A Novel Parallel Encoding Framework for Scalable Video Coding

Kai Yao, *Jun Sun, Jiaying Liu, Zongming Guo
Institute of Computer Science and Technology
Peking University
Beijing, China
{yaokai, sunjun, liujiaying, guozongming}@icst.pku.edu.cn

Longshe Huo
Core Network Group
China Unicom Research Institute
Beijing, China
huols@chinaunicom.cn

***Abstract*—In this paper, we first propose a new parallel video coding framework, considering three important factors: parallel strategy, computational complexity and task scheduling. Then combining the characteristics of scalable video coding (SVC), a novel parallel encoding structure for temporal and quality scalabilities is introduced to obtain a high speedup of parallel SVC. Since the data dependencies in SVC are complex and time variant, the scheduling of parallel SVC is extremely difficult. In order to find the optimal scheduling solution, directed acyclic graph (DAG) is exploited to model the dependencies of encoding tasks, and the complexities of the encoding tasks which are accurately estimated by the Kalman filter to weight the scheduling tasks. Finally, two heuristic scheduling algorithms are also proposed to achieve a high encoding speed of parallel SVC. Experimental results show that the speedup of our parallel method was higher (about 60%) than previous work. Using the proposed method, high definition (HD) SVC videos can be encoded in real time.**

I. INTRODUCTION

Nowadays, scalable video coding (SVC) [1] has become one of the hottest research fields. It provides three different scalabilities and satisfies various requirements. Since SVC is constructed with several layers of H.264/AVC which has a high computational complexity, the computational complexity of SVC encoder is extremely high. Therefore, it is important to explore new methods to accelerate the encoding speed of SVC. A recent trend in improving the encoding speed is to employ parallel encoding by utilizing multi-core technology. However, the data dependencies of SVC are quite complex. It is difficult to explore the parallel encoding of SVC. The traditional SVC encoders, including SVC reference software [2], do not encode in parallel and cannot fully utilize the power of a multi-core system. In order to apply SVC in multi-core environments, we should explore new parallel method by utilizing the characteristics of SVC.

Parallel encoding for non-scalable video has been an active research topic for well over a decade. There are four main parallel granularities in parallel encoding: group of picture (GOP)-level, frame-level, slice-level and MB-level [3]. Although GOP-level and slice-level parallelism are both simple to implement, GOP-level parallelism causes long latency and slice-level parallelism significantly affects coding efficiency. On the contrary, frame-level parallelism has little encoding latency. However, its parallelism is limited by the dependencies among frames. High parallelism can be obtained by MB-level parallelism, but it is with great difficulty and workload of synchronization. Since the data dependencies of SVC are complex, these parallel methods cannot be used directly for parallel SVC. For SVC, Yang [4] proposed a frame-level parallel method utilizing the temporal scalability of SVC, but only considered the frame-level parallel processing in one GOP. A frame-level parallel framework for multi-view video coding (MVC) which considered the parallel among several GOPs was introduced in [5]. However, it is not entirely applicable to SVC, because the GOP structure in MVC is different from that in SVC and more characteristics of SVC can be exploited to increase the parallelism.

In this paper, a novel parallel encoding framework is proposed to achieve a high encoding speed. First, on the basis of Yang's method, more than one GOP can be encoded in parallel and higher parallelism is acquired. Then the key picture of the next GOP is able to be processed earlier to obtain higher parallelism by utilizing the medium grain quality scalability (MGS) of SVC. Since the data dependencies of parallel SVC are extremely complex, directed acyclic graph (DAG) is applied to model the data dependencies. Besides, the complexities of the encoding tasks are accurately estimated by the Kalman filter to weight the scheduling tasks in DAG. Based on DAG, two heuristic scheduling algorithms are also proposed to gain a high speedup. It is notable that rate control of our framework is well compatible with non-parallel structure through the thread synchronization.

The rest of this paper is organized as follows: Section II briefly introduces our parallel encoding framework. Section III proposes two parallel methods utilizing the temporal and quality scalabilities of SVC. The complexity estimation based on the Kalman filter is described in Section IV. Section V contains an analysis on the problem of scheduling parallel SVC using DAG. Experiments are presented in Section VI and conclusions are given in Section VII.

* is the correspondence author. This work was supported by National Natural Science Foundation of China under contract No. 61071082, National Basic Research Program of China under contract No. 2009CB32097 and Beijing Municipal Natural Science Foundation under contract No. 4102025

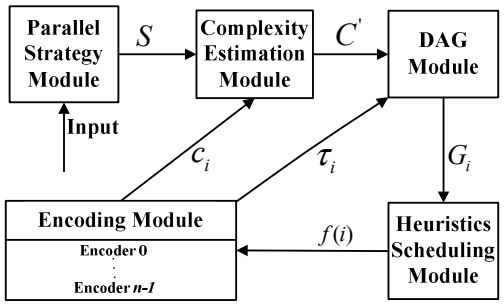

Fig. 1. The framework for parallel video coding

II. FRAMEWORK FOR PARALLEL VIDEO CODING

Fig. 1 shows our framework for parallel video coding. It is constituted by five main components: the parallel strategy module, the complexity estimation module, the DAG module, the heuristics scheduling module and the encoding module. In this framework, S denotes the parallel strategy, C' denotes the estimated complexities of the tasks, G_i denotes the DAG of the encoding tasks at the i-th scheduling process. $f(i)$ denotes the function which makes task scheduling decisions triggered by the heuristics scheduling module. τ_i represents the completed task. c_i is the actual complexity of τ_i.

First, the parallel strategy module chooses the appropriate parallel strategy S, such as frame-level parallel method or some combined parallel methods. Then according to the parallel strategy, the complexity estimation module estimates the complexities $C'$of the encoding tasks. With the parallel strategy and the estimated complexities of the encoding tasks, the DAG module builds G_i to model the data dependencies of the encoding tasks at the i-th scheduling process. Based on the DAG, heuristics scheduling module triggers the function $f(i)$ to choose a proper task to be processed. Finally, the encoding module encodes the task τ_i decided by $f(i)$. When τ_i finishes, a signal is sent to the DAG module to update the DAG and the actual complexity c_i is sent to the complexity estimation module to make the estimation more accurate.

III. PARALLEL ENCODING FOR SVC

A. Parallel Encoding for the Temporal Scalability

The parallel encoding method for SVC proposed in [4] is illustrated in Fig. 2(a). T_i is the temporal layer number of the picture at the i-th temporal layer in its GOP. Since the pictures at each temporal level in one GOP are all dyadic and independent to each other, they can be encoded in parallel. However, only the pictures in one GOP are processed in parallel in [4]. There are at most gop_size/2 frames that can be encoded in parallel, where gop_size is the size of hierarchical B-Picture structure GOP.

In order to obtain higher parallelism, we explore the parallelism among several GOPs. In the hierarchical B-picture prediction structure (shown as Fig. 2(b)), the P-picture of the next GOP just depends on the P-picture of the current GOP. After the P-picture of the current GOP has finished, the P-picture of the next GOP and the B-pictures in the current GOP are allowed to be encoded in parallel. Therefore, the pictures in several GOPs can be processed simultaneously under the hierarchical B-picture prediction structure.

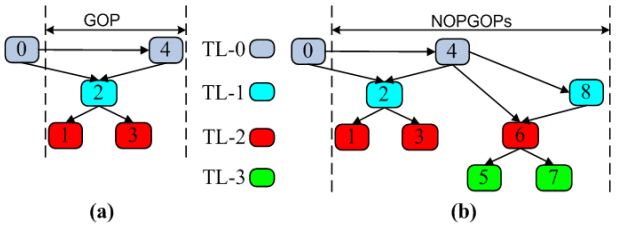

Fig. 2. Parallel SVC using temporal scalability: (a) Parallel in only one GOP, (b) Parallel in several GOPs (the size of GOP is 4 and NOPGOPs=2)

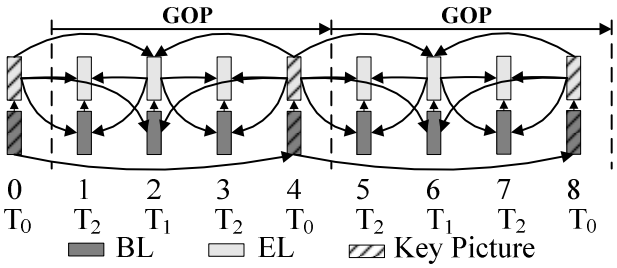

Fig. 3. MGS prediction structure of SVC

Based on the above discussion, we can modify the former parallel structure into a new structure to obtain higher parallelism (shown as Fig. 2(b)). The number of GOPs that can be encoded in parallel is called the number of parallel GOPs (NOPGOPs). For simplicity, we denote NOPGOPs as n and denote gop_size as λ. Since the prediction structure is dyadic, there are $\log_2(\lambda)+1$ temporal stages in one GOP. Furthermore, no more than $\log_2(\lambda)+1$ GOPs can be processed at the same temporal layer (this is easy to be obtained by mathematical induction). Since the numbers of the pictures in following GOPs are arranged as: $\lambda/2, \lambda/4, \ldots, 1, 1$ at the highest temporal layer of the first GOP ("the first GOP" is just used as an example), the maximum frame parallelism P can be derived as:

$$P = \begin{cases} \lambda \times \left(1 - 2^{-n}\right) & 1 \le n \le \log_2(\lambda) \\ \lambda & n = \log_2(\lambda) + 1 \end{cases} \quad (1)$$

B. Parallel Encoding for the Quality Scalability

In SVC, medium-grain quality scalability (MGS) is the most commonly used quality scalability. It uses the so-called key picture to allow adjustment of suitable tradeoff between drift and coding efficiency. Fig. 3 illustrates how the key picture concept can be efficiently combined with hierarchical B-picture prediction structure with two quality layers. The picture at the coarsest temporal layer is encoded as key picture which just refers to the base layer (BL) of the prior key picture. Besides, all B-pictures refer to the highest enhancement layer (EL) of their reference pictures.

Based on the characteristics of MGS and the temporal parallel method in Section III-A, a novel parallel encoding structure is proposed when encoding a SVC video with multiple quality enhancement layers. As illustrated in Fig. 4, each node τ_{ij} denotes the j-th layer of frame i. The most obvious feature of this parallel method in Fig. 4 is that the next key picture does not need to wait for the whole current key

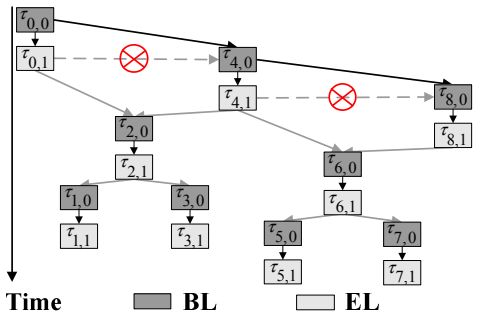

Fig. 4. Parallel encoding structure for MGS

picture to be encoded completely. Because the next key picture just depends on the base layer of the current key picture. When the base layer of the current key picture is completed, the next key picture can start to be encoded. For example, task $\tau_{4,0}$ can start to be encoded when the base layer task $\tau_{0,0}$ has been completed (as the left ⊗ shows in Fig. 4). There are two main advantages which can be obtained by this approach: 1) The next P-picture can start to be processed earlier, and as a result of that, more pictures are encoded in parallel. 2) Similarly, more B-pictures of the next GOP can start to be encoded earlier. Therefore, higher parallelism can be achieved under this parallel encoding structure.

IV. COMPLEXITY ESTIMATION USING KALMAN FILTER

In this work, Kalman filter [6] is applied to obtain the optimal estimate of the complexities of the encoding tasks. Total execution time is used as the criterion of the computational complexity. We suppose that the complexities of the frames in the same hierarchal position of two neighboring GOPs are changeless and the difference is caused by zero mean Gaussian white noise. Fig. 5(a) analyses the difference of the key picture's complexity in a sequence of GOPs (the frames at other positions also have the similar results). It can be seen that the difference is in concordance with the Gaussian distribution.

We set GOP as the estimation unit. The complexity of one GOP is denoted as $\mathbf{C}_k = [c_{k,1}, c_{k,2}, \dots, c_{k,N}]^T$, k is the number of GOP and N is the size of GOP, $c_{k,i}$ is the complexity of the i-th frame in the k-th GOP. In our Kalman filter algorithm, the estimate equations and measurement equations for the complexity the k-th GOP are expressed by

$$\mathbf{C}_k = \mathbf{\Phi}_k \mathbf{C}_{k-1} + \mathbf{w}_k \quad (2)$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{C}_k + \mathbf{e}_k \quad (3)$$

where $\mathbf{C}_k$ is the state vector of the k-th GOP's complexity, and evolved from the previous state by state-transition matrix $\mathbf{\Phi}_k$. $\mathbf{z}_k$ is the measurement vector of $\mathbf{C}_k$ and obtained by measurement matrix $\mathbf{H}_k$. $\mathbf{w}_k$ and $\mathbf{e}_k$ are assumed to be zero mean Gaussian distributions with covariances $\mathbf{Q}(k)$, and $\mathbf{R}(k)$ respectively. The state-transition matrix $\mathbf{\Phi}_k$ is an identity matrix on the basis of our supposition and the measurement matrix $\mathbf{H}_k$ can be estimated by the least square method. The error covariance matrix of the state estimate $\mathbf{C}_k$ is denoted as $\mathbf{P}_k$. The Kalman filter works as following five steps:

1) Estimate complexity: $\hat{\mathbf{C}}_{k|k-1} = \mathbf{\Phi}_k \hat{\mathbf{C}}_{k-1|k-1}$

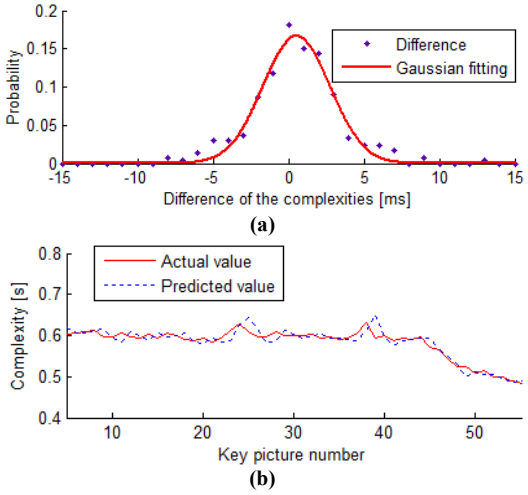

Fig. 5. Complexity Estimation: (a) Complexity difference distribution of the key picture. (b) Complexity estimation result of the key picture.

2) Get estimate covariance: $\mathbf{P}_{k|k-1} = \mathbf{\Phi}_k \mathbf{P}_{k-1|k-1} \mathbf{\Phi}_k^T + \mathbf{Q}_k$

3) Obtain Kalman gain: $\mathbf{G}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T [\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k]^{-1}$

4) Update the estimate: $\hat{\mathbf{C}}_{k|k} = \hat{\mathbf{C}}_{k|k-1} + \mathbf{G}_k [\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{C}}_{k|k-1}]$

This is the final estimate output.

5) Obtain Updating-error covariance: $\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{G}_k \mathbf{H}_k] \mathbf{P}_{k|k-1}$

Fig. 5(b) shows the complexity estimation result of the key picture. It can be seen that the complexity estimation result commendably reflects the trend of the complexity. With the constantly updated estimate and covariance, the complexity estimate is more and more accurate.

V. HEURISTIC SCHEDULING OF SVC USING DAG

A common fundamental tool named DAG [7] is exploited to model the data dependencies of parallel SVC (other scheduling model may be also suitable). Fig. 6 shows the DAG modeling of the data dependencies of the first GOP in Fig. 4. The nodes designate the tasks $\tau_{ij} \in \Gamma, 0 \le i \le n-1, 0 \le j \le l-1$, where Γ is the set of tasks, i is the frame number and j is the quality layer number. w_{ij} is the execution time of task τ_{ij}, which is used to weight the task node. The edges in the DAG are denoted as $e_i = \langle \tau_{jm}, \tau_{kn} \rangle, e_i \in E, i = 0, 1, \dots, |E| - 1$ and $\tau_{jm}, \tau_{kn} \in \Gamma$. The edges are weighted by the communication costs of the tasks and we assume that the cost of communication is negligible in this framework. According to the DAG of parallel SVC, the encoding order is restrained by the data dependencies and the temporal layer numbers: 1) If an edge exists from τ_{jm} to τ_{kn}, it means that task τ_{kn} depends on task τ_{jm}, and τ_{kn} cannot start until τ_{jm} finishes. 2) The frames at the lower temporal layer are always encoded before the frames at the higher temporal layer.

Two heuristic scheduling algorithms are designed as: Display Order (DO) and Computational Complexity (CC). 1) DO heuristic scheduling algorithm means give the highest priority to the earliest frame in display order. Since the frames at the lower temporal layer must be encoded before the frames at the higher temporal layer, the encoding order of the frames

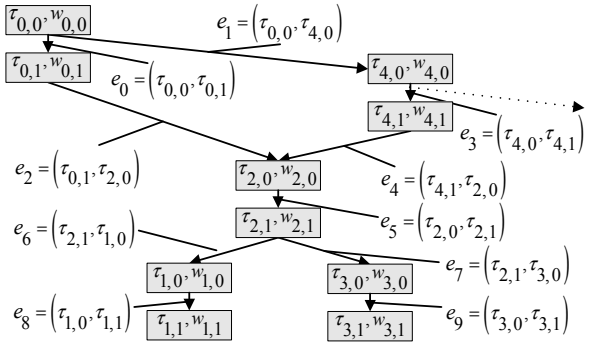

Fig. 6. DAG of MGS-based SVC

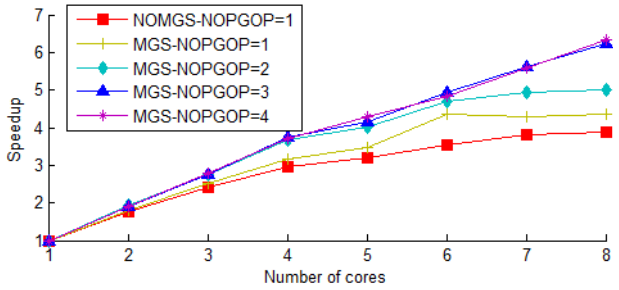

Fig. 7. Speedup of Yang's and our methods

is 0, 4, 2, 8, 1, 3, 6, 5, 7 (take Fig. 2(b) as an example). 2) CC heuristic scheduling algorithm means to give the highest priority to the frame which has the lowest computational complexity. The encoding order is also restrained by the data dependencies and the temporal layer. Besides, the encoding order depends on the computational complexity which is obtained by the Kalman filter.

VI. EXPERIMENTAL RESULTS

The parallel framework of SVC was implemented on the Intel Xeon multi-core processor E5520 at 2.27 GHz with 8 logic cores to encode high definition (HD) videos. We set the base layer QP to 32, the number of quality layer to 3, the delta QP to 6, and the size of GOP to 8. All the resolutions of the standard test sequences are 720p (1280 × 720).

The speedups of our method and Yang's method are shown in Fig. 7. The speedup of our method is similar to Yang's method when the number of cores is less than 4, because 4 pictures at most can be encoded in parallel in Yang's method. Since quality scalability is utilized in our method, a higher speedup (about 13%) is obtained when considering the parallel in only one GOP (NOPGOPs=1). When increasing NOPGOPs, more GOPs can be encoded in parallel. Therefore, a higher speedup (about 60%) can be achieved. The test sequence is "shields" (504 frames).

The speedups of the different types of heuristics scheduling algorithms are illustrated in Table I when NOPGOPs equals to 3 and 4 respectively. By reordering the encoding sequence, a higher speedup is gained. By utilizing complexity estimation, CC has better performance than DO.

It is clear from Fig. 8 that the encoder can competently achieve real-time SVC HD video coding and it is suitable for

TABLE I. SPEEDUP OF DIFFERENT SCHEDULING ALGORITHMS

Sequence	Scheduling algorithms				
	Yang's	NOPGOPs=3		NOPGOPs=4	
		DO	CC	DO	CC
parkrun	4.37	6.32	6.51	6.71	6.72
mobcal	4.18	6.03	6.28	6.55	6.51
shield	4.09	5.91	6.35	6.57	6.42
stockholm	4.15	5.97	6.51	6.74	6.77

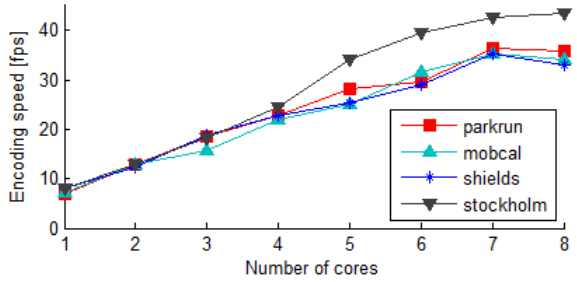

Fig. 8. Encoding speed of our framework (NOPGOPs=2)

various standard test sequences including "parkrun", "mobcal", "shields" (all are 504 frames) and "stockholm" (604 frames). When using more than 6 cores, SVC HD video can be encoded in real-time.

VII. CONCLUSION

In this paper, a novel parallel encoding framework for SVC is introduced. Under this framework, we propose a parallel encoding structure utilizing the temporal and quality scalabilities of SVC. Since the data dependencies of parallel SVC are complex and time variant, DAG is exploited to model the data dependencies of parallel SVC and the complexities of the encoding tasks are estimated by the Kalman filter. Besides, we designed two heuristics scheduling algorithms which were generic and scalable. The encoder can realize real-time SVC HD video coding with our framework.

REFERENCES

[1] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103-1120, Sep. 2007.

[2] Joint Scalable Video Model(JSVM), JSVM Software, Joint Video Team, Doc. JVT-X203, Geneva, Switzerland, Jun. 2007.

[3] Y. Chen, Q. L. Eric, X. Zhou, and G. Steven, "Implementation of H.264 Encoder and Decoder on Personal Computers," Journal of Visual Communication and Image Representation. vol. 17, no. 2, pp. 509-532, Apr. 2006.

[4] S. Yang, S. Wang, H. Chen, J. Wu, "A Parallelism Encoding Framework for the Temporal Scalability of H.264/AVC Scalable Extension," Multimedia Workshops, Ninth IEEE International Symposium on, 2007.

[5] Y. Pang, J. Wen, L. Sun, W. Hu, S. Yang, "Frame-level Heuristic Scheduling Multi-view Video Coding on Symmetric Multi-core Architecture," in Proc. ICIP, pp.2305-2308, 2009.

[6] M. S. Grewal and A. P. Andrews, Kalman Filtering Theory and Practice. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[7] R. Bajaj, D. P. Agrawal, "Improving Scheduling of Tasks in a Heterogeneous Environment," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 2, pp. 107-118, Feb. 2004