Dynamic MB-level Scheduling for Parallel Video Coding

*Shengfu Dong[1,2], Zhenyu Wang[1,2], Ronggang Wang[*1,2], Wenmin Wang[1,2], Wen Gao[1,2]*

[1]Peking University Shenzhen Graduate School, [2]Peking University National Engineering Laboratory for Video Technology
Shenzhen, Guangdong, China

{dongsf, wangzhenyu, [*]rgwang, wangwm}@pkusz.edu.cn, wgao@pku.edu.cn

***Abstract*—MB-level parallelism is widely used in parallel video coding thanks to its merits of low latency, no performance loss and high degree of parallelism. Most of video encoders with MB-level parallelism employ MB Row Scheduling (MRS) scheme. In software video encoder, early terminate algorithms tend to cause significant difference in coding time of different MBs. Consequently, the running speeds of multiple threads are unbalanced. When the number of threads is more than that of physical cores, the running speed unbalance is further worsened by computation resources competition among multiple threads. The computation resources of multiple cores can't be fully utilized without careful handling of the above running speed unbalance. Additionally, synchronization of multiple threads can also penalize the running speed of the whole video encoder. In this paper, we analyze the running speed unbalance of multiple threads in MRS scheme, and propose a new Dynamic MB-level Scheduling (DMS) scheme for parallel video coding. DMS alleviates both the running speed unbalance and synchronization delay among multiple threads on multi-core platform. Experiment results verified that video encoder with MRS can be accelerated in average 9% by our proposed DMS, when processors are fully utilized.**

Index Terms – Video coding, MB-level parallelism, parallel scheduling, speed unbalance, thread competition

I. INTRODUCTION

Real-time video coding has always been a hot topic since its heavy computational load for processors. Among various speed-up technologies, parallel video coding is one of the most efficient solutions to realize a real-time video encoder on multi-core platforms.

Data parallelism is widely exploited in software video encoders for its scalability. In summary, there are four typical levels of data parallelism, including group of picture (GOP), frame, slice and macro-block (MB). A GOP-level and a slice-level parallel encoder for H.26L are presented in [1]. For these two parallelisms, a sequence or a frame is decomposed into multiple GOPs or slices, which can be encoded in parallel. A frame-level parallelism is proposed in [2], by which continuous B frames and their successive I or P frames can be encoded in parallel. The concurrency among different MBs was analyzed, and a Wavefront MB-level parallelism is proposed in [3]. Additionally, there are also some other parallelism schemes, such as MB-region parallelism [4], and MB-Group parallelism [5][6], etc. Moreover, tasks parallelism can also be used in software encoder. Video encoder is decomposed motion estimation and other modules, and a task-data hybrid parallel scheme was proposed in [7].

Compared with other typical data parallelisms, MB-level parallelism is widely utilized own to its advantages of low latency, no performance loss and high degree of parallelism [3]. The recently issued video coding standard HEVC [8] also introduced MB-level parallelism called Wavefront [9]. To our knowledge, most of MB-level parallel encoders employ MB Row Scheduling (MRS) scheme. In our previous work, we also realized a two-stage parallel encoder based on MRS [10].

In this paper, we firstly analyze the running speed unbalance of MRS, and then we propose a new Dynamic MB-level Scheduling (DMS) scheme to alleviate both the unbalance and synchronization delay among multiple threads. Experiment results testified the effectiveness of our proposed scheme.

The rest of this paper is organized as follows. Section II gives an analysis on running speed unbalance of MRS scheme. Section III presents our proposed DMS scheme and our implementation in video encoder in detail. Section IV shows the experimental results. At last, this paper is concluded in section V.

II. ANALYSIS ON RUNNING SPEED UNBALANCE OF MRS

As mentioned above, MB-level parallelism is widely used in software video encoder. In modern video coding standard, such as H.264/AVC [11], and AVS [12] etc., data dependences exit between current MB and its left, up, up-left or up-right neighbors [3]. The cost of data exchange and synchronization can't be neglected. To alleviate the above problems, most of MB-level parallel video encoders employ MRS scheme [10].

However, in a software video encoder, many early terminate algorithms are used, such as fast motion estimation and fast mode decision, etc. These algorithms cause significant difference in coding time of different MBs and cause the

unbalance of running speeds in different threads. Additionally, threads are managed by operation system, which switches all the threads in system if necessary, including the threads not belong to the video encoder. When the number of threads is more than that of physical cores, the running speed unbalance is further worsened by computation resources competition among multiple threads.

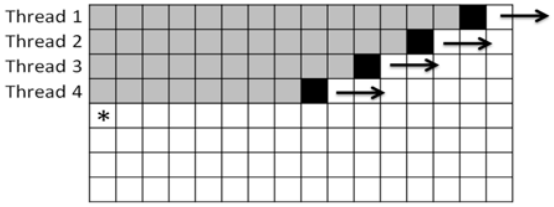


Figure 1. MB row based distribution

Fig. 1 shows the MRS scheme with four worker threads. In this scheme, each thread is assigned to encode one available MB row. After the thread 1 finished the first MB row, the fifth MB row can start to be encoded. If a thread A assigned on MB row A′ runs faster than the other thread B assigned on MB row above A′, the thread A has to wait. As shown in Fig. 1, though the MB marked by "*" is available to be encoded immediately, it still can't be chosen by the above mentioned waiting thread A. Thus, the computation resources of multiple cores can't be fully utilized in MRS scheme.

To verify our observation, we run our designed video encoder with MRS scheme using 4 worker threads on i7-2600K processor with 8 logic cores. The wait-lock status is captured by Intel VTune [13]. Figure 2 shows the waiting status when encoding one frame. Each bar denotes a thread, top three are control threads and the others are worker threads. Deep color in each bar indicates running state, light color indicates sleeping (waiting) state, and the lines between different bars indicate wake-up operations. As can be seen, there are frequent waitings between the threads encoding adjacent MB rows.

Figure 2. Wait-lock captured by VTune [13]

From the above analysis and experiment, the performance of MRS scheme can still be improved.

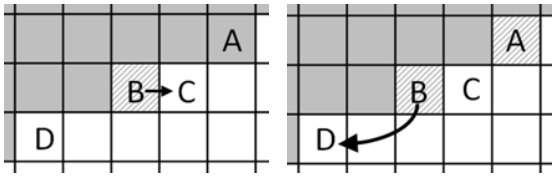
III. DYNAMIC MB-LEVEL SCHEDULING (DMS)

As analyzed in last section, we can reduce the waitings of MRS by assigning the encoding task MB by MB, other than MB row by MB row. But unfortunately, in a software realization, consistency between different threads can only be protected by mutex. If a worker thread wants to encode one MB, it has to lock a mutex and check whether the MB has been assigned to other worker threads. The encoding task has to be assigned serially, when more than one thread are waiting. This cost of synchronization waiting delay can't be neglected.

In this section, we propose an efficient Dynamic MB-level Scheduling (DMS) scheme, to alleviate both the running speed unbalance and the cost of synchronization.

A. Dynamic MB-level scheduling (DMS) Scheme

In DMS, the basic scheduling unit is a MB and each thread can choose any available MB to encode it. After a thread finished one MB, the MB on the right (if it is available) has the highest priority than any other available MBs. As shown in Fig. 3(a), if a thread finished MB *B*, and MB *A* has been encoded, this thread will encode MB *C* directly. In contrast, if MB *A* is encoding, this thread will encode another available MB, for example, MB *D* shown in Fig. 3(b).



(a) (b)

Figure 3. MB loads distribution of DMS scheme

To describe this scheme clearly, we define $\boldsymbol{S}$ as the set of all the MBs in current frame, $\boldsymbol{F}$ as the set of encoded MBs, and $\boldsymbol{A}$ as the set of available MBs. We also define $\boldsymbol{MB_{x,y}}$ as the MB with the index (x,y) in one frame. At the beginning of encoding one frame, $\boldsymbol{F}$ is empty, $\boldsymbol{MB_{0,0}}$ is in set $\boldsymbol{A}$. The process of a worker thread can be described as follows.

DMS: Dynamic MB-level scheduling
FUNCTION Worker_Thread_Encode_Frame ( )
  $\mathbf{x} = -\mathbf{1}$;
  WHILE ($\boldsymbol{F} \neq \boldsymbol{S}$)
    IF ($\boldsymbol{x} \neq -1$ & $\boldsymbol{MB_{x+1,y}} \in \boldsymbol{S}$ & ($\boldsymbol{MB_{x+2,y-1}} \in \boldsymbol{F} \mid \boldsymbol{MB_{x+2,y-1}} \notin \boldsymbol{S}$))
      x = x + 1;
    ELSE
      Lock mutex;
      Get $\boldsymbol{MB_{i,j}}$ from set $\boldsymbol{A}$ by ***TFD***;
      $\boldsymbol{A} = \boldsymbol{A} - \{\boldsymbol{MB_{x,y}}\}$;
      Unlock mutex;
      x = i; y = j;
    END IF
    Encode $\boldsymbol{MB_{x,y}}$;
    $\boldsymbol{F} = \boldsymbol{F} \cup \{\boldsymbol{MB_{x,y}}\}$;
    IF ($\boldsymbol{MB_{x-1,y+1}} \in \boldsymbol{S}$ & ($\boldsymbol{MB_{x-2,y+1}} \in \boldsymbol{F} \mid \boldsymbol{MB_{x-2,y+1}} \notin \boldsymbol{S}$))
      $\boldsymbol{A} = \boldsymbol{A} \cup \{\boldsymbol{MB_{x-1,y+1}}\}$;
    END IF
  END WHILE

Here, ***TFD*** is a "top-first" function, which always chooses the topmost MB in set ***A***. It can be described as

$$\mathrm{TFD}(\boldsymbol{A}) = MB_{x,y} \mid \forall MB_{i,j} \in \boldsymbol{A}, j \geq y. \quad (1)$$

As can be seen, after a thread finished a $MB_{x,y}$, it will encode the $MB_{x+1,y}$ directly if it is available. In this case, no lock operation is demanded. Since the $MB_{x+1,y}$ will never be added to set ***A***, any other threads don't have a chance to encode $MB_{x+1,y}$. Obviously, this scheme can keep the consistency and reduce the cost of lock operations.

B. Implement of Video Encoder with DMS

In this section, we will describe the realization of our proposed DMS scheme in video encoder.

We chose xavs platform [14], which is an optimized encoder of AVS1-P2 (Jizhun Profile) [12]. There are some early terminate algorithms, such as fast motion estimation and fast mode decision. Additionally, xavs use a cache structure to store all the data required during encoding one MB. To get a real-time HD video encoder, we optimized xavs with SIMD (Single Instruction Multiple Data) technology.

Fig. 4 shows the structure of our designed encoder with DMS. In order to reduce the influence of serial I/O, independent threads for input and output are used. A frame-level control thread is used to connect I/O threads and worker threads. In this structure, worker threads are managed with symmetrical mode. MB loads can be chosen by worker threads themselves. For communication, shared memory is used to store information of encoded MBs. The shared data is loaded into cache before encoding a MB and updated from cache after finishing the MB.

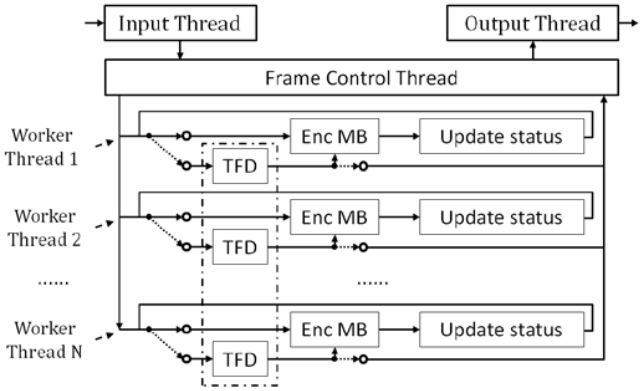


Figure 4. Structure of DMS based encoder

As shown in Fig. 4, each worker thread works as described in last section. When a thread finished a MB, it can encode the MB on the right directly without any lock operations. Only if the MB on the right is unavailable, the thread has to enter into the lock area and perform a TFD process. During TFD process, a thread can also check whether all the MBs have been encoded, and decide to endthe encoding process.

IV. EXPERIMENTS

In this section, experimental results of MRS scheme and DMS scheme are presented to demonstrate the advantages of our proposed DMS scheme. We realized a DMS based video encoder as described in above section, and also a MRS based video encoder. We tested 6 typical 1080P sequences and recorded the encoding speed in configurations of different number of worker threads. Dell R610 server is chosen as testing platform, there are two X5660 CPUs, and each CPU has 6 physical cores. Test conditions are set as follows. GOP structure is set as "IBBP…", there are 25 frames in a GOP;2 reference frames are allowed for P frames, and the QPs of I, P, B frames are set as 33, 35, and 37 respectively.

In the first experiment, we close hyper-threading technology and set worker threads number from 1 to 12. As shown in Fig. 5, video encoder with DMS scheme is faster than video encoder with MRS scheme in all configurations of worker thread number except the configuration of 1 worker thread. And the speed-up of DMS over MRS is increased along with the number of worker thread increases.

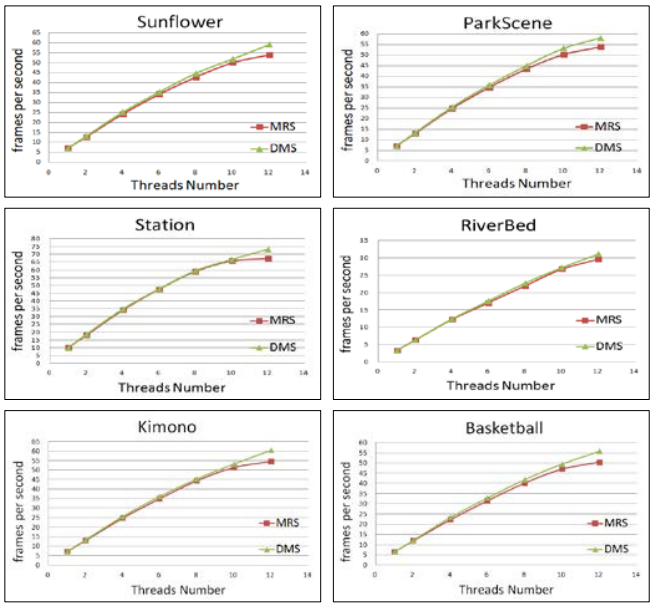


Figure 5. Encoded frames per second with MRS and DMS scheme

Table 1 lists the speed-up of our DMS based encoder compared with MRS based encoder. The speed-up ratio is calculated by

$$\text{speed\_up} = \frac{\mathrm{FPS}_{DMS}}{\mathrm{FPS}_{MRS}}. \quad (2)$$

FPS_{DMS} is encoded frames number per second of DMS based encoder, and $\mathrm{FPS}_{\mathrm{MRS}}$ is encoded frames number per second of MRS based encoder. As shown in Table 1, the speed-up ratio of DMS over MRS is from 2% to 4% when worker threads number is set from 2 to 10. The speed-up ratio increases to 9%

in configuration of 12 worker threads. With the increasing of worker thread number, the running speed unbalance becomes more and more serious. When the worker thread number is set the same as physical cores, competition of multiple threads causes the unbalance further worse. Thus, DMS can get a largest speed-up in configuration of 12 worker threads.

Table 1. Speed of MRS and DMS

Threads	1	2	4	6	8	10	12
Sunflower	0.99	1.03	1.04	1.03	1.04	1.04	1.09
ParkScene	0.99	1.03	1.03	1.03	1.04	1.06	1.08
Station	0.99	1.03	1.02	1.00	1.01	1.01	1.09
Riverbed	0.99	1.01	1.01	1.03	1.04	1.02	1.05
Kimono	1.00	1.03	1.03	1.03	1.02	1.04	1.11
Basketball	0.99	1.02	1.05	1.04	1.04	1.05	1.11
(Average)	0.99	1.02	1.03	1.03	1.03	1.04	1.09

In the second experiment, we open hyper-threading technology to compare MRS and DMS scheme. An interesting result is got from this experiment. As shown in Fig. 6, the encoding speed of DMS based encoder increases along with the increasing of worker thread number. But the encoder speed increases slower, when the worker threads number is bigger than 12. On the other hand, the encoding speed of MRS based encoder decreases suddenly when the worker threads number is set to 12, and then increases along with increasing of worker thread number. According to our knowledge, hyper-threading technology virtually builds two logic cores on every physical core. This technology can hide the memory delay and exploit the computing capability efficiently [2]. However, the two logic cores on a physical core can't provide two times performance than one physical core. Thus, if the number of worker threads equal to or more than the number of physical cores, some worker threads must be slower than others, that will cause serious running speed unbalance. According to the experiment results, our proposed DMS scheme can solve this problem, but the performance of MRS scheme even can be decreased by this kind of computation resources unbalance.

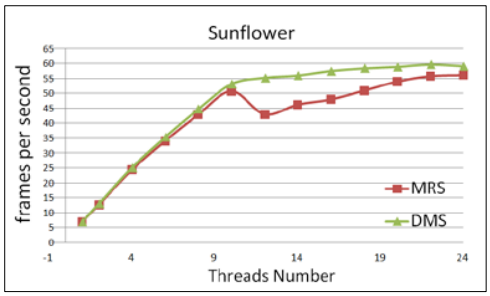


Figure 6. Speed of MRS and DMS on hyper-threading

V. CONCLUSION

In this paper, we proposed a dynamic MB-level scheduling (DMS) scheme, and realized a DMS based video encoder. This scheme alleviates both running speed unbalances and the cost of multiple threads synchronization. Experiment results testified that DMS based video encoder can get a speed-up of 9% over MRS based video encoder when the processor is fully used. When hyper-threading is opened, the speed-up of DMS over MRS is even larger.

ACKNOWLEDGEMENT

This work was partly supported by the grant of Key Projects in the National Science & Technology Pillar Program 2011BAH08B03, and Shenzhen Basic Research Program of JC201104210117A, JC201105170732A, and JCYJ2012061450301623.

REFERENCES

[1] J. C. Fernández and M. P. Malumbres. "A parallel implementation of H. 26L video encoder." Euro-Par 2002 Parallel Processing. Springer Berlin Heidelberg, 2002. 830-833.

[2] Y. K. Chen, X. Tian, S. Ge and M. Girkar. "Towards efficient multi-level threading of H. 264 encoder on Intel hyper-threading architectures." Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. IEEE, 2004.

[3] Z. Zhao and P. Liang. "A highly efficient parallel algorithm for H. 264 video encoder." Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on. Vol. 5. IEEE, 2006.

[4] S. W. Sun, D. Wang and S. Chen. "A highly efficient parallel algorithm for H. 264 encoder based on macro-block region partition." High Performance Computing and Communications. Springer Berlin Heidelberg, 2007. 577-585.

[5] R. Sachdeva and K. Saha. "Parallel MPEG-2 Video Encoder."Consumer Electronics (ICCE), 2011 IEEE International Conference on. IEEE, 2011.

[6] Z. Y. Wang L. H. Liang, G. L. Yang, X. G. Zhang, J. Sun, D. B. Zhao and W. Gao. "A novel macro-block group based AVS coding scheme for many-core processor." Journal of Signal Processing Systems 65.1 (2011): 129-145.

[7] P. Li, B. Veeravalli and A. A. Kassim. "Design and implementation of parallel video encoding strategies using divisible load analysis." Circuits and Systems for Video Technology, IEEE Transactions on 15.9 (2005): 1098-1112.

[8] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Trans. Circuits and Systems for Video Technology, Vol. 22, No. 12, pp. 1649-1668, Dec. 2012.

[9] C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux and T. Schierl. "Parallel scalability and efficiency of HEVC parallelization approaches." Circuits and Systems for Video Technology, IEEE Transactions on 15.9 (2012): 1-1.

[10] S. F. Dong, Z. H. Wang, R. G. Wang, Z. Y. Wang and W. Gao. "A two stage parallel encoder scheme for real time video encoder." Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on. IEEE, 2012.

[11] ISO/IEC 14496-10 and ITU-T Rec, "H.264, Advanced video coding for generic audiovisual services," ed, 2003.

[12] W. Gao, S. W. Ma, L. Zhang, L. Su and D. B. Zhao. "AVS video coding standard." Intelligent Multimedia Communication: Techniques and Applications. Springer Berlin Heidelberg, 2010. 125-166.

[13] Intel® VTune™ Amplifier. http://software.intel.com/en-us/intel-vtune-amplifier

[14] A Free AVS Encoder. http://xavs.sourceforge.net