Encoding H.264 by Thread Level Parallelism

Under the guidance of Dr. K.R Rao

Submitted by:
Shaheer Ahmed
ID: 1000836047
MSEE, University of Texas at Arlington
Objective:
To perform thread level parallelism to increase the performance of a H.264 encoder.

Introduction:
H.264 is an industry standard for video compression, the process of converting digital video into a format that takes up less capacity when it is stored or transmitted. Video compression (or video coding) is an essential technology for applications such as digital television, DVD-Video, mobile TV, videoconferencing and internet video streaming [1]. In order to meet the needs of the industry H.264 was developed.

The H.264/AVC standard was first published in 2003. It builds on the concepts of earlier standards such as MPEG-2 and MPEG-4 Visual and offers the potential for better compression efficiency (i.e. better-quality compressed video) and greater flexibility in compressing, transmitting and storing video [1]. It is based on modules of block motion compensation, transform, quantization and entropy coding. The H.264 baseline encoder is estimated to be 5x to 10x more complex than the H.263 encoder [2] while the decoder is 2x to 2.5x more than H.263 baseline decoder [3].

H.264 comprising of the following profiles
1. Baseline
2. Extended
3. Main
4. High

Fig.1: H.264 Profiles [4]

- Baseline Profile is used for real-time conversational services such as video conferencing and video calling. [4][11]

- Extended profile is for the multimedia services over the internet. [4][11]

- Main profile is designed for digital storage media and television broadcasting. [4][11]

- High profile is used in the fidelity range extensions for applications such as content contribution, content distribution and studio editing. [5]
The H.264 comprises of both encoder and decoder i.e. it is a codec. The encoder compresses the video into .264 formats and the decoder converts its back to the uncompressed format. This decoder is lossy.

Fig. 2 H.264 Encoder [6]

Fig. 3 H.264 Decoder [6]
Prediction Modes:
Prediction exploits the spatial or the temporal redundancy of a video sequence so that only the difference between actual and predicted image is encoded instead of encoding the entire image. There are two types of prediction
1. Intra Prediction
2. Inter Prediction

- Intra Prediction is done on the I frame. Since there exists a high similarity among neighboring blocks in a video frame, a block can be predicted and reconstructed using the neighboring pixels [7]

- Inter Prediction is done on the P frame and B frame. Since the frame rate of a video sequence is 30 to 60 fps (Nominal), the probability of having two successive frames to be similar is very high. The goal of inter prediction is to utilize this temporal redundancy and reduce the data to be encoded. [7]

Need for Parallel programming to implement H.264

Gordon Moore had stated that the transistors available on a semiconductor would double approximately every 18 to 24 months. This law has guided the computer designer for the past 40 years, but many people mistakenly think of Moore’s law as a predictor of CPU frequency which is not true. Over the years the CPU frequency has tended to follow Moore’s law but since past few years the CPU clock speeds have flattened out. This is because the higher speeds cause excessive power consumption, heat dissipation and current leakage. Thus in order to make the most efficient use of the processor resources, computer architects have developed ways to implement many instructions in parallel during the same clock cycle by either using multi cores or instruction level parallelism. [8]

Although the H.264 encoder is faster than previous versions, it is still not fast enough to meet real time video processing. Now that the encoder cannot rely on CPU clock speeds to improve the encoding time parallelization of the encoding is required at the software level. This can be done by exploiting thread-level parallelism at different levels.

- Data Domain Decomposition: H.264 encoder treats a video sequence as many group of pictures (GOP). Each GOP includes a number of frames. Each frame is divided into slices. Each slice is an encoding unit and is independent of other slices in the same frame. The slice can be further decomposed into a macro block(MB), which is the unit of motion estimation and entropy coding. All these are places to parallelize the encoder [9]
**Fig.4 Hierarchy of Data Domain Decomposition [9]**

**-Frame Level Decomposition:**
At frame-level, the input video stream is divided in GOPs. Since GOPs are usually made independent from each other, it is possible to develop a parallel architecture where a controller is in charge of distributing the GOPs among the available cores. [12]

The number of frames that can be coded in parallel is determined by the sequence of frames types in the video. A typical sequence of frames is I1 B2 B3 P4 B5 B6 B7 B8 B9 P10 , (where the subscript of the frame type indicates the frame's serial order). In this sequence, only three frames can be processed concurrently, with the following order of processing: \{I1\} ->P4 -> B2,B3,P7 ->\{B5,B6,P10\}. [10]

One more strategy that can be adopted for encoding the frames in parallel is as follows
1. Separate the total number of frames to encode into 2 equal sets.
2. Perform the parallel intra coding on two frames in both partitions.
3. Perform inter coding on frame 2 and frame 17 by incorporating changes in the encoding algorithm using Open MP. Repeat for frame 3 and frame 18 and so on till all the frames are encoded, as given in the figure 9.

**Fig.5 Parallel Processing of frames to reduce encoding time [11]**

**-Slice Level Decomposition:**
Partitioning of a frame to multiple independent slices enables parallel processing of slices. However, slicing the image and compressing each slice independently reduce the amount of spatial redundancy that can be exploited. Therefore, the more slices in the video, the higher the bit rate of the compressed video, assuming a desirable fixed quality of the compressed video. If the bit rate is kept fixed and the allowed degradation in the compressed video quality is less than
0.3db (in terms of signal-to-noise ratio), then each picture can be divided into 4-8 slices only, resulting in a limited number of threads. [10]

![Diagram](image)

Fig.6 Simultaneous Slice level and Macro Block level Parallelism [12]

**Conclusion:**
Parallel Programming in baseline profile will be studied on various test subjects and the performance boost will be measured. Slice level and Macro Block level Parallelism will be exploited.

**REFERENCES**


[8] Shameem Akhter and Jason Roberts, "Multi Core Programming, Increasing Performance through Software Multi Threading"
