GPU-BASED SPATIALLY DIVIDED PREDICTIVE PARTITIONED VECTOR QUANTIZATION FOR GIFTS ULTRASPECTRAL DATA COMPRESSION

Shih-Chieh Wei[1] *and Bormin Huang*[2*]

[1]Department of Information Management, Tamkang University, Tamsui 25137, Taiwan
[2]Space Science and Engineering Center, University of Wisconsin-Madison, Madison, WI 53706, USA

ABSTRACT

Predictive partitioned vector quantization (PPVQ) has been proven to be an effective lossless compression scheme for ultraspectral sounder data. In previous work, we have identified the two most time-consuming stages of PPVQ for implementation on GPU. By using 4 GPUs and a spectral division design in sharing the workload, we showed a 42x speedup on NASA's Geostationary Imaging Fourier Transform Spectrometer (GIFTS) dataset compared to its original single-threaded CPU code. In this paper, an alternative spatial division design is developed to run on 4 GPUs. The experiment on the GIFTS dataset shows that a 72x speedup can be further achieved by this new design of the GPU-based PPVQ compression scheme.

Index Terms—Graphic processor unit, data compression, GIFTS sounder data

1. INTRODUCTION

Used mainly for earth surface observation, ultraspectral sounder data is known for its large size due to high resolution in spectral, spatial and temporal dimensions. For retrieval of geophysical parameters, the sounder data can tolerate little noise. Thus lossless compression is often desired for archiving the ultraspectral sounder data. Predictive partitioned vector quantization (PPVQ) has been proven to be an effective lossless compression scheme for ultraspectral sounder data [1]. It consists of linear prediction, bit depth partitioning, vector quantization, and entropy coding.

Our previous work [2] showed that among the four stages, linear prediction takes up 10% and vector quantization takes up 87% of the compression time. Both stages are suitable for GPU implementation. By a spectral division design in using 4 GPUs, we have demonstrated a 42x speedup on NASA's Geostationary Imaging Fourier Transform Spectrometer (GIFTS) [3] dataset compared to its original single-threaded CPU code. In this paper, an alternative spatial division design is developed to run on 4 GPUs. Section 2 will describe our methodology. Section 3 shows the experimental result. A short summary will be given in section 4.

* Corresponding author. E-mail: bormin@ssec.wisc.edu

2. THE GPU-BASED PPVQ COMPRESSION SCHEME

GIFTS represents a revolutionary step in satellite remote sensing of geophysical parameters, and poses a challenge to process the large amount of data it will collect [3]. Most of a raw GIFTS data cube is made up of a 128x128 array of interferograms as shown in Fig. 1.

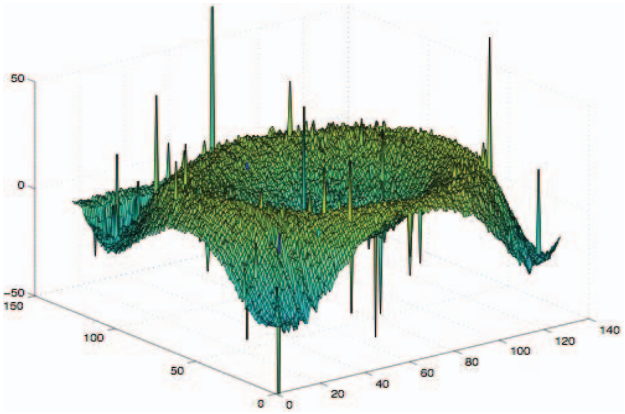


Fig. 1. A typical GIFTS data cube.

2.1. The PPVQ algorithm

A data flow diagram for the PPVQ compression scheme is shown in Fig. 2. The original sounder data mainly goes through the following 4 stages in sequence to produce a compressed result.

Linear prediction (LP) This stage can reduce the dynamic range of a pixel by knowledge of its previous channels. The spatial frame X of channel i can be linearly predicted by $\hat{X}_i$ as follows.

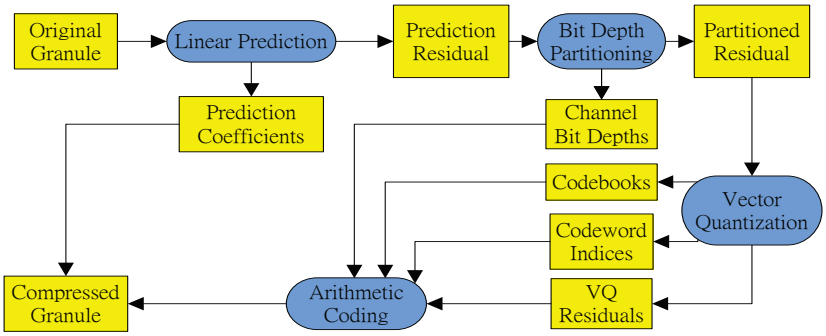
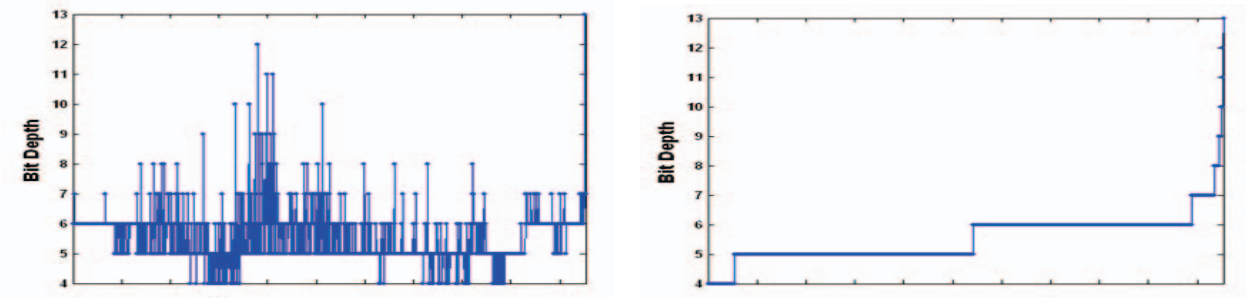


Fig. 2. The data flow diagram for PPVQ.



Channel index (a)

Reordered channel index (b)

Fig. 3. The bit depth of interferograms before (a) and after (b) bit depth partitioning.

$$\hat{X}_i = \sum_{k=1}^{n_p} c_k X_{i-k} \quad \text{or} \quad \hat{X}_i = X_p C \quad \text{with}$$

$$C = (X_p^T X_p)^{-1} (X_p^T \hat{X}_i) \qquad (1)$$

where $\hat{X}_i$ is the vector of channel i representing the predicted 2D spatial frame, Xp is the matrix consisting of n_p neighboring channels, and C is the vector of the prediction coefficients for channel i.

Bit depth partitioning (BP) This stage groups the channel residual by bit depth as shown in Fig. 3. Channels with the same bit depth are assigned to the same partition and VQ is applied to each partition separately.

Vector quantization (VQ). This stage divides high-dimensional data (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point [4].

Entroy coding (AC) This stage assigns codes to symbols so as to match code lengths with the probabilities of the symbols [5]. AC is applied on the VQ output of each bit depth partition which includes the codebook, residual, and the index.

Based on profiling of its CPU code, linear prediction and vector quantization are identified as the two most time-consuming stages of PPVQ. Both are suitable for massively parallel computation on GPUs [2].

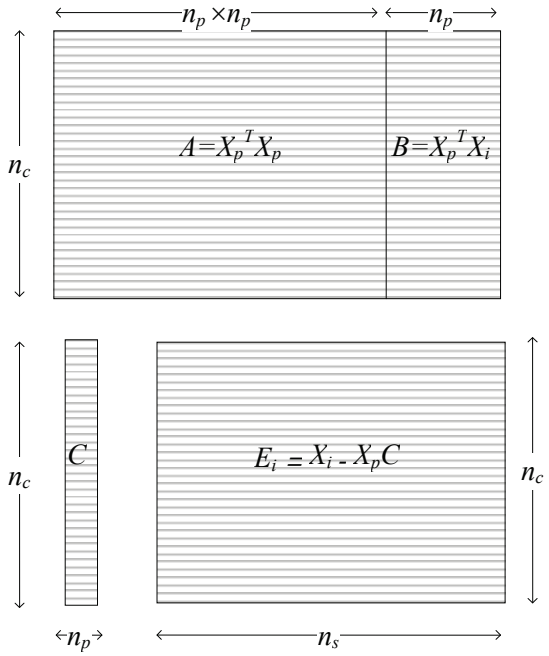


Fig.4. The three kernels used for linear prediction.

2.2.The GPU-CPU pipeline implementation

For GPU implementation, three kernels are used for performing the linear prediction as shown in Fig. 4. The first kernel prepares the matrix $A = X_p{}^T X_p$ and the vector $B = X_p{}^T X_i$ where *Xp* consists of the previous neighboring n_p channel frames of channel *i*. This kernel is launched with a total of $n_c \times (n_p \times n_p + n_p)$ threads, each responsible for computing an element in the matrix *A* (of size $n_p \times n_p$) or the vector *B* (of size n_p) for a specific channel. These threads can be executed in thread blocks of maximum capacity allowable by the hardware.

The second kernel solves the linear equation $AC = B$ for the n_p prediction coefficients in solution vector $\boldsymbol{C}$. This kernel is launched with a total of n_c threads, each responsible for computing the prediction coefficient vector $\boldsymbol{C}$ (of size n_p) for a specific channel. The third kernel then computes the linear prediction residual $E_i = X_i - \hat{X}_i$ where $\hat{X}_i = X_p C$ is the linearly predicted frame of channel *i*. This kernel is launched with a total of $n_c \times n_s$ threads, each responsible for computing the residual of a spatial location in the vector E_i for a particular channel *i*. A thread block of maximum capacity of threads can be used for execution.

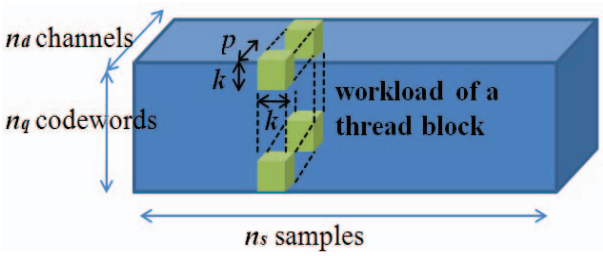


Fig.5. The schematic of a thread block in codeword assignment.

The vector quantization consists of codebook training and codeword encoding. The codebook training uses the well-known Linde-Buzo-Gray (LBG) algorithm [4] but for fast learning, just takes the first training vector in each group as the initial codeword of the codebook. LBG is an iterative process of codeword assignment and codeword computation. In particular, codeword assignment is most time-consuming for GIFTS. Given a codebook of n_q codeword vectors, the task of codebook assignment is to assign the smallest codeword in distortion to each of the n_s training vectors. All vectors have a dimension of n_d channels. For GIFTS data, typical values are n_s = 16384; n_q = 256, 512, 1024, or 2048; and n_d = 1~500. Consider n_s training vectors and each needs to find the closest one among all n_q codewords. As shown in Fig. 5, a GPU thread block shares the workload of *k* training vectors using *k* by *p* threads. All vectors have a dimension of n_d channels.

In one observation, a GIFTS data cube of 192MB is obtained which is divided into 6 sub-cubes of equal size 32MB, each containing 1031 channels. The 6 sub-cubes are the LW real part, the LW imaginary part, the SMW real part of channels 1 ~ 1031, the SMW real part of channels 1032 ~ 2062, the SMW imaginary part of channels 1 ~ 1031, and the SMW imaginary part of channels 1032 ~ 2062. As shown in Fig. 6, each sub-cube goes through the LP, BP, VQ and AC steps sequentially for compression. The LP and VQ steps are mainly done on GPU, while the BP and AC steps are done on CPU. The length of each step is not drawn to scale. The pipeline is synchronized (denoted by the 6 vertical dotted lines) such that when one sub-cube is doing AC, the next sub-cube is doing LP, BP, and VQ. Overlap of the LP, BP, VQ steps with the AC step in time can reduce the total processing time of a data cube [2].

When there are more than one GPU available, more speedup can be achieved by using multiple GPUs simultaneously. For division of the compression workload among several GPUs, the spatial division design is used in this paper compared to the spectral division design used in previous work [2]. For spectral division design, each GPU is assigned to do the linear prediction of $(n_c - n_p)/ n_g$ contiguous channels where n_g is the number of GPUs in use. In VQ stage, each GPU is responsible for quantization of a bit depth partition whose number of grouped channels might vary in size. For current spatial division design, each data cube is spatially divided into n_g equal-sized sub-cubes. Then each spatial sub-cube is assigned an independent GPU to do the subsequent LP and VQ stages of compression work.

3. THE EXPERIMENTAL RESULT

The experiment is carried out on a machine with a quad-core 2.4 GHz AMD CPU, and 4 Nvidia Tesla 1.3 GHz GPUs. Earlier we have developed a source code in C for PPVQ. For comparison, the single threaded C code was used as the CPU reference version. It was compiler-optimized by the gcc O3 optimization option. All GPU speedup was compared against this optimized CPU version. Using only one CPU core

Fig. 7 shows the speedup result of using multiple GPUs on the 5 GIFTS data cubes. In Fig. 7 (a), the speedup profile of PPVQ using 4 GPUs on the 5 GIFTS data cubes is shown for our spatial division design. Note that the speedup for LP is quite dependent on data and shows a large variation. In Fig. 7 (b), the speedup profile of PPVQ using 1 through 4 GPUs is shown for our spatial division design. The speedup in (b) is averaged over the 5 data cubes. It can be seen that when more GPUs are used, a quite scalable speedup in VQ can be obtained using the current spatial division design. Using the pipeline design of LP and VQ on GPU and AC on CPU, the average compression time of the GPU-based PPVQ on a data cube takes about 13 seconds for the spatial division design.

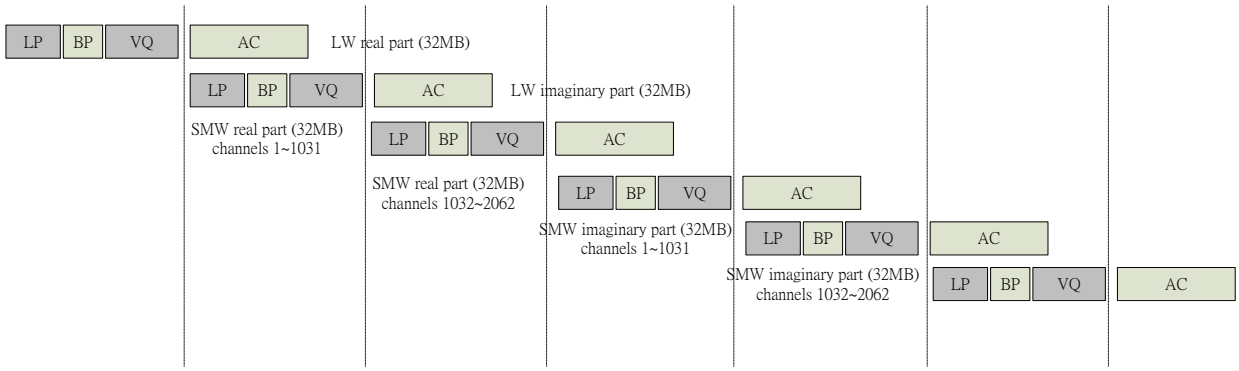
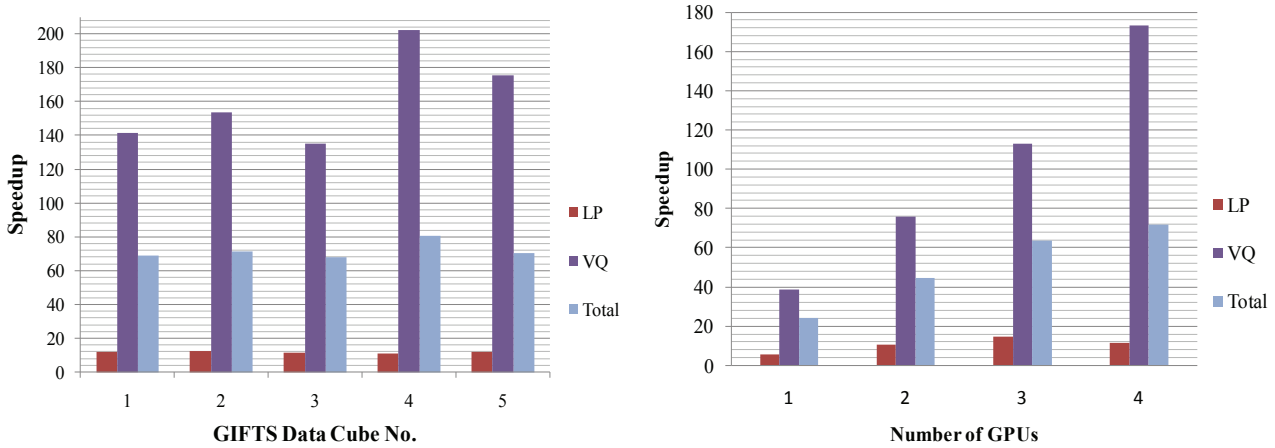


Fig.6. The pipeline design of PPVQ for GIFTS data



(a) (b)

Fig. 7. The GPU speedup profile of PPVQ on GIFTS data using the spatial division design. (a) The 4 GPU speedup for the 5 cubes in dataset; (b) The 1~4 GPU speedup averaged over the 5 cubes in dataset.

4. SUMMARY

The predictive partitioned vector quantization (PPVQ) compression scheme is known for its effectiveness in lossless compression of ultraspectral sounder data [1]. In previous work [2], the two most time-consuming stages of linear prediction and vector quantization are identified and implemented on GPU. Furthermore a sounder dependent speedup technique is designed for a GIFTS data cube which is divided into 6 equal-sized parts to facilitate CPU and GPU processing in pipeline. In current work, we found that when up to four GPUs are used to share the workload, the spatial division design can achieve a promising 72x speedup compared to its original single-threaded CPU code for lossless compression of NASA's GIFTS ultraspectral sounder data.

5. REFERENCES

[1] B. Huang, A. Ahuja, and H.-L. Huang, "Predictive partitioned vector quantization for hyperspectral sounder data compression, " *Proc. SPIE.*, vol. 5548, pp.70-77, 2004.

[2] S.-C. Wei and B. Huang, "A GPU-based implementation of predictive partitioned vector quantization for compression of ultraspectral sounder data," *Proc. SPIE.*, vol. 7810, p.781017, 2010.

[3] W. L. Smith, F. W. Harrison, D. E. Hinton, H. E. Revercomb, G. E. Bingham, R. Petersen, and J. C. Dodge, "GIFTS - the precursor geostationary satellite component of the future Earth Observing System," *Proc. IGARSS'02*, vol. 1, pp. 357-361, 2002.

[4] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantization design, " *IEEE Trans. Commun.*, vol. 28, pp. 84-95, 1980.

[5] I. H. Witten, R. M. Neal, and J. C. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp.520-541, June 1987.