SIMD libraries is potentially erroneous, as we show below.

Bus bandwidth is almost unanimously considered a bottleneck for multimedia [6, 13]. The general claim is that the demand for continuous media data in applications like MPEG-4 makes their performance severely limited by bus bandwidth (internally and externally). We find that for our experiments on readily available systems, only a small fraction of the available bus bandwidth is needed. Ironically, this is due to *good* cache performance. Zucker *et al.* [26] assume bus bandwidth will not be a problem in future processors, but they do so based on an economic argument, not based on an analysis of cache performance.

Another closely related, popular misconception is that higher resolution in video will significantly increase memory usage, thus exacerbating the memory-wall problem. For example, Ranganathan [18] *et al.* conclude that a 1024×1024 image requires a factor of $12\times$ increase in L2 cache size over that needed for a 352×240 image. We find that even at extremely high resolution images (2048×1024 pixels, for example), the cache performance of MPEG-4 video remains at approximately the same level. Our findings agree with the later MPEG-2 working set studies of Hughes *et al.* [8], in which they scale image input sizes as part of their research on the variability of multimedia application performance. This independence from image size is counterintuitive.

3 Experimental Results

In this section, we first describe our experimental setup. Then we address each of the misconceptions about performance that we initially held, and explain how our experience refutes them.

3.1 Experimental Methodology

We experiment on three machines: an SGI O2 (MIPS R12000 with 1MB L2 cache), an SGI Onyx VTX (MIPS R10000 with 2MB L2 cache) and an SGI Onyx2 InfiniteReality (MIPS R12000 with 8MB L2 cache). Ta-

Component	Features
L1 data cache	32KB, 2-way set associative, 32 bytes/line, LRU, WB
L2 unified cache	2-way pseudo set associative, 128 bytes/line, LRU, WB (size varies for each machine)
system bus	64 bits, 133 MHz, split transaction
main memory	4-way interleaved SDRAM
operating system	680MB/s sustained, 800MB/s peak IRIX64 v6.5

Table 1. Common Platform Highlights

ble 1 highlights the relevant, common features of these three systems. The Irix kernel implements 32 virtual performance counters via multiplexing two actual hardware counters. We use the SGI SpeedShop [23] performance analysis package and the Irix Perfex [22] profiling tool library to access these counters. We use the publicly available MPEG-4 visual encoder and decoder from the ISO reference software developed by the European Union ACTS project MoMuSys (Mobile Multimedia Systems) [7]. We compile with the MIPS `cc` compiler at optimization level `-O3`.

Our experiments manipulate a 30-frame video at two resolutions: the 720×576 used for PAL [5], and a 1024×768 size that exceeds NTSC but is less than HDTV [5]. Pixel depth is eight bits. The frame rate is 30 Hz, as in HDTV (note that PAL uses a 25 Hz rate), and the target bitrate is 38400.

Table 2 and Table 3 summarize our MPEG-4 visual encoding and decoding experiments, respectively. The numbers for instruction cache and TLB misses are negligible, and are omitted. *Cache line reuse* is the mean number of times a cache line is used after being loaded and before being evicted. For example, *L1C line reuse* is the graduated loads plus graduated stores, minus L1 data cache misses, all divided by L1 data cache misses. Likewise, *L2C line reuse* is L1 data cache misses minus L2 data misses, all divided by L2 data misses. *DRAM time* refers to the cycles during which the processor is stalled due to secondary data cache misses; this is the latency that out-of-order execution hardware and compilation techniques fail to hide. *L2-DRAM b/w* is the amount of data moved between the secondary cache and main memory, divided by the total program execution time. The amount of data moved is calculated as the sum of the L2 cache misses multiplied by the L2 cache line size, plus the number of bytes written back from L2. *L1-L2 b/w* is similar. *Prefetch L1C miss* refers to the proportion of prefetch instructions that do not become nops. A high prefetch miss rate (near one) is desirable, since prefetch hits waste instruction bandwidth and decoding resources. The MIPS R10000 cannot track the number of prefetches that hit in L1 cache; this statistic is only available on our MIPS R12000-based machines.

3.2 Fallacies and Paradoxes

We now examine in turn each of the popular assumptions described in Section 1.

Fallacy: MPEG-4 Exhibits Streaming References.

Primary cache performance is nearly optimal across all hardware configurations and input sizes. Even at 1024×768 pixels/frame, the L1 data cache hit rate is up to 99.91%. Of the many data accesses that constitute the load streams and store streams dispatched by the processor, only 0.1% and 0.4% go beyond L1 cache for encoding and decoding, respectively. These high hit ratios make L1 data misses account for less than 0.50% and 1.76% of execution time in encoding and decoding, respectively. On average, each L1 cache line is reused about 1000 times before eviction in encoding, and more than 200 times in decoding. The intuition that streaming MPEG-4 is a poor match for small caches is therefore false in this context: the data references in “streaming MPEG-4” do not really stream.

This phenomenon occurs because, in spite of the streaming nature of the *kernels* used, their composition into multimedia programs generates locality in two ways: (a) streams have high degrees of data overlap, and (b) different stages of the application’s “pipeline” process the same data resident in L1 cache. Furthermore, the MPEG-4 protocol itself dictates that data be organized in chunks (*e.g.*, 16×16 elements in motion estimation or 8×8 in discrete cosine transform).

Consider the encoder’s *motion estimation* (responsible for the majority of the program execution time). Motion estimation detects movement of objects along different video frames, searching for an image block best matching a reference block. The “resemblance” criterion is the minimum *sum of absolute differences*(SAD) between pixels of the two blocks. From the computation kernel’s perspective, processing data streams across different blocks exhibits little data locality. MPEG-4 performs this search sequentially over restricted windows inside the image, with an offset between searches of just one pixel. The overlap among streams for searching an image subset yields high locality both in the *x* axis, due to the data layout in memory, and in the *y* axis, due to the restricted size of the window. Simply put, the protocol-dictated blocking structure naturally creates locality.

Fallacy: MPEG-4 is Bound by DRAM Latency.

Memory requirements for MPEG-4 video processing are indeed large. This fact gives rise to the common assumption that memory latency is critical to MPEG-4 performance, especially since much MPEG-4 data is time-sensitive. In our experiments, over 99.5% of the data references hit the primary cache, and even a small 1MB secondary cache catches more than 60% of the rest. Very few references reach main memory. Out-of-order issue

metrics	720x576 pixels			1024x768 pixels		
	CPU and L2C size			CPU and L2C size		
	R12K 1MB	R10K 2MB	R12K 8MB	R12K 1MB	R10K 2MB	R12K 8MB
L1C miss rate	0.08%	0.08%	0.08%	0.11%	0.10%	0.09%
L1C miss time	0.34%	0.40%	0.29%	0.45%	0.50%	0.34%
L1C line reuse	1254.3	1287.9	1310.8	946.4	1020.3	1118.8
L2C miss rate	32.62%	15.70%	7.28%	41.23%	14.02%	9.87%
L2C line reuse	2.1	5.4	12.7	1.4	6.1	9.1
DRAM time	2.4%	1.3%	0.2%	4.0%	1.5%	0.4%
L1-L2 b/w (MB/s)	4.5	4.2	4.0	5.7	5.0	5.8
L2-DRAM b/w (MB/s)	4.9	2.7	1.9	7.8	3.1	2.7
prefetch L1C miss	41.4%	n/a	36.0%	41.5%	n/a	39.6%

Table 2. Video Encoding: One Visual Object, One Layer

metrics	720x576 pixels			1024x768 pixels		
	CPU and L2C size			CPU and L2C size		
	R12K 1MB	R10K 2MB	R12K 8MB	R12K 1MB	R10K 2MB	R12K 8MB
L1C miss rate	0.37%	0.38%	0.35%	0.40%	0.41%	0.37
L1C miss time	1.36%	1.61%	1.16%	1.44%	1.76%	1.25%
L1C line reuse	268.7	264.1	288.1	251.7	241.7	268.5
L2C miss rate	39.27%	19.31%	10.72%	36.48%	19.10%	13.91%
L2C line reuse	1.5	4.2	8.3	1.7	4.2	6.2
DRAM time	11.6%	6.6%	1.5%	11.3%	7.1%	2.0%
L1-L2 b/w (MB/s)	18.9	18.3	22.4	20.3	20.3	23.2
L2-DRAM b/w (MB/s)	24.3	14.9	9.8	24.0	15.9	13.3
prefetch L1C miss	36.4%	n/a	45.2%	41.6%	n/a	36.1%

Table 3. Video Decoding: One Visual Object, One Layer

and the MIPS optimizing compiler hide another portion of the latency of these main-memory references. For encoding on a system with a large L2 cache, the processor stalls as low as 0.2% of the time for medium-sized 720×576 frames. Even for a small L2C and large 1024×768 frames, the processor stalls only about 4.0% of the time. Decoding spends slightly more time on processor stalls while waiting for DRAM accesses. In the worst case, we observe a processor stall time of no more than 12%. In spite of initial claims that multimedia codes do not use caches well, Ranganathan *et al.* [18, 19] observe similarly small percentages of memory stall time (less than 10%) for their studies of MPEG-2 and other multimedia codecs.

With compiler-generated software prefetching, the number of executed prefetches is around 1/7000 the number of graduated loads in encoding and 1/1000 in decoding (not shown in the tables). Even for this conservative use of prefetching, over half of the prefetches hit the primary cache, and thus constitute a waste of system resources. Prefetching is therefore unlikely to improve MPEG-4 performance on the systems we study.

Fallacy: MPEG-4 is Hungry for Bus Bandwidth. In terms of sustained bandwidth, traffic is less than 2% between the L1 and L2 caches, and less than 4% between L2 cache and main memory. Again, this comes from high reuse and high primary cache hit rates. Given the high L1C hit rate, bandwidth between the ALUs and the primary cache might be a limiting factor, but the R10000 and R12000 counters prevent exploration of this

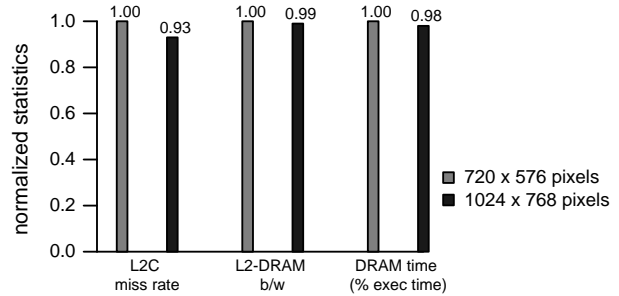


Figure 2. Memory Statistics for Growing Image Size (Decoding, 1MB L2C)

conjecture. Simulation studies [4] indicate that a non-SIMD/non-vector MPEG-4 code with a dual-ported L1C is not limited by bandwidth constraints, while a vector version *is*, even with a four-ported cache.

Fallacy: MPEG-4 Memory Performance Degrades with Growing Image Size. At first glance, this extrapolated assumption from image processing experiences seems plausible, even likely. Nonetheless, our results suggest otherwise. Despite the fact that memory requirements grow about linearly with respect to image size, performance remains almost the same when the image size is almost doubled (from 720×576 to 1024×768 , a factor of 1.9). Even with extremely large frames (2048×1024 pixels) we see equally good memory performance (data not shown here). The blocking nature of the algorithms makes the image size largely

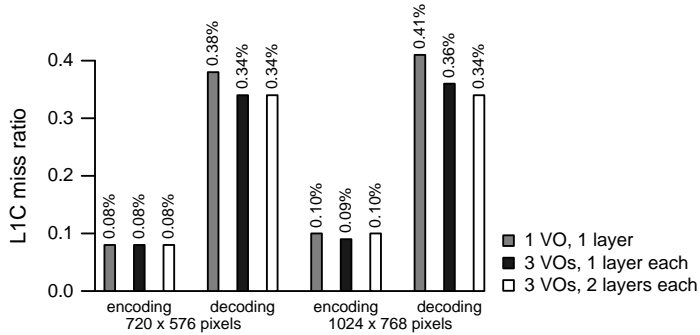


Figure 3. L1C Miss Rates for Varying Numbers of Objects and Layers

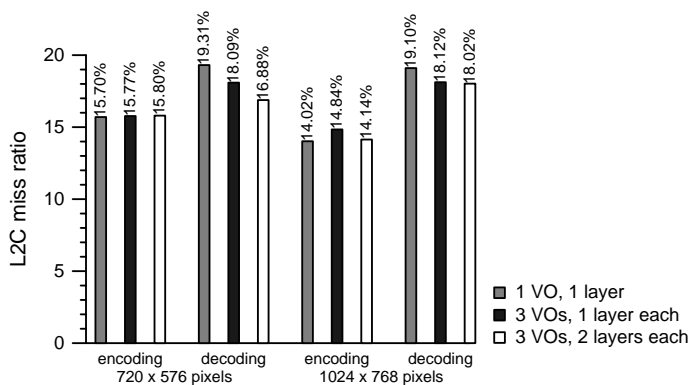


Figure 4. L2C Miss Rates for Varying Numbers of Objects and Layers

irrelevant. Interestingly, in several cases memory performance actually *improves* as the image size increases. For example, for decoding on a system with a 1MB L2, as the image size grows, the L2 miss rate, memory bandwidth consumption and DRAM stall time all decrease, as shown in Figure 2. Counterintuitively, cache performance of MPEG-4 video proves to be independent of frame size.

Fallacy: MPEG-4 Memory Performance Degrades as the Number of Visual Objects and Layers Grows. Thus far, we have concentrated on memory statistics for a single visual object with one layer. Increasing the number of visual objects (VOs) and visual object layers (VOLs) increases memory requirements accordingly. For example, on a workload of (1 VO, 1 VOL, 1024×768), encoding uses about 120 MB of stable, resident memory. At (3 VOs, 2 VOLs, 1024×768) encoding requires 400 MB. Intuition might suggest that cache performance would degrade as multiple VOs and VOLs compete for system resources. Surprisingly, we find this not to be the case. Table 4 and Table 5 contain

statistics for processing three visual objects. Each of the three objects is encoded and decoded for the same configuration as in the single object experiments, with the single-object input becoming a subset of the multiple-object input. Table 6 and Table 7 present the numbers for three visual objects with two visual object layers each. Table 4, Table 5, Table 6, and Table 7 with Table 2 and Table 3 show that cache performance does not change noticeably as the number of VOs and VOLs increases. Figure 3 and Figure 4 visually depict part of this comparison, showing L1C and L2C data miss rates on an R10K machine with an L2 cache of 2MB (note the different orders of magnitude in the y axis scales). More interesting is the change in performance for the decoding process as the number of VOs increases. Cache performance actually *improves* as we go from single object/single layer (Table 3), to multiple objects/single layer (Table 5), and up to multiple objects/multiple layers (Table 7). Consider the 1024×768 case. The L1 cache miss rates drop from 0.41% to 0.36% and 0.34%. L2 cache miss rates drop from 19.10% to 18.12% and 18.02%. And as a result, DRAM stall time drops from 7.1% to 5.9% and 5.6%. A similar trend occurs for the 720×576 case. This paradoxical behavior of “improving under pressure” reinforces our conclusions about MPEG-4 performance on the platforms we study.

3.3 Burstiness

To detect the inherent burstiness of MPEG-4 memory traffic, we instrument two of the most important functions — *VopCode()* in the encoder and *DecodeVopCombMotionShapeTexture()* in the decoder — by wrapping them in performance counter operations. Specifically, we want to determine if these two functions exhibit burstier memory behavior compared to the rest of MPEG-4 video. *VopCode()* performs shape, texture and motion coding of the input visual object plane, generating an encoded VOP. Motion estimation is the most time-consuming function in the VOP encoding process. *DecodeVopCombMotionShapeTexture()* is the reverse of *VopCode()*, decoding a VOP from the input bitstream. 2D DCT is the primary technique used in the decoding process. In the literature, motion estimation and DCT are the most frequently used examples for MPEG-4 memory optimizations. We do not apply instrumentation to the macroblock level of motion estimation or DCT, since at such fine granularity the inserted system calls would affect the execution noticeably and the accumulated error would have been unacceptable.

In Table 8, the functions *VopCode()* and *DecodeVopCombMotionShapeTexture()* are renamed *VopEncode* and *VopDecode*, respectively. Data are collected on a (R12K, 8MB L2C) machine.

metrics	720x576 pixels			1024x768 pixels		
	CPU and L2C size			CPU and L2C size		
	R12K 1MB	R10K 2MB	R12K 8MB	R12K 1MB	R10K 2MB	R12K 8MB
L1C miss rate	0.09%	0.08%	0.08%	0.10%	0.09%	0.09%
L1C miss time	0.35%	0.40%	0.27%	0.41%	0.48%	0.32%
L1C line reuse	1172.9	1235.6	1276.2	999.3	1060.6	1091.1
L2C miss rate	32.24%	15.77%	10.04%	33.02%	14.84%	10.97%
L2C line reuse	2.1	5.3	9.0	2.0	5.7	8.1
DRAM time	2.4%	1.4%	0.3%	2.9%	1.5%	0.4%
L1-L2 b/w (MB/s)	4.5	4.3	4.7	5.2	4.9	5.4
L2-DRAM b/w (MB/s)	4.9	2.8	2.0	5.8	3.1	2.6
prefetch L1C miss	39.6%	n/a	41.7%	42.5%	n/a	42.5%

Table 4. Video Encoding: Three Visual Objects, One Layer Each

metrics	720x576 pixels			1024x768 pixels		
	CPU and L2C size			CPU and L2C size		
	R12K 1MB	R10K 2MB	R12K 8MB	R12K 1MB	R10K 2MB	R12K 8MB
L1C miss rate	0.31%	0.34%	0.28%	0.33%	0.36%	0.30
L1C miss time	1.20%	1.46%	0.96%	1.27%	1.52%	1.06%
L1C line reuse	318.6	291.5	358.6	299.3	280.3	327.9
L2C miss rate	36.56%	18.09%	12.41%	35.22%	18.12%	14.92%
L2C line reuse	1.7	4.5	7.1	1.8	4.5	5.7
DRAM time	9.5%	5.6%	1.4%	9.7%	5.9%	1.9%
L1-L2 b/w (MB/s)	16.8	16.7	17.8	17.9	17.3	19.7
L2-DRAM b/w (MB/s)	20.2	12.3	9.5	20.6	13.0	12.0
prefetch L1C miss	44.4%	n/a	40.3%	41.2%	n/a	41.5%

Table 5. Video Decoding: Three Visual Objects, One Layer Each

metrics	720x576 pixels			1024x768 pixels		
	CPU and L2C size			CPU and L2C size		
	R12K 1MB	R10K 2MB	R12K 8MB	R12K 1MB	R10K 2MB	R12K 8MB
L1C miss rate	0.08%	0.08%	0.08%	0.11%	0.10%	0.10%
L1C miss time	0.34%	0.41%	0.29%	0.45%	0.51%	0.35%
L1C line reuse	1179.1	1202.7	1249.4	910.5	966.9	1028.3
L2C miss rate	32.27%	15.80%	9.97%	40.83%	14.14%	10.15%
L2C line reuse	2.1	5.3	9.0	1.4	6.1	8.9
DRAM time	2.4%	1.4%	0.3%	4.0%	1.5%	0.4%
L1-L2 b/w (MB/s)	4.5	4.3	4.9	5.7	5.2	5.9
L2-DRAM b/w (MB/s)	4.9	2.8	2.1	7.8	3.2	2.6
prefetch L1C miss	41.7%	n/a	43.8%	43.6%	n/a	40.6%

Table 6. Video Encoding: Three Visual Objects, Two Layers Each

metrics	720x576 pixels			1024x768 pixels		
	CPU and L2C size			CPU and L2C size		
	R12K 1MB	R10K 2MB	R12K 8MB	R12K 1MB	R10K 2MB	R12K 8MB
L1C miss rate	0.31%	0.34%	0.27%	0.33%	0.34%	0.29%
L1C miss time	1.20%	1.48%	0.95%	1.21%	1.46%	1.02%
L1C line reuse	317.3	289.5	367.7	304.8	289.3	338.1
L2C miss rate	34.83%	16.88%	12.67%	34.42%	18.02%	15.04%
L2C line reuse	1.9	4.9	6.9	1.9	4.5	5.7
DRAM time	9.1%	5.3%	1.4%	9.0%	5.6%	1.8%
L1-L2 b/w (MB/s)	16.9	17.1	18.2	17.1	16.6	19.2
L2-DRAM b/w (MB/s)	19.5	12.4	9.1	19.3	12.5	11.8
prefetch L1C miss	39.4%	n/a	41.1%	40.4%	n/a	38.7%

Table 7. Video Decoding: Three Visual Objects, Two Layers Each

metrics	VopEncode		VopDecode	
	3 visual objects 1 layer each 720x576 pixels	3 visual objects 2 layers each 1024x768 pixels	3 visual objects 1 layer each 720x576 pixels	3 visual objects 2 layers each 1024x768 pixels
L1C miss rate	0.05%(0.08%)	0.05%(0.10%)	0.72%(0.28%)	0.73%(0.29%)
L2C miss rate	3.09%(10.04%)	4.04%(10.15%)	9.80%(12.41%)	11.33%(15.04%)
L1-L2 b/w (MB/s)	5.7(4.7)	5.6(5.9)	38.3(17.8)	38.1(19.2)
L2-DRAM b/w (MB/s)	0.9(2.0)	1.3(2.6)	5.9(9.5)	6.8(11.8)

Table 8. Encoding and Decoding a VOP

For convenient comparison, we include in brackets the whole-program statistics taken from previous tables. Inspection reveals that the memory performance of these two functions is consistent with overall trends. The L2C miss rate and L2-DRAM traffic are both smaller than in the whole program. VopEncode sees better memory performance than overall encoding for all metrics shown in Table 8. For VopDecode, L1 cache references miss about twice as often as the whole-program average, increasing L1C-L2C traffic. Even with this degraded L1C performance, the L1 cache still captures over 99.2% of all accesses in the load streams and store streams dispatched by the processor. Improved L2C hit rate further filters the memory references, reducing traffic to DRAM. This test of burstiness for key encoding/decoding phases tells us that, although each of the kernels (*e.g.* SAD and DCT) references data in a streaming manner, at the VOP level the comprehensive effect of multiple streams is a working set that fits well into cache.

4 Conclusions and Future Work

We find that for the experiments we conduct, the MPEG-4 video profile has good memory performance: high primary cache hit ratios, high cache-line reuse, low main-memory stall times, and low bus-bandwidth requirements. Although we experiment with general-purpose processors lacking MMX-like SIMD extensions, our experience has shown that even in the presence of these ISA extensions, the performance bottleneck is still the fetch/issue rate [3]. Only in the presence of longer vector SIMD instructions does L1 bandwidth surpass fetch rate as a limiting performance factor. Given our experiences, we caution against generalizing results from older workloads, different processor/ISA models, and different domains (*e.g.*, image processing) to modern multimedia workloads on general-purpose platforms. On the other hand, care should be taken when generalizing observations made in this paper to other platforms or MPEG-4 implementations. Fortunately, the increasing availability and accessibility of hardware performance counters in such platforms makes it easier to examine application behavior to understand where the performance bottlenecks are (or are not).

This work represents our preliminary findings. While performance numbers from commercial hardware using a commercial compiler have the advantage of being more realistic than those generated via software simulation, the parameters of our experiments are necessarily rigid. In order to investigate how MPEG-4 behaves with different architectural configurations, we are extending our experiments to a spectrum of representative platforms (including IA32, IA64, and Power4). Our intuition is that the memory performance of the MPEG-4 visual profile is unlikely to change qualitatively on any mainstream workstation with a conventional cache hierarchy. These studies will also incorporate the effect of SIMD ISA extensions, evaluating the appropriateness of adding such extensions to general-purpose processors. Previous work [4] finds that vector versions of MPEG-4 become L1 cache bandwidth-limited. Investigating the behavior of a non-vectorized MPEG-4 that exploits multimedia ISA extensions on general-purpose machines is a next step, and the following one is gathering data for other MPEG-4 profiles. Finally, we will conduct simulation studies to determine at what ratio of processor-to-memory speed and at what bandwidths among various levels of the memory hierarchy the performance of MPEG-4 does finally become memory limited.

5. Acknowledgments

We thank Jesus Corbal for comments on drafts of this paper.

References

- [1] S. Bonnett. MP3 technical information. <http://www.iocon.com/das/technical.shtml>, 2001.
- [2] T. Conte, P. Dubey, M. Jennings, R. Lee, A. Peleg, S. Rathnam, M. Schlansker, P. Song, and A. Wolfe. Challenges to combining general-purpose and multimedia processors. *IEEE Computer*, 30(12):33–37, Dec. 1997.
- [3] J. Corbal, R. Espasa, and M. Valero. DLP + TLP processors for the next generation of media workloads. In *Proceedings of the Seventh Annual Symposium on High Performance Computer Architecture*, pages 219–228, Jan. 2001.

- [4] J. Corbal, R. Espasa, and M. Valero. Three-dimensional memory vectorization for high bandwidth media memory systems. In *Proceedings of IEEE/ACM 35th International Symposium on Microarchitecture*, pages 149–160, Nov. 2002.
- [5] CVC Productions. World TV standards. http://avconvert.com/video/worldtelevision_standards.html, 2001.
- [6] K. Diefendorff and P. Dubey. How multimedia workloads will change processor design. *IEEE Computer*, 30(9):43–45, Sept. 1997.
- [7] European Union ACTS Programme. Mobile Multimedia Systems Project (MoMuSys). <http://www.infowin.org/ACTS/RUS/PROJECTS/ac098.htm>.
- [8] C. Hughes, P. Kaul, S. Adve, R. Jain, C. Park, and J. Srinivasan. Variability in the execution of multimedia applications and implications for architecture. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 254–265, June 2001.
- [9] ISO/IEC 14496-2. Coding of audio-visual objects:visual, final draft international standard, Oct. 1998.
- [10] ISO/IEC Moving Picture Experts Group. The MPEG Webpage. <http://mpeg.telecomitalia.com/>, 2002.
- [11] M. Jennings and T. Conte. Subword extensions for video processing on mobile systems. *IEEE Concurrency*, 6(3):13–16, July-September 1998.
- [12] B. Khailany, W. Dally, S. Rixner, U. Kapasi, P. Mattson, J. Namkoong, J. Owens, B. Towles, and A. C. hang. Imagine: Media processing with streams. *IEEE Micro*, 21(2):35–46, March/April 2001.
- [13] J. Kneip, B. Schmale, and H. Moller. Applying and implementing the MPEG-4 multimedia standard. *IEEE Micro*, 19(6):64–74, Nov. 1999.
- [14] R. Koenen. Mpeg-4: Multimedia for our time. *IEEE Spectrum*, 36(2):26–34, Feb. 1999.
- [15] I. Kuroda and T. Nishitani. Multimedia processors. *Proceedings of the IEEE*, 86(6):1203–1221, June 1998.
- [16] A. Prati. Exploring multimedia applications locality to improve cache performance. In *ACM Multimedia*, pages 509–510, 2000.
- [17] S. Raman, V. Pentkovski, and J. Keshava. Implementing streaming SIMD extensions on the Pentium III processor. *IEEE Computer*, 33(7):47–57, July 2000.
- [18] P. Ranganathan, S. Adve, and N. Jouppi. Performance of image and video processing with general-purpose processors and media isa extensions. In *Proceedings of the 26th Annual International Symposium on Computer Architecture*, pages 124–135, June 1999.
- [19] P. Ranganathan, S. Adve, and N. Jouppi. Reconfigurable caches and their application to media processing. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 214–224, June 2000.
- [20] J. Scourias. GSM. <http://www.shoshin.uwaterloo.ca/jscouria/GSM/index.html>, 1999.
- [21] J. Sebot and N. Drach. Memory bandwidth: The true bottleneck of simd multimedia performance on a superscalar processor. In *Proceedings of the 2001 European Conference on Parallel Computing*, pages 439–447, Aug. 2001.
- [22] Silicon Graphics, Inc. Origin 2000 and Onyx2 Performance Tuning and Optimization Guide (document number: 007-3430-003). <http://techpubs.sgi.com>, 2001.
- [23] Silicon Graphics, Inc. SpeedShop User's Guide (document number: 007-3311-007). <http://techpubs.sgi.com:80/library/tp1/cgi-bin/init.cgi>, 2001.
- [24] N. Slingerland and A. Smith. Cache performance for multimedia applications. In *Proceedings of the 2001 International Conference on Supercomputing*, pages 204–217, June 2001.
- [25] L. Zhang, Z. Fang, M. Parker, B. Mathew, L. Schaelicke, J. Carter, W. Hsieh, and S. McKee. The impulse memory controller. *IEEE Transactions on Computers*, 50(11):1117–1132, Nov. 2001.
- [26] D. Zucker, R. Lee, and M. Flynn. Hardware and software prefetching techniques for MPEG benchmarks. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(10):782–796, August 2000.