Computers and Electrical Engineering 000 (2016) 1–14

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Performance optimization of real-time video decoding ☆

S. Sankaraiah*, C. Eswaran

Center for Visual Computing, Multimedia University, Cyberjaya, Selangor 63100, Malaysia

ARTICLE INFO

Article history:
Received 8 October 2015
Revised 27 August 2016
Accepted 29 August 2016
Available online xxx

Keywords:
Dynamic scheduling
Speed up
Cache misses
Barrier
Scalability
Load balancing

ABSTRACT

One of the most challenging tasks in the multimedia domain is the full high definition (FHD) video decoding. A high level data parallelization technique using group of pictures (GOP) concept is proposed in this paper for the implementation of the FHD video decoder. Though GOP level parallelism has advantages such as higher scalability and independent frame decoding possibility, its requirement of higher memory resources remains its main drawback. In order to solve the short comings related to the problem of memory issues, a new dynamic memory scheduling algorithm for GOP level parallelism is proposed. This paper focuses on techniques for optimizing the memory requirements and speedup of FHD video decoding with GOP level parallelism. Experimental results show that the proposed dynamic memory scheduling technique reduces elapsed time significantly making real time FHD video decoding possible. Further, the results show that the proposed technique does not result in frame skipping and degradation in video quality. With the proposed technique, speedup values of 1.96, 3.96 and 7.95 are obtained for 2, 4 and 8 core processors respectively which are almost equal to the theoretical values.

1. Introduction

Nowadays, multicore architecture is widely used in the implementation of multi-tasking applications to enhance the performance. Multicore architecture provides a platform for speeding up the application by performing thread-level and data-level parallelism rather than by increasing the operating frequency of the processor. The H.264/AVC FHD video decoding process still remains a challenging task for current multicore processor architectures, especially in situations where the processor needs to provide real time performance with low cost implementations [1,2]. Various optimum-threading methods have been proposed to speed up the performance on multi-core platform [3–6]. In this paper, an optimization procedure is presented for FHD video decoding using group of pictures (GOP) level parallelism on multicore architecture. The strategy of implementing the real-time FHD decoding is to support video playback devices such as VLC player, media player and real-time FHD video conferencing. In this paper, problems such as optimum GOP size required, complexity of video decoding, and preservation of video quality are investigated. Main focus is to find the optimum number of frames to be used in each GOP for enhancing the decoding efficiency and minimizing distortion.

The remaining part of the paper is organized as follows. Section 2 presents the details of background works on decoder parallelization techniques. In Section 3, the design and implementation of GOP level parallelism using dynamic memory scheduling for FHD video decoding are explained in detail. Section 4 presents the results and discussions related to the

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Area Editor Dr. E. Cabal-Yepez.

* Corresponding author.
E-mail address: sankar2510@ieee.org (S. Sankaraiah).

http://dx.doi.org/10.1016/j.compeleceng.2016.08.022
0045-7906/

performance in terms of speedup, visual quality, thread utilization and memory optimization for reduction of cache misses. Finally, Section 5 presents the conclusion and future work.

2. Literature review

Video decoding is used in various systems, such as live video streaming for conferencing and video playback from media player. For the past few years, video compression techniques are becoming more and more complex as they strive to achieve the lowest bit-rates possible without degrading the video quality [2]. The complexity of the video decoding algorithms usually makes the video decoding process computation-intensive and time-consuming. Hence running such a process on a traditional single-core processor might overload the CPU significantly. Thus, to speed up the FHD video decoding process, parallelization techniques are required. According to Cor Meenderinck et al. [7], H.264 parallelization can be performed either at the task-level or at the data-level. When applying task-level parallelism, the H.264 decoding task is decomposed into smaller entities or sub-tasks [7,8]. Each sub-task is then assigned to a single processor or core for achieving parallelism. Task-level parallelism needs frequent communications and synchronizations between processors [9]. Thus, task-level parallelism will impose an extra overhead on the time needed to communicate and synchronize with the events occurring between the processors. Task-level parallelism also suffers from performance degradation when the resolution of the video increases from CIF (352×288) to FHD (1920×1080) [10,11]. Moreover, parallelizing H.264 decoder by using task-level parallelism for FHD is impractical, since load-balancing among the processors is more difficult to achieve due to the different execution times for different tasks [12,13].

When data-level parallelism is implemented on the video decoder, the data is decomposed into smaller chunks, so that the workload can be shared easily among the processor cores [14]. Data-level parallelism has better scalability feature when compared to task-level parallelism [15]. This is due to the huge amount of data that is contained in the video frame for processing. In H.264, the scalability of data-level parallelism increases with the resolution of the video [16]. Besides scalability, the load-balancing among processors can also be more easily achieved in data-level parallelism when compared to task-level parallelism since the time taken for processing of data is relatively the same [17,18]. In data level parallelism, each thread can be assigned to one processor core and executed in parallel. Thus, the speedup of video decoding process can be enhanced in accordance with Amdahl's law [19]. Theoretically, the speedup of the decoding process can be increased by a factor equal to the number of cores in the processor. As a result of the parallel video decoding implementation, the overall elapsed time of the decoding process is significantly reduced when compared to the sequential video decoding.

Data-level parallelism on H.264 decoder has been discussed in several papers [12–20]. Many researchers describe different data-level parallelisms that can be applied on H.264 decoder such as GOP level, frame level, slice level, and macro-block level. Out of these, GOP level parallelism is often favored for its fine granularity and its ability to prevent any video quality losses [10,16]. The advantage of GOP-level parallelism is that it does not have data dependencies since each GOP starts with a new video sequence [17]. Thus, there is neither communication overhead nor synchronization problem between the processors or cores [18]. Dumitrasand and Haskell [21] discussed the details of how many B frames can be used in between the two reference frames. As per the results of [21], the number of B frames required is 1 to 5, whereas, the results published by Yokoyama [22] show that the required B frames are 0 to 2. Huahui et al. [23], studied the effect of choosing GOP size by assessing the results of GOP on both static MPEG videos and MPEG videos streaming on a high loss network. Also, the results in [23] show that only two B frames can be used as reference frames, but P frames can be a maximum of 5. In GOP level parallelism, GOPs are assigned to different processors for concurrent decoding. Therefore, for a larger GOP size, GOP level parallelism requires a large amount of memory for storing the decoded frames, especially with higher resolutions [20]. The extra memory requirement will eventually cause cache pollution, which is one of the factors affecting the performance. It is shown in this paper that these problems can be solved if GOP level parallelism is implemented with dynamic memory scheduling in an optimized manner.

3. Design and implementation of dynamic memory scheduling

In the proposed implementation, Hantro 6100 H.264 decoder is chosen as the reference, since it is fully open-source and fully written in ANSI C standard and it can be easily ported to different platforms [24]. The proposed work focuses on the parallel decoding of H.264 for FHD, since this decoding process is computation-intensive and time-consuming. The Hantro 6100 H.264 decoder is modified using advanced data structures and some Windows supporting libraries so that it can display FHD on Windows platform without skipping any frames. The H.264 decoder is implemented using a dynamic memory scheduling (DMS) algorithm at GOP level for memory optimization to get higher speedup values and to meet the requirement of displaying the FHD resolution video in real-time. The main contribution of the proposed research work is to optimize the memory resources for decoding the FHD video with resolution 1920×1080 without affecting the quality. Fork-join model as specified by OpenMP standard is selected as the programming model [3–5]. The video rendering is implemented with the use of Simple Direct Media Layer (SDL) library.

Please cite this article as: S. Sankaraiah, C. Eswaran, Performance optimization of real-time video decoding, Computers and Electrical Engineering (2016), http://dx.doi.org/10.1016/j.compeleceng.2016.08.022