IMPROVING THE SCALABILITY OF PARALLEL ALGORITHMS FOR HYPERSPECTRAL IMAGE ANALYSIS USING ADAPTIVE MESSAGE COMPRESSION

Antonio Plaza, Javier Plaza, Abel Paz

Department of Technology of Computers and Communications
Escuela Politécnica de Cáceres, University of Extremadura
Avda. de la Universidad s/n, E-10071 Cáceres, Spain

ABSTRACT

In previous work, we have reported that the scalability of parallel processing algorithms for hyperspectral image analysis is affected by the amount of data to exchanged through the communication network of the parallel system. However, large messages are common in hyperspectral imaging applications since processing algorithms are often pixel-based, and each pixel vector to be exchanged through the communication network is made up of hundreds of spectral values. Thus, decreasing the amount of data to be exchanged could improve the scalability and parallel performance. In this paper, we propose a new framework based on intelligent utilization of data compression techniques for improving the scalability of a standard spectral unmixin-based parallel hyperspectral processing chain on heterogeneous networks of workstations. Our experimental results indicate that adaptive, wavelet-based lossy compression can lead to improvements in the scalability of the parallel algorithms without significantly sacrificing algorithm analysis accuracy.

Index Terms— Hyperspectral imaging, parallel computing, spectral mixture analysis, data compression

1. INTRODUCTION

High performance computing has become a standard solution in order to deal with high response times in large-scale remote sensing applications [1]. For instance, a trend in geocomputation is to utilize highly heterogeneous and distributed parallel platforms which can benefit from local (user) computing resources in order to efficiently store and handle high-dimensional data archives resulting from latest-generation imaging instruments such as NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [2], which can record the visible and near-infrared spectrum (wavelength region from 0.4 to 2.5 micrometers) of the reflected light of an area 2 to 12 kilometers wide and several kilometers long using 224 narrow spectral bands. Such heterogeneous networks of computers (HNOCs) enable the use of existing computing resources and provide incremental scalability of hardware components [3].

In the context of hyperspectral imaging applications, it has been demonstrated that the scalability of parallel algorithms is directly related to the size of the messages to be exchanged through the communication network of the system when the parallel algorithm is run [4], i.e. the latency for large message sizes (typical in hyperspectral imaging applications, since each pixel vector is made up of hundreds of spectral values) is extremely sensitive to the size of the message. Thus, decreasing the size of the messages could significantly reduce the latency and hence improve the scalability or parallel performance [5]. In this paper, we propose a new framework based on the utilization of lossy data compression techniques for improving the scalability of parallel hyperspectral imaging algorithms on both homogeneous and heterogeneous parallel computers. Lossy compression involves some *acceptable* loss of information, where the term acceptable needs to be substantiated via extensive experiments analyzing the balance between the increase in parallel performance and the potential reduction in algorithm analysis accuracy resulting from the loss of information. Our proposed approach makes use of wavelet-based data compression techniques [6] for reducing the amount of communications required by parallel hyperspectral algorithms.

The remainder of the paper is organized as follows. In Section 2, we describe a parallel hyperspectral processing chain used for evaluation in this work. Section 3 discusses the proposed wavelet-based compression scheme. Section 4 presents an experimental validation of the proposed approach, using different platforms and scenes. Section 5 concludes with some remarks and hints at future research directions.

2. PARALLEL ALGORITHM

2.1. Spectral mixture problem formulation

Let us assume that a hyperspectral scene with N bands is denoted by $\mathbf{F}$, in which a pixel of the scene is represented by a vector $\mathbf{f}_i = [f_{i1}, f_{i2}, \cdots, f_{in}] \in \Re^N$, where $\Re$ denotes the set of real numbers in which the pixel's spectral response f_{ik} at sensor wavelengths $k = 1, \ldots, N$ is included. Under a linear assumption, each pixel be modeled as:

$$\mathbf{f}_i = \sum_{e=1}^{E} \mathbf{e}_e \cdot a_{\mathbf{e}_e} + \mathbf{n}, \quad (1)$$

where $\mathbf{e}_e$ denotes the spectral response of a pure spectral signature (*endmember* in hyperspectral imaging terminology), $a_{\mathbf{e}_e}$ is a scalar value designating the fractional abundance of the endmember $\mathbf{e}_e$, E is the total number of endmembers, and $\mathbf{n}$ is a noise vector. The solution of the linear spectral mixture problem in (1) relies on the correct determination of a set $\{\mathbf{e}_e\}_{e=1}^{E}$ of endmembers and their correspondent abundance fractions $\{a_{\mathbf{e}_e}\}_{e=1}^{E}$ at each pixel $\mathbf{f}_i$.

2.2. Parallel processing chain

Let us assume that p_i denotes a processing node weighted by its relative cycle-time t_i. Similarly, let us assume that c_{ij} denotes the maximum capacity of the slowest link in the path of physical communication links from p_i to p_j. In order to balance the load in a fully heterogeneous environment, processor p_i should accomplish a share of $\alpha_i \cdot W$ of the original workload, denoted by W, to be accomplished by a certain algorithm, with $\alpha_i \geq 0$ for $1 \leq i \leq P$ and $\sum_{i=1}^{P} \alpha_i = 1$, being P the total number of processors in the system. With the above notation in mind, the inputs to the parallel hyperspectral processing chain considered in this work[1] are a hyperspectral image cube $\mathbf{F}$ with N spectral bands and T pixel vectors; the number of endmembers to be extracted, E, a maximum number of projections, K; a cut-off threshold value, v_c, used to select as final endmembers only those pixels that have been selected as extreme pixels at least v_c times after K projections; and a threshold angle, v_a, used to discard redundant endmembers. The chain can be summarized by several steps:

1. *Workload estimation.* Set $\alpha_i = \lfloor \frac{(1/t_i)}{\sum_{i=1}^{P}(1/t_i)} \rfloor$ for all $i \in \{1, \cdots, P\}$, i.e. this step first approximates the $\{\alpha_i\}_{i=1}^{P}$ so that $\alpha_i \cdot t_i \approx$ *const* for all processors. Then use the estimated values of $\{\alpha_i\}_{i=1}^{P}$ to generate P partitions of the input hyperspectral data set as follows:
   (a) Establish an initial partitioning of the data so that the number of pixel vectors in each partition is proportional to the values of $\{\alpha_i\}_{i=1}^{P}$, assuming that no upper bound exist on the number of pixel vectors that can be stored by the local memory.
   (b) For each processor p_i, check if the number of pixel vectors assigned to it is greater than the upper bound. For all the processors whose upper bounds are exceeded, assign them a number of pixels equal to their upper bounds. Now, solve the partitioning problem of a set with remaining pixel vectors over the remaining processors until all pixels in the input data have been assigned.

[1]Pixel purity index available in commercial software packages such as ITTVIS ENVI software, see http://www.ittvis.com for additional details

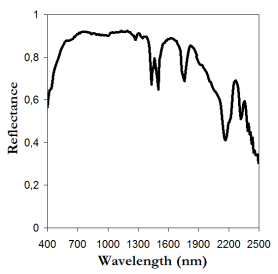
2. *Skewer generation.* Generate K random unit vectors $\{\mathbf{skewer}_j\}_{j=1}^{K}$ in parallel, and broadcast the entire set of skewers to all the workers.
3. *Extreme projections.* For each $\mathbf{skewer}_j$, each worker projects all the sample pixel vectors at its local partition l (where $1 \leq l \leq P$) onto $\mathbf{skewer}_j$ to find sample vectors at its extreme projections, and forms an extrema set for $\mathbf{skewer}_j$ which is denoted by $S_{extrema}^{(l)}(\mathbf{skewer}_j)$. Now each worker calculates the number of times that each pixel vector $\mathbf{f}_i^{(l)}$ in each local partition is selected as extreme using:

$$N_{times}^{(l)}(\mathbf{f}_i^{(l)}) = \sum_{j=1}^{K} I_{S_{extrema}^{(l)}(\mathbf{skewer}_j)}(\mathbf{f}_i^{(l)}) \quad (2)$$

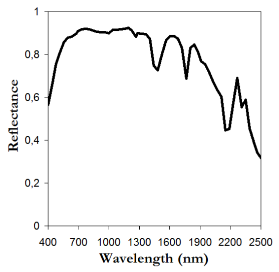
4. *Endmember selection.* Each worker selects those pixel vectors which satisfy $N_{times}^{(l)}(\mathbf{f}_i^{(l)}) > v_c$, and then sends the spatial coordinates of those pixels to the master node. The master now forms a unique set $\{\mathbf{e}_e\}_{e=1}^{E}$ by calculating the SAD for all possible pixel vector pairs provided by the workers in parallel, and discarding those pixels with angle values below v_a.
5. *Spectral unmixing.* The master broadcasts the final set of endmembers $\{\mathbf{e}_e\}_{e=1}^{E}$ to all workers. Each worker then obtains a set of fractional abundances $\{a_{\mathbf{e}_1}^{(l)}, a_{\mathbf{e}_2}^{(l)}, \cdots, a_{\mathbf{e}_E}^{(l)}\}$ for each pixel vector $\mathbf{f}_i^{(l)}$ in its local partition l, so that the term $\mathbf{n}$ in the following expression is minimized: $\mathbf{f}_i^{(l)} = \mathbf{e}_1 \cdot a_{\mathbf{e}_1}^{(l)} + \mathbf{e}_2 \cdot a_{\mathbf{e}_2}^{(l)} + \cdots + \mathbf{e}_E \cdot a_{\mathbf{e}_E}^{(l)} + \mathbf{n}$. The workers finally send the local results to the master, which combines them and forms the final output.

3. ADAPTIVE DATA COMPRESSION

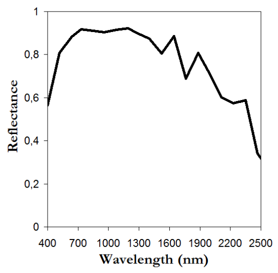
The idea of the adopted lossy compression method is to apply a discrete wavelet transform [6] in the spectral domain before communicating each vector through the network. The lossy compression procedure does not only reduce the data volume to be communicated, but it can also preserve the characteristics of the spectral signatures. This is due to the intrinsic property of wavelet transforms, which preserve high and low-frequency features during the signal decomposition, therefore retaining peaks and valleys found in typical spectra. An advantage of using the one-dimensional wavelet transform (across spectral bands) is the fact that this is a very localized operation, which means less accesses to secondary storage and better cache memory utilization. This can be beneficial in terms of achieving data locality while, at the same time, reducing interprocessor communications. Since the compression is pixel-based, it can be effectively integrated in the parallel algorithm described in section 2. A description of the wavelet-based encoder/decoder modules follows:



(a) Level 1: 200 bands



(b) Level 2: 100 bands



(c) Level 3: 25 bands

Fig. 1. Example of a mineral spectral signature and different levels of wavelet decomposition for the lowpass component.

- *Encoder*. Each time the master processor or a slave processor needs to communicate a pixel vector through the system, its associated signal (spectral signature) is decomposed (encoded) using a Daubechies wavelet (filter size four) [6]. This transform decomposes each signature into a set of composite bands that are linear, weighted combinations of the original spectral bands. An example of the encoding process for a standard spectral signature is shown in Fig. 1. As the number of wavelet decomposition levels increases, the structure of the spectral signature becomes smoother than the structure of the original signature. It can be seen that the overall structure of the signal is recognizable until level 2 and more degraded at level 3.
- *Decoder*. An approximation of the original spectrum at pixel vector can be reconstructed (decoded) after reception of the compressed signal and the wavelet coefficients (which are transmitted through the network) using an inverse discrete wavelet transform [6]. Because the reconstructed spectral data are produced from the approximation, the more levels is which the signal is decomposed, the more different the reconstructed signal is with regards to the original signal, because of the loss of high-pass components.

4. EXPERIMENTAL RESULTS

4.1. Hyperspectral data

The image scene used for experiments in this work was collected by the AVIRIS instrument, flown over the World Trade Center (WTC) area in New York City on September 16, 2001, just five days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. The data set consists of 614×512 pixels, 200 spectral values per pixel (after removal of noisy bands), for a total size of about 123 MB. Fig. 2(left) shows a false color composite of the data set selected for experiments. Fig. 2(right) shows a thermal map displaying the target locations of the thermal hot spots, used in this work as ground-truth to validate the parallel hyperspectral processing algorithms.

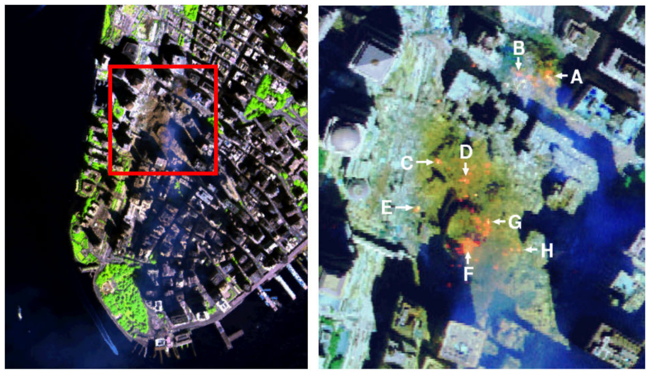


Fig. 2. Hyperspectral image (left). Location of fires in World Trade Center (right).

Table 1 evaluates the accuracy of the hyperspectral processing chain described in section 2 in estimating the sub-pixel abundance of fires in Fig. 2(right), taking advantage of the information about the area covered by each thermal hot spot available from U.S. Geological Survey (USGS)[2]. Since each pixel in the AVIRIS scene has a size of 1.7 square meters, it is inferred that the thermal hot spots are sub-pixel in nature, and thus require spectral unmixing in order to be characterized. Experiments in Table 1 demonstrate that the processing chain can provide accurate estimations of the area covered by thermal hot spots. Here, the processing chain was run using a total of $K = 10^4$ skewers, with the cutoff threshold parameter v_c set to the mean of N_{times} scores after 10^4 iterations, and the threshold angle value set to $v_a = 0.1$, a reasonable limit of tolerance for this metric. The number of endmembers was set to $E = 15$ after estimating the virtual dimensionality of the scene.

4.2. Parallel performance

The fully heterogeneous network considered in our study consists of 16 different workstations with the main features summarized on Table 2, and four heterogeneous communication

[2]http://speclab.cr.usgs.gov/wtc

Table 1. Comparison of area estimation (in square meters) for each thermal hot spot by the hyperspectral processing chain (USGS reference values are also included).

Thermal hot spot	Latitude (North)	Longitude (West)	Temperature (Kelvin)	Area (USGS)	Area (Chain)
'A'	$40^{o}42'47.18''$	$74^{o}00'41.43''$	1000	0.56	0.55
'B'	$40^{o}42'47.14''$	$74^{o}00'43.53''$	830	0.08	0.06
'C'	$40^{o}42'42.89''$	$74^{o}00'48.88''$	900	0.80	0.78
'D'	$40^{o}42'41.99''$	$74^{o}00'46.94''$	790	0.80	0.81
'E'	$40^{o}42'40.58''$	$74^{o}00'50.15''$	710	0.40	0.45
'F'	$40^{o}42'38.74''$	$74^{o}00'46.70''$	700	0.40	0.37
'G'	$40^{o}42'39.94''$	$74^{o}00'45.37''$	1020	0.04	0.05
'H'	$40^{o}42'38.60''$	$74^{o}00'43.51''$	820	0.08	0.09

Table 2. Specifications of heterogeneous processors.

Processor number	Architecture Specification	Cycle-time (seconds/megaflop)	Memory (MB)	Cache (KB)
p_1	Free BSD – i386 Intel Pentium 4	0.0058	2048	1024
p_2, p_5, p_8	Linux – Intel Xeon	0.0102	1024	512
p_3	Linux – AMD Athlon	0.0026	7748	512
p_4, p_6, p_7, p_9	Linux – Intel Xeon	0.0072	1024	1024
p_{10}	SunOS – SUNW UltraSparc-5	0.0451	512	2048
$p_{11} - p_{16}$	Linux – AMD Athlon	0.0131	2048	1024

segments. Table 2. Table 3 shows the total time spent by the parallel heterogeneous algorithm in computations and communications, for different values of t_a. In our implementation the communications can only be partially overlapped with computations due to data dependencies. In this regard, the communication times reported on Table 3 actually correspond to the non-overlapped portions of the different types of communications. The total execution time of the parallel algorithm without adaptive compression ($t_a = 0.0$) is 88.58 seconds, out of which 14.2 seconds (16.03% of the total execution time) correspond to communications that could not be overlapped with computations. When compression was introduced, $t_a = 0.1$ provided the most appropriate reconstruction threshold in terms of both algorithm accuracy and parallel performance. In this case, the total execution time of the heterogeneous algorithm in the considered network was 85.28 seconds, out of which 7.3 seconds (8.56% of the total execution time) correspond to communications. This resulted in an improvement in the scalability or speedup (number of times that the parallel algorithm was faster than the sequential one), as reported on Fig. 3.

5. CONCLUSIONS

In this paper, we have studied the impact of incorporating a lossy compression strategy at run-time in order to reduce the amount of information that is communicated in a parallel and heterogeneous remote sensing application. The proposed approach adaptively determines the most appropriate level of compression in order to avoid degrading the spectral signatures associated to each pixel vector in the remotely sensed scene. Our experimental results indicate that the incorporation of adaptive lossy compression can significantly reduce the communications at the expense of only slightly increasing computations, which results in improvements in the measured speedups without sacrificing analysis accuracy.

Table 3. Computation and communication times (seconds) for different steps of the parallel hyperspectral processing chain: WE (workload estimation), SG (skewer generation), EE (endmember extraction), SU (spectral unmixing), with ($t_a > 0.0$) and without ($t_a = 0.0$) adaptive compression.

	Computations		Communications				
t_a	Sequential	Parallel	WE	SG	EE	SU	Total
0.0	13.25	61.13	6.21	4.45	0.65	2.89	88.59
0.1	15.44	62.54	3.12	2.21	0.43	1.54	85.28
0.2	14.89	62.33	2.79	1.86	0.56	1.46	83.89

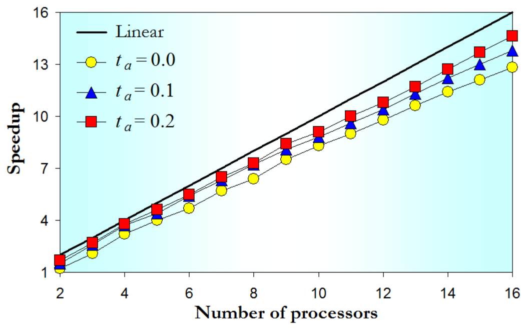


Fig. 3. Speedups of the parallel algorithm with ($t_a > 0.0$) and without ($t_a = 0.0$) adaptive compression.

6. ACKNOWLEDGEMENT

This work has been supported by the EC Marie Curie Research Training Networks Programme (HYPER-I-NET, ref. MRTN-CT-2006-035927) and the Spanish Ministry of Science and Innovation (EODIX, ref. AYA2008-05965-C04-02).

7. REFERENCES

[1] A. Plaza and C.-I Chang, *High Performance Computing in Remote Sensing*, CRC Press, Boca Raton: Florida, 2007.

[2] R. O. Green et al., "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, pp. 227-248, 1998.

[3] A. Plaza, "Parallel techniques for information extraction from hyperspectral imagery using heterogeneous networks of workstations," *J. Parallel Distr. Com.*, vol. 68, pp. 93-111, 2008.

[4] A. Plaza, "Parallel processing of remotely sensed hyperspectral imagery: full-pixel versus mixed-pixel classification," *Concurr. Comp-Pract. E.*, vol. 20, pp. 1539-1572, 2008.

[5] A. Plaza, D. Valencia and J. Plaza, "An experimental comparison of parallel algorithms for hyperspectral analysis using homogeneous and heterogeneous networks of workstations," *Parallel Comput.*, vol. 34, pp. 92-114, 2008.

[6] S. Kaewpijit, J. Le Moigne, and T. El-Ghazawi, "Automatic reduction of hyperspectral imagery using wavelet spectral analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, pp. 863-871, 2003.