ON PARALLELIZATION OF A VIDEO MINING SYSTEM

Wenlong Li, Eric Li, Nan Di, Carole Dulong, Tao Wang, Yimin Zhang

Microprocessor Technology Lab, Intel Corporation
{wenlong.li, eric.q.li, nan.di, carole.dulong, tao.wang, yimin.zhang}@intel.com

ABSTRACT

As digital video data becomes more pervasive, mining information from multimedia data becomes increasingly important. Although researches in multimedia mining area have shown great potential in daily life, the huge computational requirement prohibits its wide use in practice. Since our personal computer is shifting from uniprocessors to multicore processors, exploiting thread level parallelism in multimedia mining applications is critical to utilize the hardware resources and accelerate the mining process.

This paper presents three different parallel approaches (task level, data slicing and hybrid parallel) to parallelize one widely used application in video mining system. The hybrid scheme, with the exploration of data level and task level parallelism, delivers much better performance than other two schemes. We get 10x performance improvement on a 16-way multiprocessor system. Besides, we perform several efficient optimization techniques, such as subexpression optimization, SIMD, and data blocking, to improve the performance by more than 60%. Therefore, our parallelization and optimization of the application makes it 16x faster than it used to be. Our study shows that with proper parallelization and optimization, multimedia mining can be used widely in our daily life soon.

1. INTRODUCTION

Broadcast soccer video is a popular program in commercial television broadcasts. With the advance of storage capabilities, computing power and digital home entertainment, the browsing and search in this kind of videos become more and more active[1, 2, 3, 4]. Through highlights detection, the consumers can retrieve specific video events quickly from the long videos and save time. However, the large input data and the underlying complex algorithm require more excessive computation than the commodity PC could afford. For example, it may take several hours to perform a task on one-hour MPEG-2 video raw file, which limits its wide use in practice. Our experiment shows detecting view type and playfield information from one hour of MPEG2 soccer video content takes 2 hours on today's platforms, but less than 8 minutes to obtain them on 16 way SMP system with proper optimization and parallelization.

In this paper, we propose three different schemes to parallelize the view type and playfield position detection application, an essential component in video mining system for soccer game highlight detection, and evaluate their effectiveness on 16 way shared-memory multiprocessor system. The hybrid scheme, with the combination of data level and task level parallelism, outperforms the other two candidates, delivering much better speedup than other two schemes.

2. OVERVIEW OF SOCCER HIGHLIGHT DETECTION SYSTEM

Soccer videos happened in restricted playfields with game rules and a defined broadcast layout. Taking advantage of the specific domain knowledge, soccer highlight detection is promising to achieve good performance in semantic content than other kinds of videos, e.g. personal home video and movies etc.

2.1. Framework of highlight detection

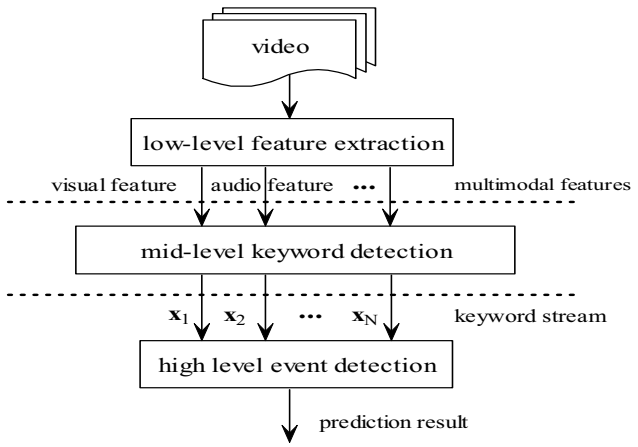

Figure 1. Overview system framework

We view a broadcast soccer video from the perspective of a program editor. Based on a predefined semantic intention, an editor combines certain multimedia layout and content elements to express highlight events. Hence, highlight detection can be viewed as a reverse process of authoring [5]. Fig.1 illustrates our system framework. To minimize the semantic gap between low-level features and high-level

events, we adopt the mid-level keyword representation [6]. The framework consists of three levels, i.e. low-level feature extraction, mid-level semantic keywords generation and high-level event detection. In processing, the low-level module first extracts audio/visual features from the video stream. Then the midlevel module uses the above audio/visual features to detect semantic keywords, e.g. view type, playing field position, excited speech, etc. Finally, the high-level module infers highlight events in the semantic space of these keyword streams.

2.2. Mid-level keyword generation

The mid-level module generates relevant semantic keywords from low-level audio/visual features. Details of keywords generation are described as following.

-x1 View type: By accumulating HSV color histogram, we get the dominant color to segment the playing field region. According to the area of playing field and the size of player, we then classify each shot into global view, medium view, close-up view and out of view [3, 4]. Fig.2 shows examples of these view types.

-x2 Play-position: We classify the play-position of those global-view shots into 15 regions as shown in Fig. 3. We first execute Hough transform to detect field boundary lines and the penalty box lines. A decision tree based classifier determines the play position according to lines' slope and position [2, 4].

Figure 2. From left to right, these are examples of global view, middle view, close-up view, out of view, and replay logo

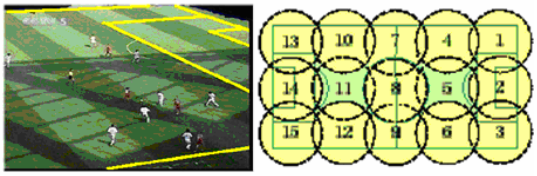

Figure 3. Hough line detection on a segmented playing field and fifteen regions of play field

-x3 Replay: It is an important cue for highlights, since replay usually follows a highlight. At the beginning and ending of each reply, there is generally a logo flying in high speed. We detect these logos to identify replay by dynamic programming [7].

-x4;5 Audio keywords: There are some significant sounds that have strong relations to some soccer highlights such as goal, foul, etc. Our system detects two types of audio keywords: commentator's excited speech, and referee's whistle. Gauss mixture model (GMM) and SVM classifiers are used to detect the above two keywords respectively from low-level audio features including Mel frequency Cepstral coefficients (MFCC), linear prediction coefficient (LPC) and pitch [1, 7].

2.3. Highlight detection

Highlights in soccer videos are the special events that audiences are especially interested in, e.g. goals, shoots, and free-kicks etc. A stack fusion framework detects these highlights according to the mid-level keywords in Section 2.2.

3. PARALLEL SCHEME

3.1. Application study

The application can be divided into three modules: video decoding, middle level keyword detection, and postprocessing. The video decoder first decodes continuous frames from the input video stream, and then the middle level keyword extraction module is followed to extract view type and playfield position at fixed frame interval. This process proceeds until all the frames are processed. Finally, all the view type and playfield results are fed into a postprocessing module to generate the continuous mid-level keywords. The execution time breakdown indicates that the former two modules are most time-consuming, constituting around 17% and 83% of total time for the application. On the other hand, the postprocessing module is extremely fast, therefore, is not considered in this parallelization work.

3.2. Task level parallel scheme

The working pattern of middle level visual keywords detection application resembles the producer-consumer model, where the video decoder works very similar to the task producer, generating a sequence of video frames, while middle level keyword extraction, acts as a task consumer, operating on the decode frames to detect view type and playfield position. Obviously, the working pattern well matches the well-known producer-consumer threading model.

This multithreading scheme works very similar to the task queue model provided by Intel OpenMP extension [8], which provides an efficient way to exploit task level parallelism. Fig.4 depicts a basic OpenMP taskQ working model, when all the threads encounter the taskq pragma, one is chosen to initialize task queue, and then the code inside the taskq block is executed single-threaded. When a task pragma is encountered within a taskq block, the code inside the task block is conceptually enqueued as a task and put in the queue. All the other worker threads will wait to fetch a task from the queue until task is available.

Besides the convenience of exploiting task level parallelism with TaskQ, the associated dynamic scheduling capability also enables to achieve good load balance in the parallel implementation. When the task queue is full, the producer thread will turn into the worker thread, which helps to

overcome the load imbalance and maximize the processor utilization dramatically.

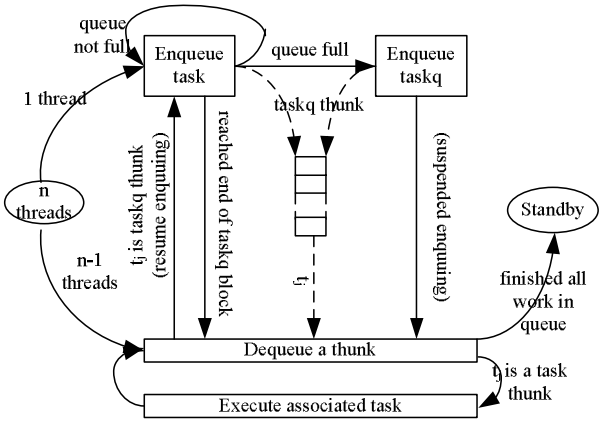

Figure 4. Execution model of task parallel scheme

Though the parallel scheme is simple and straightforward, it has some scaling limitations. Apparently, the maximal theoretical speedup of this parallel scheme is determined by the ratio of keyword extraction and video decoding module, indicating that the maximal speedup is at most 6x according to Amdahl's law [9].

3.3. Data slicing parallel scheme

Since the task level parallel scheme can't provide sufficient parallelism, we turn to explore the natural data parallelism to slice the raw data into several video chunks. Each thread uses similar routine as the serial application, but operates on different video chunk. In contrast to the task level parallelism, this parallel scheme provides more concurrency than the task level parallel scheme. However, it also has some deficiencies. First, it can't support enough number of threads when we cannot find the explicit sequence synchronization codes in the segmented video chunk. Second, with more video decoder and sliced data, the parallel overhead increases. Furthermore, it may also suffer from load imbalance since the computations are tightly coupled with the video content itself, e.g., the computation complexity of Hough line detection heavily depends on the number of lines contained in the frame. Finally, all simultaneous video decoders may excessively use the shared resources, such as saturating the system bus.

3.4. Hybrid parallel scheme

To take advantage of both task level and data slicing parallel scheme, we propose a hybrid parallelization approach to merge these two schemes together. At first, we decompose the video stream into several chunks, and then we use the same task level scheme on each particular chunk of data as illustrated in Section 3.2. Since the computation for each task differs with different input, we employ a centralized taskq to solve the load imbalance issue, where all tasks from all video decoders are put into one universal queue, while all the worker threads fetch tasks from this queue. The hybrid scheme works similarly to task level parallel scheme shown in Fig.4 except that we use multiple video decoding threads as the task producers.

4. PERFORMANCE CHARACTERIZATION AND ANALYSIS

This section examines the overall performance of parallel shot detection application. The measurements are conducted on a 16-way Intel Xeon shared-memory multiprocessor system. Each processor runs at 3.0GHz, and 8K L1 data cache, 512KB L2 unified cache, 4MB L3 unified cache. Every 4 processors share a 32MB L4 cache. As for the interconnect, the system uses two 4x4 crossbars. The input data are chosen from the TRECVID data suite [10] in MPEG-2 format.

For the software, we use Intel 8.1 OpenMP compiler tool chain, OpenCV and IPP library [11] to take advantage their highly optimized routines. Furthermore, we use Intel VTune[12] performance analyzer and thread profiler to guide the optimizations and qualify the parallel performance.

4.1. Performance optimization

Before studying the parallel performance, we first describe several optimization techniques to improve the application's performance. The profiling results indicate that the most time consuming module is Hough transformation. First, we apply sub-expression optimization, although it causes some extra memory requirement, it significantly reduces the repeated computations and achieves more than 30% performance improvement. Second, we employ SIMD instructions to avoid the penalty of floating point to integer data type conversion, and the performance is further improved by 24%. Finally, we apply data blocking, loop splitting, and data structure reorganization techniques, to increase the data locality, which contributes another 7% performance gain. As a result, the aggregated performance improvement is more than 60% comparing to the original application.

4.2. Scalability performance study

Fig.5 reports the achieved speedup for three different parallel schemes with 30 minutes MPEG-2 video dataset on up to 16 processors. As expected, the hybrid scheme delivers the highest scalability performance on 16 processors, whereas the task parallel scheme scales poorly beyond 8 processors. The data slicing scheme is much worse than the other two schemes on small amount of processors due to load imbalance issue, with the increase of processors, it performs a little better than the task scheme by providing more parallelism, but still worse than the hybrid scheme due to load imbalance problem and excessive use of shared resources.

Since the hybrid scheme outperforms the other two in terms of scalability performance, we will only study this scheme in the following.

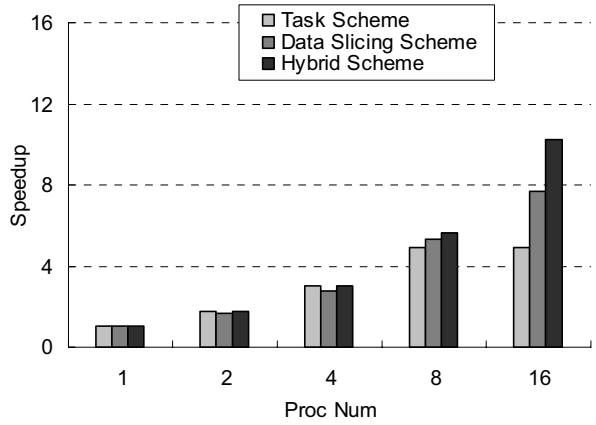

Figure 5. Speedup of three different parallel schemes with MPEG2 data input

To deeply understand the scaling limiting factors, we characterize the parallel performance from the high level general parallel overheads, e.g., synchronizations penalties, load imbalance, and sequential sections, to the detailed memory hierarchy behavior, e.g., cache miss rates and FSB (front side bus) bandwidth.

The profiling information suggests that the hybrid parallel scheme almost has no synchronization overhead and load imbalance problem as well as parallel overhead, the sequential area goes up steadily with the increase of processor number, but maintains at an extremely low percentage.

Besides the general scalability performance factors, memory subsystem also plays an important role in identifying the scaling performance bottlenecks. We profile the application with VTune, and performance metrics are chosen to be different level cache misses and system memory bandwidth. We find that the bus utilization rate increases linearly with the number of processors. In this application, for MPEG2 data, each thread holds about 15MB private data, and all threads share 7MB read-only data, which cannot fit in L3 cache, or even L4 cache with more than 2 threads. Therefore, when multiple threads running on processors within one cluster, they will compete for the L4 cache and excessively use the bus bandwidth, which accounts for the hybrid scheme only achieves inferior speedup on 16 processors. In the future, when we have multiple CPU-cores on the same die, interconnect bandwidth among these cores will be much higher. At that time, we believe the scaling limiting factor will be gone for this application.

5. CONCLUSION

In this paper, we presented a novel hybrid parallel scheme to accelerate the middle level keyword detection of soccer highlight detection system. With the exploration of thread level parallelism, the execution time is reduced by more than ten times to make the multimedia application run much faster than before.

Our study shows that with proper parallelization and optimization, multimedia mining can be used widely in our daily life soon. Future algorithm developers should keep future parallel programming in mind when they develop new algorithms. When everyone "multi-threads" the algorithm, we should be able to enjoy richer multimedia application in our daily life.

6. ACKOWLEDGMENTS

We would like to thank Bo Yang, Fei Wang, Prof Lifeng Sun, and Prof Shiqiang Yang of Dept. CS @ Tsinghua Univ for sharing with us their middle-level module codes studied in this work.

7. REFERENCES

[1] L. Duan, M. Xu, T.-S. Chua, Q. Tian, and C. Xu. A mid-level representation framework for semantic sports video analysis. In ACM Multimedia Conference, 2003.

[2] J. Wang, C. Xu, E.Chng, K. Wan, and Q. Tian. Automatic replay generation for soccer video broadcasting. In ACM Multimedia Conference, 2004.

[3] A. Ekin, A. M. Tekalp, and R. Mehrotr. automatic soccer video analysis and summarization. IEEE Trans. on Image processing, 12(7):796–807, 2003.

[4] M. Luo, Y. Ma, and H.J. Zhang. Pyramidwise structuring for soccer highlight extraction. In ICICS-PCM, page 1-5, 2003.

[5] C. G. Snoek and M. Worring. Multimedia event-based video indexing using time intervals. IEEE Trans on Multimedia, 7(4):638–647, 2005.

[6] X. Yang, P. Xue, and Q.Tian. Repeated video clip identification system. In ACM Multimedia 2005, pages 227–228, 2005.

[7] M. Xu, N. Maddage, C.Xu, M. Kankanhalli, and Q.Tian. Creating audio keywords for event detection in soccer video. In IEEE ICME 2003, volume 2, pages 281–284, 2003.

[8] E. Su, X. Tian, M. Girkar, et. al. Compiler support of the workqueuing execution model for Intel SMP architectures. In the fourth European workshop on OpenMP (EWOMP), 2002

[9] Y. Shi. Reevaluating Amdahl's law and Gustafson's law. Available at http://www.cis.temple.edu/~shi/docs/amdahl/amdahl.html

[10] W. Kraajj, A.F. Smeaton, P. Over, TRECVID 2004-An introduction, in TRECVID 2004 Proceedings, http://www-nlpir.nist.gov/projects/trecvid/

[11] Intel Corp. Intel® Integrated Performance Primitives (Intel® IPP). Available at http://www.intel.com/software/products/ipp

[12] Intel Corp. VTune performance analyzer. Available at http://www.intel.com/software/products/vtune