



Intel[®] Technology Journal

Tera-scale Computing

Media Mining—Emerging Tera-scale Computing Applications

Media Mining—Emerging Tera-scale Computing Applications

Yurong Chen, Corporation Technology Group, Intel Corporation
Eric Li, Corporation Technology Group, Intel Corporation
Wenlong Li, Corporation Technology Group, Intel Corporation
Tao Wang, Corporation Technology Group, Intel Corporation
Jianguo Li, Corporation Technology Group, Intel Corporation
Xiaofeng Tong, Corporation Technology Group, Intel Corporation
Patricia Wang, Corporation Technology Group, Intel Corporation
Wei Hu, Corporation Technology Group, Intel Corporation
Yimin Zhang, Corporation Technology Group, Intel Corporation
Yen-Kuang Chen, Corporation Technology Group, Intel Corporation

Index words: tera-scale computing, media mining, video processing, parallel computing, performance analysis

ABSTRACT

With the exponential increase in media data on personal computers and the Internet, it is critical for end users to efficiently manage metadata to find the information they are looking for. Media mining refers to a technique whereby a user can retrieve, organize, and manage media data. However, most media-mining applications are compute intensive, and they require tera-operations per second. This paper focuses on how tera-scale computing enables new usage models with media-mining techniques. Several representative media-mining usage examples are explored in detail.

First, we look at how these new usage models are enabled by a different kind of parallelism. For maximum performance, we provide a general parallel framework to abstract various parallelisms. We also present a detailed architectural performance analysis of several representative workloads on a dual-socket, quad-core system and on a 32-core Chip Multiprocessor (CMP) simulator. The results indicate that these media-mining applications have no obvious limits on concurrency and are amenable to future large-scale, multi-core architectures. They can take full advantage of tera-scale computing power in the form of thread-level parallelism to meet users' needs.

Because the underlying techniques and fundamental algorithms in media mining are widely used in other applications, many of our findings are applicable to other emerging applications as well.

INTRODUCTION

Rapid advances in the hardware technology of media capture, storage, and computation power have contributed to an amazing growth in digital media content. As content generation and dissemination grows, extracting meaningful knowledge from large amounts of multimedia data becomes increasingly important. Media mining is a kind of technology that helps end users search, browse, and manage large amounts of multimedia data [1]. It yields a wide range of emerging applications with various mass-market segments, e.g., image/video retrieval, video summarization, scene understanding, visual surveillance, digital home entertainment, smart health care, etc. Most of these applications are very complicated and have real-time or even super-real-time processing demands, which require tera-scale computing power to make them usable.

In this paper, we present several media-mining applications that require target architectures capable of delivering tera-scale computing. Our study shows that today's single-core processor system performance is 10x–1000x slower for acceptable human interactions. To accelerate these compute-intensive applications, we exploit the inherent data and function parallelism of these workloads. Our experiments show that with proper parallelization, these workloads can scale well, achieving a speedup of up to 7.5x on a 2-socket, quad-core machine and a speedup of up to 30x on a 32-core CMP simulator.

This paper is organized as follows. First, we explore several media-mining usage models and their key techniques. Next, we present several different parallel schemes and a general parallel video-mining framework. Then, we show our performance analysis results of the parallelized workloads.

MEDIA-MINING APPLICATIONS

Media mining has a huge number of emerging applications with different usage models. We highlight three typical usage models developed at Intel.

Media-Mining Usage Models

- **Sports video analysis:** Broadcast sports videos are very popular on television. Using highlights detection, consumers can quickly retrieve specific video clips without having to browse through the whole video. Sports video analytics can be viewed from the perspective of an editor. Based on a predefined semantic intention, an editor combines certain multimedia content elements and their temporal layout to achieve the desired highlighted events. Hence, detecting highlighted events is similar to a reverse process of authoring. The system framework consists of three levels: low-level audio/visual feature extraction, mid-level semantic keywords generation, and high-level event detection [8]. To minimize the semantic gap between low-level features and high-level events, we use mid-level semantic “keywords” followed by a classifier to infer events of interest. Our sports video analysis system can work with a multitude of sports including soccer, hockey, badminton, tennis, and diving. Given a video in a specific domain with predefined semantic intentions, the system can extract the desired events and features and interpret a summarization output video in terms of high-level semantics.
- **Personal video editing:** Home videos are increasingly popular as digital video cameras become more user friendly and portable. However, because home videos for the most part are shot by amateurs, shaking, blurring, under-exposure artifacts, and redundant content are always present. Therefore, the demand for an automated home video editing system [2] is high. Such a system has to be able to recognize how many people and how many scenes are involved, mine the relationship between various people and scenes, and synthesize a short artistic video clip from a long raw video. A typical personal video editing system includes three key modules: intelligent analysis, adaptive selection, and seamless composition. The first module extracts the multi-modal and multi-level audio-visual features; the second module selects the most interesting,

important, and informative content; and the third module produces a near-professional story with incidental music. The overall automated home video editing system must be easily extended to the personal video recorder and digital home entertainment system.

- **Personal video retrieval:** A personal video retrieval system is a desktop application that works much like the Google desktop search to help end users manage more and more personal multimedia data from all kinds of mobility digital camera devices. In response to a user query, the personal video retrieval application finds the relevant video clips from a large video database such as from movies, TV, sports games, and home videos. Generally, a retrieval system first extracts low-level audio/visual features from videos, and then detects semantic concepts (keywords) to represent the video content. Finally, a query engine returns retrieval results based on the user’s query and on a similarity model. The query can be text keywords, image examples, hand-drawn sketches, or short video clips, and the output is relevant video clips ranked not only by their content similarity to the query, but also by their importance, according to a concept-link relationship analysis. To gradually improve system performance during the query procedure, the system provides user-friendly relevant feedback and active learning modules.

Key Media-Mining Techniques

Although the above usage models are quite different from one another, the underlying technologies are common and can be extended to a broad range of media-mining applications. In this paper, four key techniques are extracted from previous usage models to show how media-mining applications are built.

- **Sports keyword detection:** The mid-level module generates semantic “keywords” from the previously described low-level extraction. Listed below are some keywords in sports video analysis. These keywords are used as input for high-level event detection.
 - View type: Based on color histograms of each frame, we can obtain the dominant color to segment the playing field region. We then classify each frame as a global view, medium view, close-up view, and out of view [5].
 - Play-field: A Hough transform from digital image processing is used to detect field boundaries and penalty box sections. Then a decision-tree-based classifier determines the play

position according to the slope and position of the lines.

- **Replay:** In broadcast sports videos, to capture clues for significant events, there typically is a replay following an important event. At the beginning and end of each replay, there is generally a logo flying in high speed. We detect logos to identify replays by discovering repeat video segments through dynamic programming [6].
- **Audio keywords:** There are two types of audio keywords: commentator's excited speech and referee's whistle: these have a strong correlation to key events in the game such as a foul, a goal, or player entanglements. A Gauss Mixture Model (GMM) is used to detect keywords from low-level audio features including Mel frequency Cepstral coefficients (MFCC), energy, and pitch [7].
- **Human detection and tracking:** Human detection and tracking is a significant and challenging task in many application scenarios. Different from rigid objects, humans are articulated and jointed by several human-parts, which may lead to pose variance, self-occlusion, etc. In human detection, the first problem is to select the proper features to characterize human regions/parts: Haar wavelets [3] and orientation histograms are mostly used to do this. The second problem with human detection is to use a discriminator to determine whether there are humans and where they are if they are present. The Boosting learning-based detector is preferred [3]. It is an aggressive learning algorithm that produces a strong classifier by choosing features in a family of simple classifiers and combining them linearly. Then a cascaded structure is introduced in order to quickly reject the background regions. Human tracking is essentially finding body regions or parts that correspond with successive frames by using data association and occlusion inference techniques.
- **Face detection and tracking:** Face detection and face tracking have been an important technology and pre-requirement for many person-analysis relevant applications, such as face recognition/identification, emotion analysis, and cast indexing. Face detection has been studied for many years. Viola and Jones Boosting learning-based detection algorithms are the most successful algorithm to date [2]. Recently, some improvements are proposed to enable the algorithm to handle multi-view faces more efficiently for high-quality videos [12]. Generally, Boosting-based face detection characterizes image regions by very simple Haar wavelet features, and it learns cascade detection from a training set to

separate a face set from a non-face set. In the detection phase, the learned detector will slide by a window over the image to detect whether the window contains a face or not. Face tracking [13] is an extension of face detection technology, which can detect a person's continuous faces from a video sequence. Spatial and temporal constraints are employed to avoid much unnecessary calculation. Since it detects faces only in predicted face image regions, it doesn't waste time scanning all the positions of every frame.

- **Concept ontology indexing:** Concept ontology indexing represents multimedia data by large-scale concept ontology for indexing and fast retrieval. There are several concept lexicons for multimedia: large-scale ontology for multimedia (LSCOM) [9] is the most popular. LSCOM currently contains about 1000 concepts that are relevant to objects, people, locations, scenes, and events. LSCOM has been successfully used by the TREC video retrieval evaluation (TRECVID) hosted by NIST [10]. Concepts are detected from more than 20 low-level MPEG-7 compatible audio/visual features, e.g., color histogram, Gabor texture, shape context, edge histogram, motion, and MFCC audio features, etc. Given these low-level features, a supervised classifier (such as an SVM) is learnt for each concept from a training set to identify whether the concept exists or not in each video shot [11]. Employing all of the concept detectors, a video shot is therefore represented and indexed by the semantic concept ontology that makes next-stage search similar to text retrieval.

Common Characteristics in Media Mining

Three attributes of media-mining applications can be summarized as follows:

- First, a media-mining system is basically a bottom-up framework as shown in Figure 1. The framework is a three-layer architecture, i.e., low-level feature extraction, mid-level semantic keywords detection, and high-level concept detection. In processing, low-level visual/audio/textual features are extracted from raw media data. Then in the second layer, mid-level features or keyword concepts are detected from low-level features to bridge the semantic gap between low-level features and high-level concepts. Finally, high-level modules infer the desired concepts in the semantic keyword spaces.

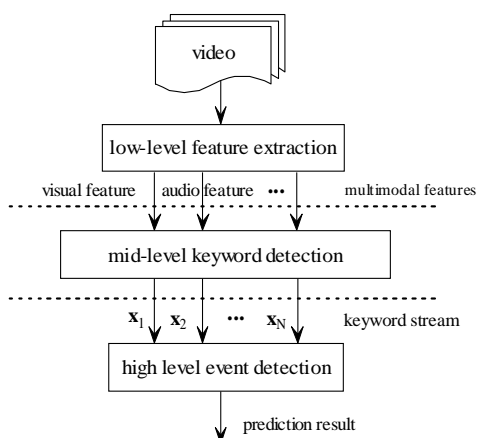


Figure 1: General video-mining framework

- Second, media mining is a hybrid technique of computer vision, pattern recognition, machine learning, and data mining. For example, human detection/tracking techniques involve Haar and HoG feature extraction from video frames, Boosting (cascade learning) training-based candidate detection, and associate rule learning from quite large examples to identify relationships between articulations. In these techniques, Haar and HoG features are essentially computer vision methods; Boosting is a famous machine-learning algorithm; and associate rule learning is a typical data-mining method.
- Third, media-mining applications usually combine multiple components. For example, in the automatic home video editing application, the application needs to recognize people, mine the relationship between people, and synthesize a short artistic video clip from a long raw video.

Media mining has mass-market potential and is therefore quite a suitable and important proxy not only for workload analysis on future architectures, but also for developing parallel programming models for multimedia applications. Furthermore, due to its similar framework for different usage models, we only use one technique as an example to study its computational requirements.

Computational Requirement: a Case Study

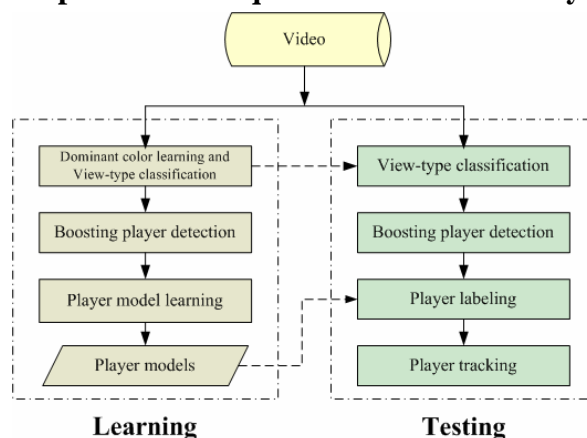


Figure 2: Flowchart of player detection, tracking and classification

In the sports domain, we look at multiple player detection, tracking, and classification in broadcast soccer video for our example. Its flowchart is shown in Figure 2 [4]. To make the algorithm robust and adaptive, we construct the background (playfield) color model and three player appearance models (Team A, Team B, and Referee) through unsupervised learning procedures. In the learning phase, the background color model is obtained by accumulating color histograms over hundreds of frames in the video in HSV color space. Player appearance models are learned by player sample collection with a boosted player detector, color histogram representation, and clustering. In the testing phase, we first perform background segmentation, playfield extraction, and view-type classification. Only global views are selected for player detection. We then apply a boosted cascade of Haar features for player detection on each foreground pixel within the playfield. Multiple detections will usually occur around each player after scanning the image. We merge adjacent detected rectangles and get final detections with proper scale and position. In the player classification procedure, each player sample is represented by the learned codebook histogram. We calculate the Bhattacharyya distance between the histogram and each sub-model. The player sample is assigned the sub-model's label by the nearest neighbor rule. With this procedure, players are labeled as Team A, Team B, Referee, or Outlier (if the minimum distance is larger than a threshold). Player tracking is performed by efficient forward and backward nearest neighbor data association. We take both binary mask overlap and color histogram intersections in player upper-body as observations within a certain spatial displacement range to find the optimal player regions correspondence, and we generate players' trajectories across frames.

Figure 3 is an example of player tracking results, in which white ellipses and rectangles indicate two teams' players and a black rectangle is the referee.

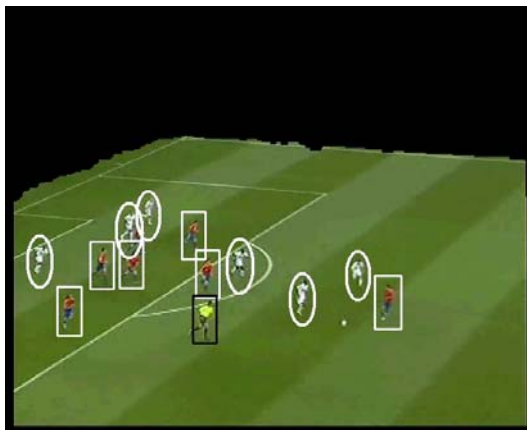


Figure 3: Player tracking results on soccer video

Player detection is achieved by background elimination and a boosted cascade of Haar features. In this paper, we only show the detailed detection procedure since this procedure is most compute intensive compared to tracking and classification. The cascade detector with multiple stages has the capability of quickly rejecting the regions and focus on the harder-to-classify windows. The number of features selected in each stage is different depending on the expected performance and sampling criterion. Therefore, increasingly complex classifiers are combined sequentially. This improves both the detection speed and efficiency.

- Input: image frame, background model
- Playfield elimination and view-type classification
- Player detection
 - For each scale
 - Scan each point to be detected
 - For each point
 - Evaluate its response with cascaded stages
 - Calculate normalized constant
 - For each stage
 - Evaluate the response
 - For each selected Haar feature
 - Calculate Haar feature response
 - Normalize Haar feature response
 - Get weak classifier response
 - Accumulate all Haar response
 - If verified by the threshold, begin next stage; else, label the point as negative, break;
 - If pass all stages, label the point as positive
- Post-processing to merge adjacent detection instances
- Output: vector of player regions (rectangles)

Based on the above description, one can easily infer its computation complexity, which is proportional to the size

of the video frame, the number of weak classifiers, and the number of scales. For player tracking between two adjacent frames, it is proportional to the number of players and player size. For player classification, it is linear to the number of players, player size, size of codebook, and size of sub-model. For an MPEG-2 video, the frame size is 720x576; we use about 1000 weak classifiers and three different scales. Thus, one minute of MPEG-2 video will need 1.86 tera-operations. Its serial processing speed on today's processors is about 3 frames per second, which is 10x slower than real-time.

MEDIA-MINING PARALLEL FRAMEWORK

With the boom in multi-core processors and the prevalence of shared memory processing, it is important to exploit thread-level parallelism within applications to take advantage of next-generation microprocessor capability. In this section, we present the parallelization methodology, characterize different parallel schemes, and provide insights for parallelizing these media-mining applications on future multi/many-core systems. Besides the parallelization study, we also made intensive optimizations, e.g., genetic loop-level optimization, SIMD acceleration, and cache-conscious optimizations, to provide a fully optimized baseline for further workload parallelization and analysis.

Video-Mining Parallelism

Most video-mining applications can be partitioned into three modules: video decoding, feature extraction, and post processing. We use an MPEG-2 video decoder to divide the input video stream into a number of consecutive decoded frames. Then we use a feature extraction module to extract a set of visual features from these decoded frames. This process continues until all the frames are processed. Finally, all the feature results are fed into a post-process module to detect the final visual information. The breakdown of execution time indicates that the video decoding and image processing modules are the most time consuming. The post-processing module is extremely fast and is therefore not the focus of this paper.

We use a top-down analysis methodology to analyze the coarse-grained parallelism in each module and the whole application. In general, people tend to use data-domain decomposition rather than functional-domain decomposition to take advantage of the inherent parallelism in multimedia applications. Though fine-grained parallelism within each module is of interest, we don't explore this kind of parallelism as it's not profitable because of serial regions and insufficient parallelism in these modules. Therefore, we choose

coarse-grained parallelism to explore both functional domain decomposition and data domain decomposition within each task with the goal of load balancing, and we examine the issue of scalability when using a large number of processors.

A **task-level parallel scheme** uses the producer-consumer model, where the video decoder works as a producer, generating a sequence of video frames, while the image processing modules act as a consumer, operating on decoded frames to obtain the corresponding feature information for each frame. This multi-threading scheme is very similar to the task queue model provided by the Intel® OpenMP extension [14], which provides an efficient way to exploit functional-domain decomposition. The video decoder serves as the task producer to encapsulate the decoded frame as a task and conceptually put it in the task queue, and all the other worker threads will wait until the task is available. Though this parallel scheme is straightforward, it may experience limited scalability performance on a large number of processors when the ratio between feature extraction modules and video decoders is not sufficiently high.

A **static data slicing parallel scheme** slices the raw data into several video bit-stream chunks. Each thread performs a similar routine to that of a sequential application: decoding the bit-stream chunk and extracting features from the decoded frames. Because the raw video stream is split manually, each thread has to find the new sequence synchronization position. Therefore, there is an explicit synchronization between two adjacent threads to guarantee no excess or loss of decoded frames. In addition, the static data-slicing scheme may experience a load imbalance problem when the work is not evenly distributed across threads.

To take advantage of both task-level and data-level parallel schemes, a **dynamic hybrid parallelization approach** is proposed to combine these two schemes. At first, we decompose the video stream into several chunks to exploit the data-domain decomposition, and then we exploit the functional-domain decomposition on each particular chunk of data as previously mentioned. In this parallel scheme, there are multiple queues to buffer tasks. Master threads are responsible for video decoding and for enqueueing the feature extraction tasks in different queues. Worker threads fetch tasks from their associated queue and execute the tasks. Further, in order to reduce load imbalance, we use the work stealing strategy. When one queue is empty, it will steal tasks from other non-empty queues and execute the tasks on the idle physical processor. With work stealing enabled, the load imbalance disappears. In addition, due to reduced contention on the access to each queue, the synchronization overhead is reduced significantly.

Figure 4 illustrates the hybrid task-stealing scheme. The whole video is partitioned into four chunks and assigned to four thread groups. Within each thread group, a task queue is implemented with one master thread and three worker threads. The task will not be migrated to other queues unless its private task queue is out of tasks.

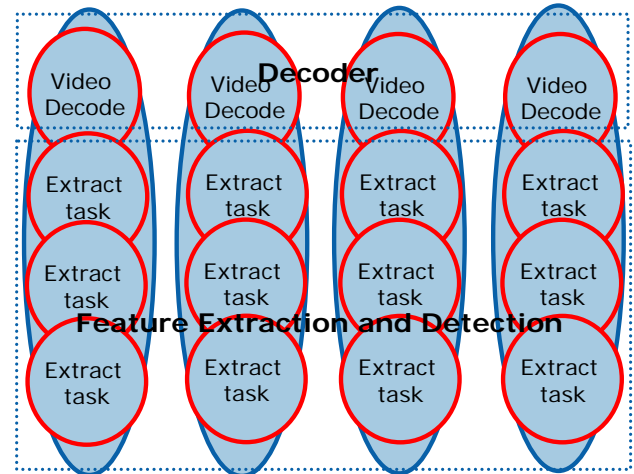


Figure 4: Dynamic hybrid task-stealing scheme

In summary, the dynamic hybrid parallelization scheme has several advantages. First, it significantly improves performance by manipulating multiple queues of video data in parallel, which reduces the competition for shared resources. Second, it solves the load imbalance issue by enabling dynamic task scheduling and stealing. Finally, the hybrid scheme provides enough flexibility by specifying the number of decoding and worker threads to maximize system resources utilization and deliver good scalability. However, from the perspective of programming, this dynamic hybrid parallelization approach is the most difficult of the three parallel schemes to build.

Parallel Pattern in Video Mining Applications

Because of the difficulty of parallelization in these media-mining applications, we construct a universal parallel video-mining framework to encapsulate the parallel scheme and provide an ease-of-use interface to the programmer.

The video-mining parallel framework [15] is built in C++, and OpenMP is the default parallel language. It includes the parallel implementation, an abstract interface, and a set of configuration parameters. There are four primary components in this framework:

- An image-processing engine that serves as the interface to invoke the user codes in the library and

perform feature extraction functionalities for each decoded frame.

- A video-decoding engine that acts as an interface to enable most video codec standards with parallel support.
- A portal video-mining function that serves as an interface to link the user codes with the framework.
- Configuration parameters and core image data structures.

The parallel video-mining framework has several advantages. It provides a unified parallel computing environment for video-mining applications. Programs written in this framework can be automatically parallelized and efficiently executed on a multi-core architecture. The run-time library takes care of the details. Furthermore, this framework is easily extensible and maintainable. The programmer can extend it to meet new requirements.

To summarize, video-mining workloads have abundant parallelism. The dynamic hybrid parallelization approach that combines both functional-domain decomposition and data-domain decomposition can achieve optimal parallel performance. In addition, the particular execution pattern of video-mining applications can be abstracted into a parallel video-processing framework to help programmers easily construct a parallelized video-mining application.

PERFORMANCE ANALYSIS ON MULTI-CORE SYSTEMS

In this section we analyze three typical media-mining workloads (Player, Face and Shot detection), which are parallelized via our video-mining parallel framework. To generate best-performing executable codes, the Intel 9.1 OpenMP compiler tool chain and highly optimized OpenCV and IPP library [16] are used. Furthermore, we also use the Intel VTune™ Performance Analyzer [17] to identify the hotspots in functional profiling and guide the optimizations. To characterize the parallel performance, the Intel® Thread Profiler is used to quantify the parallel performance metrics, i.e., synchronization, locks, load imbalance, etc.

We evaluate the scaling performance of these parallel media-mining workloads on a real multi-core machine and a large-scale CMP simulator. The multi-core platform is a dual-socket, quad-core machine, with two Intel® Core™2 Quad processors running at 2.33GHz. Each socket has four cores, and each core is equipped with a 32KB L1 data cache and a 32KB L1 instruction cache. The two cores on one chip share a 4MB L2 unified cache. The maximum Front-Side-Bus (FSB) bandwidth is 21GB/s. In addition to the existing multi-core system, we

further study these media-mining applications' performance on a large-scale CMP simulator with cycle-accurate simulation to see how they will scale with the increasing number of cores. We assume a very high main memory bandwidth so that we do not artificially limit the scalability of the modules.

For the workloads studied in this experiment, we choose application parameters and datasets so as to represent realistic executions. For Player detection, we used a 30-minute MPEG-2 soccer video as the input. For Face and Shot detection, we used a 10-minute MPEG-2 movie video as the input.

Performance Scalability Analysis

Our video-mining workloads scale very well as the number of threads increases, as shown in Figures 5 and 7. That is, media-mining applications can efficiently use the computational power provided by multi-core processors.

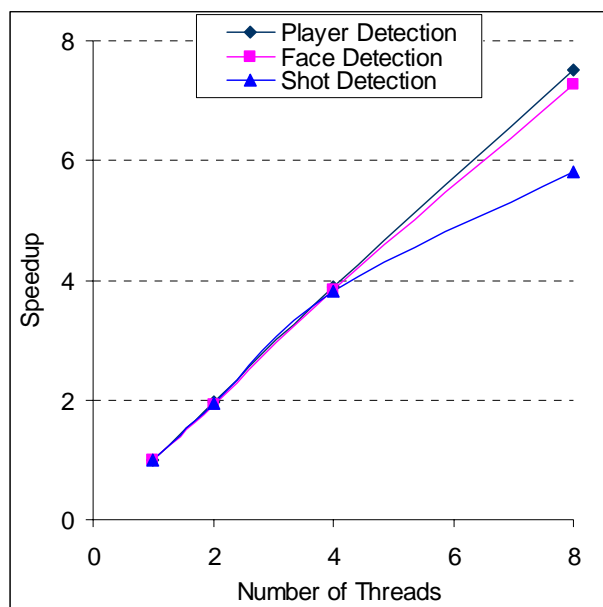


Figure 5: Scalability of parallel video-mining workloads on an 8-core system

However, as also shown in Figure 5, our workloads, in particular, Shot detection, do not have linear scaling on the 8-core system. To fully understand the scaling-limiting factors on an 8-core system, we characterize the parallel performance from the perspective of the high-level parallelization overhead, e.g., synchronization penalties, load imbalance, and sequential regions, and from the detailed memory behavior, e.g., cache miss rates and FSB bandwidth.

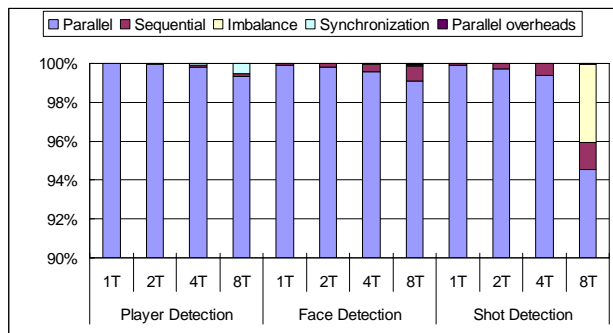


Figure 6: Execution time breakdown

In general, our parallelized workloads expose good parallel performance metrics. Figure 6 depicts the parallel profiling metrics for these three workloads. The higher the parallel region, the better speedup can be achieved on highly threaded architectures. Shot detection has slightly more load imbalance than other workloads. Because of frame dependency, it is more challenging to implement two-level task queues in Shot detection than in other workloads. In Shot detection, we use the static scheduling scheme, which leads to a slightly higher load imbalance. Nonetheless, the profiling information suggests these parallel video-mining workloads expose good parallel performance metrics. If we assume the parallel region can scale perfectly, the three workloads should achieve the theoretical speedups of 7.95, 7.93, and 7.56, respectively, on eight cores. They are higher than the results shown in Figure 5. Therefore, we believe the scalability of our workloads is limited by some other factors that are discussed in the next subsection.

On the simulated 32-core CMP system with a huge amount of memory bandwidth, two selected parallel video-mining workloads have very good scalability, as depicted in Figure 7. First, the size of the serial sections in the applications is reasonably small—the serial code accounts for much less than 1% of the execution time for the one-thread runs. Second, there is little contention on the locks: the locking overhead does not increase with the thread number due to coarse-grained parallelism. Third, the load imbalance is not a major issue; most of our video-mining workloads adopt a dynamic hybrid parallelization scheme. In short, when we assume a very high main memory bandwidth so that we do not artificially limit the scalability of the workloads, these applications scale very well.

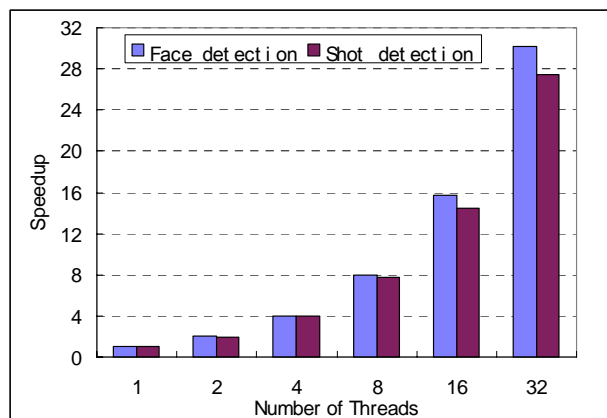


Figure 7: Scalability of two video-mining workloads on a 32-core CMP simulator

Memory Behavior Analysis

Besides the general parallel performance metrics, the memory subsystem also plays an important role in scalability. As shown earlier in Figure 6, our workloads with good parallel performance metrics should achieve the theoretical speedup of 7.6–7.9x on 8 cores, if the parallel region can scale perfectly. We now investigate why these workloads cannot achieve this perfect scaling performance from the perspective of the memory subsystem. We use the Intel VTune Performance Analyzer and a command-line tool for hardware-based performance counter sampling to further analyze the memory behavior of the applications on the real system, e.g., system memory bandwidth and L1/L2 cache miss rates.

Our first observation is that average bus bandwidth is not limiting the scalability of these workloads on the 8-core system. Figure 8 shows how the average FSB bandwidth utilization varies with the number of threads. The bandwidth usages of all workloads are far below the 21GB/s capacity supported by the system. This seems to indicate bus bandwidth does not limit the scalability of our workloads on the 8-core system.

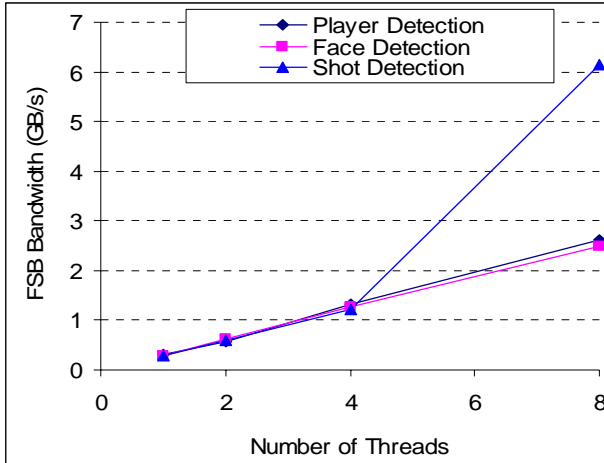


Figure 8: Average FSB bandwidth utilization vs. number of cores

Although workloads are not bounded by the average bandwidth usage, the scalability is limited by the instantaneous bandwidth usage. We perform interval sampling of the memory subsystem behavior over time. Figure 9 shows a representative phase of the bandwidth usage over time for the single-threaded Shot detection workload on a single core. It goes without saying that there are some bursty memory access behaviors—the instantaneous bandwidth usage is much higher than the average bandwidth usage. In particular, one of the modules demands about 7x more bandwidth over the average bandwidth. When the bandwidth demand of the module is higher than the system’s capability, its speedup from 8 cores is less than 3x, and it becomes the bottleneck of scalability. In short, the workload is not able to scale perfectly when the instantaneous bandwidth usage is higher than the system’s capability. This is what limits the scalability.

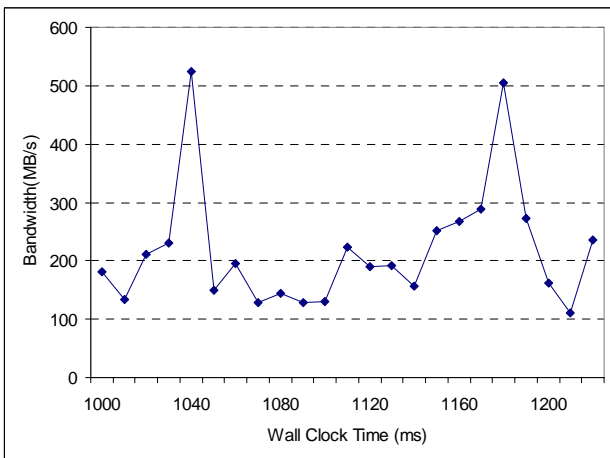


Figure 9: Bandwidth usage over time for single-threaded Shot detection workload

Additionally, there is a significant increase in bandwidth usage from four threads to eight threads for Shot detection. Figure 10 shows that L1 cache miss rates vary little with the number of threads, while L2 cache performance deteriorates when scaling the thread count. In particular, the external memory access rate for Shot detection increases from 0.05 bytes per instruction for a single thread to 0.30 bytes per instruction for eight threads. Because we exploit coarse-grained parallelism for these three workloads, each thread operates on a large private working set, about 32MB per thread for Player detection, 8MB per thread for Face detection, and 4MB per thread for Shot detection. As the total working set size increases with the number of threads, there are more L2 cache misses for more threads. For Shot detection, while the working set of four threads fits well into 16MB L2 caches, the working set of eight threads cannot fit. This explains the significant increase in cache misses from four threads to eight threads. Together with the instantaneously high bandwidth usage, the speedup of Shot detection from four threads to eight threads is much slower than the speedup from two threads to four threads.

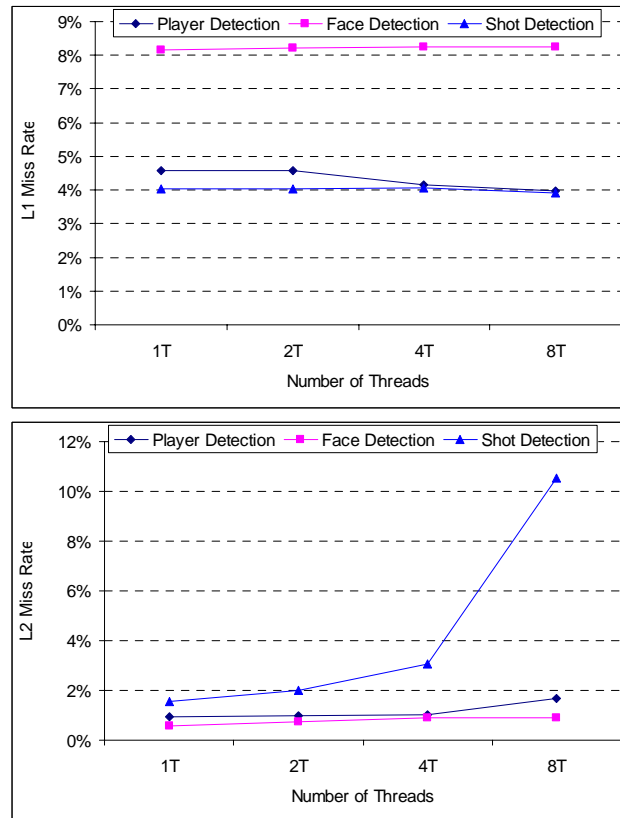


Figure 10: L1/L2 cache miss rates

To summarize, most of the video-mining workloads demonstrate fairly good parallel performance on both existing multi-core systems and future large-scale CMP platforms. As most of them can be partitioned into a large

number of parallel tasks, they have little lock overhead and serial region. Since the workloads are parallelized in coarse-grain fashion, which exposes a huge working set with the increase in thread numbers, large cache size and sufficient memory bandwidth will be necessary to enable large-scale, video-mining computing. To reduce the working set sizes and the external bandwidth usage in the future, we may need to exploit fine-grain parallelism. This could be a tradeoff between memory subsystem performance and parallelism overheads.

CONCLUSION

Media mining can help us retrieve, organize, and manage the exponentially growing media data easily. We explored several usage models in media mining and showed that most applications require tera-scale computing. To efficiently use the processing power provided by multi-core processors, we studied common parallelization schemes and proposed a general parallel framework for these media-mining applications. Furthermore, we conducted a performance analysis of several representative media-mining workloads on an 8-core system and a 32-core CMP simulator. Our analysis shows they have no obvious limits on parallelism. With a proper parallelization scheme, future large-scale CMP systems can deliver real-time performance for these media applications. Taking advantage of next-generation tera-scale computing platforms, new usage models in media mining will be enabled.

ACKNOWLEDGMENTS

We acknowledge the encouragement and help that we have received from Dr. Bob Liang, Director of the Applications Research Lab. Our thanks also go to Prof. Haizhou Ai, Prof. Jianming Li, Prof. Zhijian Ou, Prof. Lifeng Sun, Prof. Shiqiang Yang, and Prof. Bo Zhang from Tsinghua University for collaborating with us on the media-mining project and providing some original source code for our analysis. We also thank the reviewers for their valuable comments.

REFERENCES

- [1] Chabane Djeraba, *Multimedia Mining: A Highway to Intelligent Multimedia Documents*, Kluwer, Norwell, 2002.
- [2] X.S. Hua, L. Lu, and H.J. Zhang, "AVE - automated home video editing," *ACM Multimedia*, 2003.
- [3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *IEEE Int'l Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.
- [4] J. Liu, X. Tong, W. Li, T. Wang, and Y. Zhang, "Automatic player detection, labeling and tracking in broadcast soccer video," accepted by *BMVC* 2007.
- [5] A. Ekin, A.M. Tekalp, and R. Mehrotr, "Automatic soccer video analysis and summarization," *IEEE Trans. on Image Processing*, 12(7), July 2003, pp. 796–807.
- [6] H. Bai, W. Hu, T. Wang, X. Tong, and Y. Zhang, "A novel sports video logo detector based on motion analysis," *International Conference on Neural Information Processing (ICONIP)*, 2006.
- [7] M. Xu, N. Maddage, C. Xu, M. Kankanhalli, and Q. Tian, "Creating audio keywords for event detection in soccer video," *IEEE International Conference on Multimedia & Expo (ICME)*, 2003.
- [8] J. Li, T. Wang, W. Hu, M. Sun, and Y. Zhang, "Two-dependence Bayesian network for soccer highlight detection," *IEEE International Conference on Multimedia & Expo (ICME)*, 2006.
- [9] L. Kennedy, "LSCOM lexicon definitions and annotations (version 1.0)," *DTO Challenge Workshop on Large Scale Concept Ontology for Multimedia, Columbia University ADVENT Technical Report #217-2006-3*, March 2006.
- [10] NIST, TREC Video Retrieval Evaluation, at <http://www-nlpir.nist.gov/projects/trecvid/>*
- [11] J. Cao, Y. Lan et al., "Intelligent multimedia group of Tsinghua University at TRECVID 2006," in *Proceedings TRECVID*, 2006.
- [12] C. Huang, H. Ai, et al., "Vector boosting for rotation invariant multi-view face detection," *IEEE International Conference on Computer Vision (ICCV)*, 2005.
- [13] Y. Li, H. Ai, C. Huang, and S.H. Lao, "Robust head tracking with particles based on multiple cues fusion," *HCI/ECCV 2006, LNCS 3979*, pp.29–39.
- [14] E. Su, X. Tian, M. Girkar, et al., "Compiler support of the workqueuing execution model for Intel SMP architectures," *4th European workshop on OpenMP (EWOMP)*, 2002.
- [15] E. Li, W. Li, C. Dulong et al., "User transparent parallel video mining library," *PMUP workshop*, 2006.
- [16] Intel® Integrated Performance Primitives, at <http://www.intel.com/software/products/IPP>
- [17] Intel® VTune™ Performance Analyzer, at <http://www.intel.com/software/products/VTune>

AUTHORS' BIOGRAPHIES

Yurong Chen is a researcher at the Microprocessor Technology Lab, Beijing. Currently, he conducts research on parallel processing of emerging applications, scalable workloads, and benchmarking and performance analysis for next-generation microprocessors/platforms. He joined Intel in 2004. Before that he did two years' postdoctoral research on large-scale scientific computing in the Institute of Software, Chinese Academy of Sciences. He received his Ph.D. degree from Tsinghua University in 2002. His e-mail is yurong.chen at intel.com.

Eric Li is a researcher in the Microprocessor Technology Lab, Beijing. Currently, he is working on media-mining technology development and performance analysis on multi-core architecture. Prior to this, he was involved in several projects related to bioinformatics, multimedia, and parallel computing. He received his M.S. degree from Tsinghua University in 2002 and joined Intel that same year. His e-mail is eric.q.li at intel.com.

Wenlong Li is a researcher in the Microprocessor Technology Lab, Beijing. Currently he is working on algorithmic and workload analysis on data-mining applications. Before this, he did research in loop compilation techniques for IPF architecture. He received his Ph.D. degree from Tsinghua University in 2005 and joined Intel that same year. His e-mail is wenlong.li at intel.com.

Tao Wang is a researcher in the Microprocessor Technology Lab, Beijing. Currently, he conducts research on video-mining, computer-vision, and machine-learning techniques. At Intel, he has been involved in several projects related to visual tracking, bioinformatics, soccer highlights detection, cast indexing, video summarization, and concept detection, etc. He received his Ph.D. degree in Computer Science from Tsinghua University in 2003 and joined Intel the same year. His e-mail is tao.wang at intel.com.

Jianguo Li is a researcher in the Microprocessor Technology Lab, Beijing. Currently, he works on multimedia mining and parallel algorithm design and implementation. He has been involved in several projects related to sports video analysis and content-based media mining/retrieval. He received his Ph.D. degree from Tsinghua University in June 2006 and joined Intel after graduation. His e-mail is jianguo.li at intel.com.

Xiaofeng Tong is a researcher in the Microprocessor Technology Lab, Beijing. His work is on personal desktop multimedia applications. His technical interests include computer vision and pattern recognition. He received his Ph.D. degree in Computer Science from the Institute of Automation, Chinese Academy of Sciences in 2006. His e-mail is xiaofeng.tong at intel.com.

Patricia P. Wang is a researcher in the Microprocessor Technology Lab, Beijing. Her current research focuses on video mining, machine learning, and pattern recognition. She joined Intel in July 2006. She received her Ph.D. and B.S. degrees from the Department of Computer Science and Technology, Tsinghua University, China, in 2006 and 2001, respectively. Her e-mail is patricia.p.wang at intel.com.

Wei Hu is a researcher in the Microprocessor Technology Lab, Beijing. His research interests include many areas of artificial intelligence, computer vision, and data mining. He is currently working on video-mining projects, such as commercial detection in TV programs and automatic speaker recognition in videos. Before joining Intel, he was a Research Associate for the Department of EEE at the University of Hong Kong (1998-1999). He received his Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences in 1998. His e-mail is wei.hu at intel.com.

Yimin Zhang is a researcher in the Microprocessor Technology Lab, Beijing. He leads a team of researchers working on various statistical computing techniques and their scalability analysis, recently focusing on media mining, data mining, etc. He joined Intel in 2000. At Intel, he has been involved in several projects related to natural language processing and speech recognition, especially focusing on Chinese-named entity extraction and DBN-based speech recognition. He received his B.A. degree from Fudan University in 1993, his M.S. degree from Shanghai Maritime University in 1996, and his Ph.D. degree from Shanghai Jiao Tong University in 1999, all in Computer Science. His e-mail is yimin.zhang at intel.com.

Yen-Kuang Chen is a Principal Engineer in the Microprocessor Technology Lab at Santa Clara, California. His research interests include developing innovative multimedia applications, studying the performance bottleneck in current architectures, and designing next-generation microprocessors/platforms. He has 10+ US patents, 25+ pending patent applications, and 75+ technical publications. He is one of the key contributors to Supplemental Streaming SIMD Extension 3 in Intel Core 2 processor family. He received his Ph.D. degree from Princeton University. His e-mail is yen-kuang.chen at intel.com.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2,

IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, IPLink, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel's trademarks may be used publicly with permission only from Intel. Fair use of Intel's trademarks in advertising and promotion of Intel products requires proper acknowledgement.

*Other names and brands may be claimed as the property of others.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Bluetooth is a trademark owned by its proprietor and used by Intel Corporation under license.

Intel Corporation uses the Palm OS[®] Ready mark under license from Palm, Inc.

Copyright © 2007 Intel Corporation. All rights reserved.

This publication was downloaded from
<http://www.intel.com>.

Additional legal notices at:
<http://www.intel.com/sites/corporate/tradmarx.htm>.

For further information visit:

developer.intel.com/technology/itj/index.htm