PARALLEL COMPUTER ARCHITECTURE AND PROGRAMMING
15-618

Parallel Video Compression using the MPEG-1 algorithm

1 URL

`https://spenceryu.github.io/15618-final-project/`

2 Summary

We are aiming to parallelize the video compression mechanism described in the MPEG-1 standard. Audio compression will not be part of the scope of our project because it is far more heavily a signals processing project rather than an exploration of parallelization.

3 Background

We chose the MPEG-1 algorithm over newer standards such as MPEG-4 due to the fact that patents claiming ownership for MPEG-1 have expired. The MPEG-1 algorithm is a lossy compression algorithm, meaning data is lost in the compression process. A color space is the system in which tuples are mapped to specific colors. Discrete Cosine Transforms (DCT) are the standard transform used in image and video compression algorithms due to their ability to encode information about signals in a more space efficient manner than using sine functions to transform the signal. Huffman encoding is a special case of entropy encoding, which compresses repeating values. Several types of frames are used in the video compression process: I-Frames (the image at a frame), P-Frames (Forward deltas), and B-Frames (bidirectional deltas).

4 The Challenge

1. The algorithm consists of several sequential steps, which will require the use of synchronization for communicating intermediate values and progress.

2. Computing P-Frames and B-Frames has a temporal dependency, since these frames require information from previous or subsequent frames of the video. In order to efficiently parallelize this process, there are multiple possible options with tradeoffs to investigate. An example would be calculating deltas between the actual image frames (I-Frames) and then creating deltas using those results, versus calculating the deltas sequentially from split I-frame locations.

3. Entropy coding will require a tradeoff between the speed of the coding process and the compression that can be achieved. Trying to increase compression will require a Huffman tree covering more of the image, which will require communication between blocks during the creation of Huffman trees. However, if the entire Huffman tree for the image is calculated on each individual processor, there is no communication overhead, which might be faster.

5 Resources

The algorithm is well described in a mathematical sense but we have to implement it without any other baseline code.

Useful links:
`https://en.wikipedia.org/wiki/MPEG-1#Part_2:_Video`
`https://mpeg.chiariglione.org/standards/mpeg-1`
`https://www.researchgate.net/figure/Block-diagram-of-MPEG-1-algorithm-encoder_fig8_2985381`

6 Goals and Deliverables

1. Baseline sequential algorithm for MPEG-1 video compression.
2. Parallelization of preprocessing steps (color space, etc).
3. Parallelization of entropy encoding.
4. Analysis comparing our sequential vs. parallelized solution. How does the architecture we chose affect our performance and any inherent limitations?

7 Platform Choice

Since image processing problems are often best solved in a fashion utilizing many small blocks, we want an environment that separates many small parallelizable units of work effectively. The ghc clusters (GTX 1080) with CUDA are ideal as CUDA inherently parallelizes into many small blocks.

8 Schedule

Week 1 (10/28-11/3): Understanding algorithm

Week 2 (11/4-11/10): Sequential MPEG-1

Week 3 (11/11-11/17): Sequential MPEG-1 (cont) + Parallelizing preprocessing

Week 4 (11/18-11/24): Parallelizing preprocessing (cont) + entropy encoding

Week 5 (11/25-12/1): Parallelizing entropy encoding (cont)

Week 6 (12/2-12/9): Performance analysis