Performance Grading of GPU-based Implementation of Space Computing Systems Image Compression

Olympia Kremmyda, Vasilis Dimitsas, Dimitris Gizopoulos
ol-krem@di.uoa.gr, vdimi@di.uoa.gr, dgizop@di.uoa.gr
Dept. of Informatics & Telecomm., University of Athens, Greece

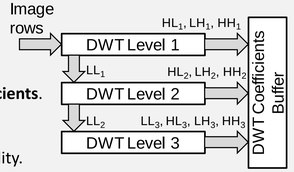
Introduction

- First full CUDA implementation of an important image compression algorithm for space data systems recommended in CCSDS standards [1], [2].
- The algorithm consists of two major parts:
  - Discrete Wavelet Transform (DWT) on the image raw data
  - Bit Plane Encoding (BPE).

Algorithms

DWT

- Three-level
- 2D wavelet transformation
- Image data is decomposed and transformed into **wavelets coefficients**.
- floating-point DWT version
- **lossy compression**
- effective image compression quality.



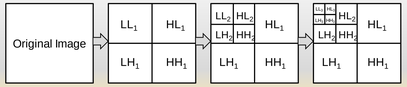


Figure 2: DWT operates in three levels (up) and separates the image in 10 subbands (down).

BPE

- Coefficients are grouped into blocks of 64
- Blocks are further grouped into **segments**[1]
- Encoded with Rice Algorithm[3]

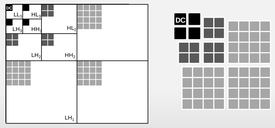
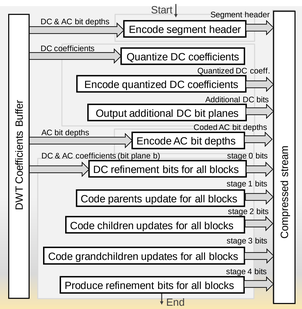


Figure 3: The structure of the DWT wavelet coefficients block and its relation to the image pixels (left). The flow of the BPE part of the image compression algorithm (right).

1. CCSDS recommends that the minimum number of blocks per segment be set to 16



Implementation

DWT

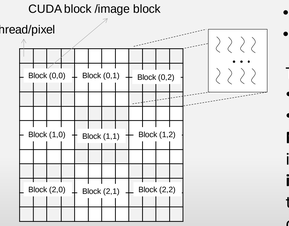


Figure 4: Assignment of blocks to the image input array (image size 12x12, block size 4x4).

The image is distributed into threads.
- Each thread processes a single pixel.
- A CUDA block processes an image block.

Two steps for DWT:
- row transform
- column transform

No transposition is done, instead **shared memory is used** to store the transformed pixels during the 3 levels.

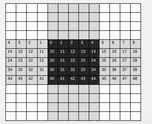


Figure 5: Shared memory layout after the transfer and the extension of the chunk of data of block 0. (image size 10x10, , block size 5x5)

BPE

Wavelet Coefficients are grouped into blocks and segments using:
- CUDA
- and shared memory

In BPE:
- Each segment is separately and independently encoded.
- A 1D CUDA grid is formed
- Each thread encodes a single segment.
- Encoding information are copied into dynamically allocated local variables in each thread

Results

	Server 1	Server 2
CPU	AMD Phenom™ II X4 965 4 cores, @ 3.4GHz, 45nm (Released **November 2009**) 8 GB main memory	Intel® Core™ i7-3970X 6 cores, @ 3.50GHz, 32nm (Released **November 2012**) 32 GB main memory
GPU	NVIDIA Tesla C2070 (Fermi architecture), 6GB GDDR5 RAM, 448 CUDA cores, 40nm (Released **July 2010**)	NVIDIA Tesla K20 (Kepler architecture), 5GB GDDR5 RAM, 2496 CUDA cores, 28nm (Released **November 2012**)

Table 1: Configurations of the two evaluation systems.

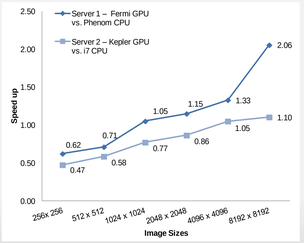
- 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512 threads per CUDA block;
- 4, 16 and 32 image coefficient blocks per segment [2]

The execution time reported below represent the best cases after testing all the possible combinations.

Image size	Server 1 GPU (Fermi)	Server 1 CPU (Phenom)	Server 2 GPU (Kepler)	Server 2 CPU (i7)
256 x 256	19.63	12.19	18.83	8.90
512 x 512	49.56	35.19	47.52	27.78
1024 x 1024	142.69	149.86	123.33	95.08
2048 x 2048	533.14	612.45	449.46	388.50
4096 x 4096	1983.89	2636.69	1795.63	1884.27
8192 x 8192	6344.08	13038.89	7008.64	7733.71

Table 2: Execution times (msec) of the complete algorithm on Server 1 and Server 2. Baseline CPU and GPU executions are shown for all different image sizes.

2. The CCSDS standard requires a minimum of 16 coefficients blocks per segment. For the completeness of our space exploration we considered the 16 and 32 blocks per segment cases which are standard-compliant and the case of 4 blocks per segment as a lower limit of our experiments.



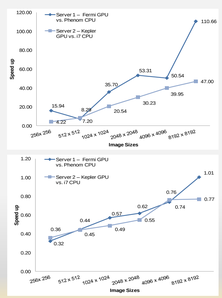


Figure 6: GPU vs. CPU overall speedup (CPU/GPU time) (left). Speedup of DWT execution on the GPUs of both servers (upper right) . BPE kernel vs. BPE function (CPU) speedup (in most cases a slowdown is measured; CPU/GPU < 1). (lower right)

The **main remarks** are the following:
- GPU execution outperforms CPU execution
- DWT delivers speedup over CPU execution
- Slowdown of the BPE execution
  - very control-intensive task
  - memory coalescing inability
- GPU execution in Server 1 delivers better speedups compared to Server 2 (better performance of the i7 CPU of Server 2)

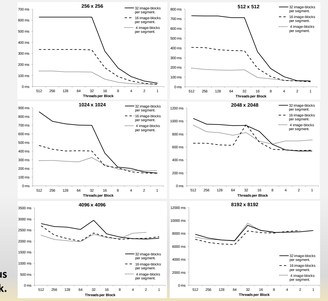


Figure 7: Execution on Server 1 with various numbers of threads per CUDA block.

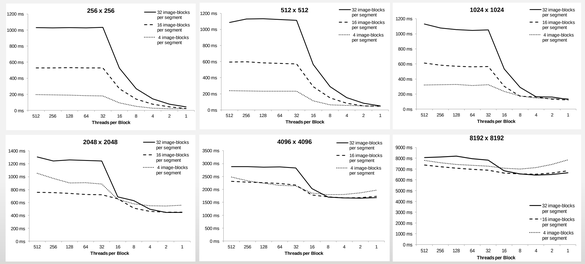
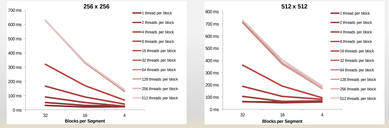


Figure 8: Execution on Server 2 with various numbers of threads per CUDA block.

- Smaller images (256x256, 512x512, 1024x1024)
  - less synchronization inside the block
- Larger images (2048x2048, 4096x4096, 8192x8192)
  - threads can be uniformly separated into two warps
  - switching of warps succeeds to hide latency.



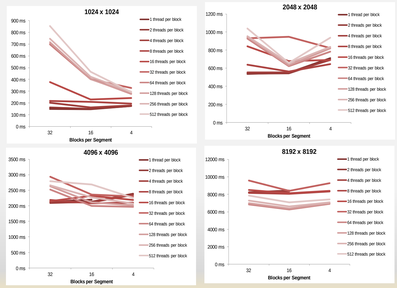


Figure 9: Execution on Server 1 with variations on image-blocks per segment in the BPE kernel

- Smaller images
  - each thread is assigned less workload
  - branch divergence and ineffective memory coalescing have smaller impact
- Larger images
  - execution time decreases, regardless the number of image-blocks per segment
  - the workload is larger
  - can effectively hide the inter-warp latencies.

Conclusion

- **First** full CUDA implementation of CCSDS 122.0-B-1.
- GPU execution delivers to **1.1x to 2.1x** speedup.
- **BPE** part of the algorithm limits the overall speedup.

Acknowledgments

This work is supported by EU's European Social Fund (ESF) and Greek national funds under the "DIaSTEMA" project and the "Thales/HOLISTIC" project as well as by the Special Account for Research Grants (SARG) of the University of Athens.

References

1. Image Data Compression CCSDS-120.1-G-1. 2007. http://public.ccsds.org/publications/archive/120x1g1e2.pdf
2. Image Data Compression CCSDS-122.1-B-1. 2005. http://public.ccsds.org/publications/archive/122x0b1c3.pdf
3. Rice R.F., "Some Practical Universal Noiseless Coding Techniques", PL-PUB-79-22; NASA-CR-158515. 1979.