Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 3252148, 9 pages
http://dx.doi.org/10.1155/2016/3252148

Research Article

Distributed Parallel Endmember Extraction of Hyperspectral Data Based on Spark

Zebin Wu,[1,2] Jinping Gu,[1] Yonglong Li,[1] Fu Xiao,[2] Jin Sun,[1] and Zhihui Wei[1]

[1]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[2]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing 210003, China*

Correspondence should be addressed to Zebin Wu; zebin.wu@gmail.com

Received 24 February 2016; Revised 6 May 2016; Accepted 22 May 2016

Academic Editor: Laurence T. Yang

Due to the increasing dimensionality and volume of remotely sensed hyperspectral data, the development of acceleration techniques for massive hyperspectral image analysis approaches is a very important challenge. Cloud computing offers many possibilities of distributed processing of hyperspectral datasets. This paper proposes a novel distributed parallel endmember extraction method based on iterative error analysis that utilizes cloud computing principles to efficiently process massive hyperspectral data. The proposed method takes advantage of technologies including MapReduce programming model, Hadoop Distributed File System (HDFS), and Apache Spark to realize distributed parallel implementation for hyperspectral endmember extraction, which significantly accelerates the computation of hyperspectral processing and provides high throughput access to large hyperspectral data. The experimental results, which are obtained by extracting endmembers of hyperspectral datasets on a cloud computing platform built on a cluster, demonstrate the effectiveness and computational efficiency of the proposed method.

1. Introduction

Hyperspectral remote sensing images are characterized by their large dimensionalities and volumes, with hundreds of nearly contiguous spectral channels. The hyperspectral image obtained from the earth's surface contains abundant information of space, radiation, and spectrum, which provides great help to the researchers for analyzing, processing, and monitoring the earth's surface information. However, due to the limitation of the sensor in spatial resolution and the diversity of the ground cover, the pixels of the image are generally mixed pixels. One of the most important techniques for hyperspectral data exploitation is endmember extraction [1], which characterizes mixed pixels as a combination of spectrally pure components (i.e., endmembers). Under the assumption of minimal secondary reflections and multiple scattering effects in data collection procedure, a number of techniques have been developed under the linear unmixing model in recent years [2], such as iterative error analysis (IEA) [3], independent component analysis (ICA) [4], dependent component analysis (DECA) [5], vertex component analysis (VCA) [6], simplex growing algorithm (SGA) [7], and minimum volume simplex analysis (MVSA) [8].

The abovementioned works have improved accuracy of hyperspectral endmember extraction enormously. However, most of them are very computationally intensive and therefore compromise their applicability in time-critical scenarios including military reconnaissance, environmental quality surveillance, monitoring of chemical contamination, wildfire tracking, and biological threat detection. As a result, in recent years, many techniques have been developed towards the improvement of these algorithms in high-performance computing architectures [9, 10]. For instance, low-weight integrated components such as field programmable gate arrays (FPGAs) [11], multicore central processing units (CPUs) [12, 13], and commodity graphics processing units (GPUs) [14, 15] have been successfully applied to accelerate computations. Nevertheless, with the development of hyperspectral imaging technology and the volume of the hyperspectral image growing, the traditional mechanism of allocating computational resources to a single machine is insufficient to meet the requirements of efficient hyperspectral processing. Accordingly, the fast endmember extraction of large hyperspectral dataset has been an important issue in the field of hyperspectral remote sensing. Fortunately, cloud computing has recently become more and more popular in the research

```
Input: Hyperspectral data $\mathbf{X}_{N\times L}$, the number of endmembers $m$.
(1) Initialization: Threshold of $\theta$ (spectral angle), the number $R$ for averaging the vectors
      with the largest error, and endmember matrix $\mathbf{U}$.
(2) Calculate the mean vector $\mathbf{mean}_{1\times L}$ of the hyperspectral data $\mathbf{X}_{N\times L}$.
(3) Perform the constrained unmixing on $\mathbf{X}_{N\times L}$ using $\mathbf{mean}_{1\times L}$ as endmember matrix, and
    get the image of the errors (named "error image") remaining after the unmixing.
repeat
(4) Find $R$ pixel vectors with the largest error in the error image, and extract the subset of
    the set of $R$ vectors, consisting of all those vectors which fall within the angle $\theta$ of the
    maximum error vector.
(5) Average the vectors in the subset to decrease the effects of outliers and noise, denoted
    as $\mathbf{p}_i$, and update endmember matrix $\mathbf{U} = [\mathbf{U}; \mathbf{p}_1]$.
until $m$ endmembers are extracted.
```

Algorithm 1: Endmember extraction based on IEA.

and commercial fields due to its homogeneous operating environment and full control over dedicated resources (e.g., networks, servers, storage, applications, and services) [16, 17]. Cloud computing can be considered as the improved processing for distributed processing, parallel processing, and grid computing [18]. However, to the best of our knowledge, despite the potential of large-scale distributed parallel computing in cloud computing and the demands of massive data processing in hyperspectral imaging, there are few cloud computing implementations of this category of algorithms in the literatures. In order to efficiently extract endmembers from massive hyperspectral data, a novel distributed parallel endmember extraction method based on iterative error analysis (IEA_DP) is proposed by utilizing cloud computing principles to efficiently process massive hyperspectral data. In particular, the storage of hyperspectral data is well organized to reduce the correlation among data partitions as well as to avoid data skew. The processing logic of IEA algorithm is optimized by reducing the intermediate data generated by each execution node and avoiding transitional large data. The newly developed method is implemented and evaluated on Spark and MapReduce model. Its efficiency is evaluated in terms of accuracy and parallel execution performance through the comparison with a serial IEA implementation on a single CPU.

2. Endmember Extraction Based on IEA

Let $\mathbf{X}_{N\times L} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T \in \mathbf{R}^{N\times L}$ denote a hyperspectral image with N pixels, where $\mathbf{x}_i \in \mathbf{R}^L$ is an L-dimensional hyperspectral pixel observation. The linear mixture model identifies a collection of spectrally pure constituent spectra (endmembers) and expresses the measured spectrum of a mixed pixel as a linear combination of the endmembers, weighted by fractional abundances that indicate the proportion of each endmember contained by the pixel [1]. This procedure can be described in mathematical terms as follows:

$$\mathbf{x}_i = \mathbf{U}\mathbf{f}_i + \mathbf{n}, \quad (1)$$

where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ denotes an L-by-m mixing matrix in which the endmembers correspond to the columns. This matrix is in general of full column rank. Here, m denotes the number of endmembers, $\mathbf{f}_i = [f_{i1}, f_{i2}, \ldots, f_{im}]^T$ denotes an m-by-1 vector containing the respective fractional abundances of the endmembers, f_{ik} is the abundance fraction of the kth endmember, with $k = 1, 2, \ldots, m$, and the notation $(\cdot)^T$ stands for vector transpose operation; $\mathbf{n}$ denotes an additive L-by-1 noise vector representing the errors that affect the measurement of the pixel at each spectral band. Endmember extraction of hyperspectral data aims at obtaining a good estimation of the mixing matrix $\mathbf{U}$. Several methods have been used to perform endmember extraction, including geometrical, statistical, and sparse regression-based approaches [1]. Among these methods, the IEA algorithm is one of the most successful algorithms of the first category and therefore has been widely used.

Assuming the existence of relatively pure pixels, the IEA algorithm performs a series of linear constrained unmixing [19] and chooses endmembers by minimizing the remaining error in the unmixed image [3]. This procedure is executed directly on the spectral data, without the requirement of transformation into Principal Components (PCs) or any other elimination of redundancy. A step-by-step description of the IEA algorithm is given as shown in Algorithm 1.

3. Processing Framework Based on Cloud Computing

In general, cloud computing uses MapReduce programming model, which is essentially a coarse granularity parallel programming model. MapReduce model can automatically parallelize the large-scale computing tasks. More importantly, the implementation details are transparent to the users. Users define the computation of map and reduce functions, and the underlying operating system automatically performs the parallel computation across large-scale clusters of machines and makes efficient use of network and disks by scheduling intermachine communication [20]. Hadoop, a cloud computing framework that uses MapReduce model, is well known for its fault tolerance and scalability [21]. However, Hadoop

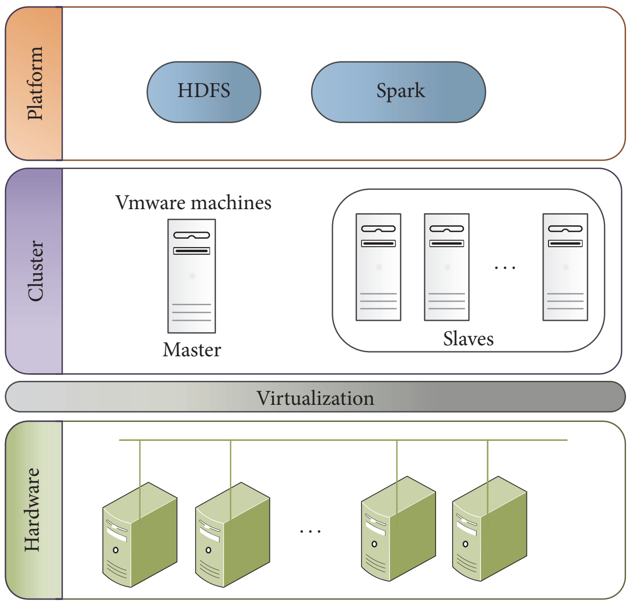

Figure 1: The processing framework based on cloud computing.

solutions rely on writing and reading data from HDFS and therefore are of slow speed.

Fortunately, Apache Spark, a novel high-performance framework capable of tackling massive data processing workloads while coping with larger and larger scales, has been proposed in [22]. This framework enables streaming and interactive queries and demonstrates its scalability, fault tolerance, and the ability of handling batch processing. Apache Spark is a cluster-computing platform, which is open source, Hadoop-compatible, fast, and expressive. In terms of data storage, Spark abstracts out of the distributed memory storage structure by Resilient Distributed Dataset (RDD) [23]. The RDD can control the data in the partition of different nodes and is compatible with HDFS. A large amount of existing data in HDFS can be loaded into RDD for being processed as a data source. Spark runs on top of existing HDFS infrastructures to provide enhanced and additional functionality. Moreover, Spark is based on memory computing, which holds intermediate results in memory rather than writing them to HDFS.

To be specific, the processing framework for distributed parallel endmember extraction of hyperspectral data based on cloud computing can be summarized graphically as shown in Figure 1.

4. Distributed Parallel Optimization for IEA Based on Spark

Spark is an extensible platform for data analysis that integrates the calculation of primitive memory. Therefore, Spark achieves better performance compared with Hadoop cluster storage methods. With the development of remote sensing technology, the data quantity of hyperspectral remote sensing data is increasing. Even a single pixel may contain hundreds of spectral information types, leading to more difficulty in the calculation of hyperspectral data processing. On the other hand, accelerating the processing of large amounts of data in hyperspectral imagery is of great importance. In this work, we present a distributed parallel implementation of IEA algorithm (IEA_DP) for endmember extraction of massive hyperspectral image based on Spark.

It can be observed that the most time-consuming parts of Algorithm 1 are the procedure of constrained unmixing, the calculation of error image, and the selection of the vectors with the largest error. Therefore, we concentrate on the parallel optimization of these parts. In what follows, we describe the distributed implementation of the different phases of Algorithm 1 and describe the architecture-level optimizations performed during the development of the parallel implementation.

Storage is a critical issue of our distributed parallel implementation. In hyperspectral remote sensing, with increasingly growing data volumes, it is important to efficiently store and utilize potentially unlimited amounts of hyperspectral datasets. HDFS represents a perfect choice for the reliability and elasticity of the task of storing very large files on different resources on large clusters. As a result, we store the original hyperspectral datasets on HDFS, taking advantage of its capabilities for distributed storage, fault tolerance, and flexibility in a transparent way.

A class (named `HSIInputFormat`) is defined to read original hyperspectral datasets from HDFS to `NewHadoopRDD` instances `ByteRDD`. In HDFS, the original hyperspectral image is divided into many spatial-domain partitions [16]. In order to reduce the I/O (input/output) overhead to the most extent, we read every data partition on HDFS as a `key-value` pair, in which the key (named `Offset`) is the offset of this partition in the original dataset and the value (named `Pixels`) is the hyperspectral data partition of byte type. Subsequently, `ByteRDD` is mapped onto a `MapPartitionsRDD` (which consists of formatted pixel vectors data, denoted as `DataRDD`). Finally, we cache `DataRDD` in RAM for fast access. The flowchart of the procedure for reading hyperspectral datasets is graphically illustrated in Figure 2.

Firstly, the `DataRDD` is mapped onto partitions $\mathbf{Pixel}_{p\times L}$, which are accumulated to $\mathbf{Sum}_{1\times L}$ by column. The `reduce` operation is then performed to aggregate theses $\mathbf{Sum}_{1\times L}$, and the mean vector $\mathbf{mean}_{1\times L}$ of the hyperspectral data $\mathbf{X}_{N\times L}$ is calculated on driver.

Secondly, the constrained unmixing is performed on $\mathbf{X}_{N\times L}$ using $\mathbf{mean}_{1\times L}$, as follows:

$$\mathbf{abu}_{1\times L}^{T} = \underbrace{\left(\mathbf{I} - \frac{\left(\mathbf{E}\mathbf{E}^{T}\right)^{-1}\mathbf{a}\mathbf{a}^{T}}{\mathbf{a}^{T}\left(\mathbf{E}\mathbf{E}^{T}\right)^{-1}\mathbf{a}}\right)\left(\mathbf{E}\mathbf{E}^{T}\right)^{-1}\mathbf{E}\mathbf{x}_{1\times L}^{T}}_{\mathbf{ta}} + \underbrace{\frac{\left(\mathbf{E}\mathbf{E}^{T}\right)^{-1}\mathbf{a}}{\mathbf{a}^{T}\left(\mathbf{E}\mathbf{E}^{T}\right)^{-1}\mathbf{a}}}_{\mathbf{tb}}, \quad (2)$$

where $\mathbf{I}$ is an N-order identity matrix, $\mathbf{a}$ is an N-dimensional column vector with all 1 entries, $\mathbf{E} = \mathbf{mean}_{1\times L}$, and $\mathbf{x}$ is a pixel vector.

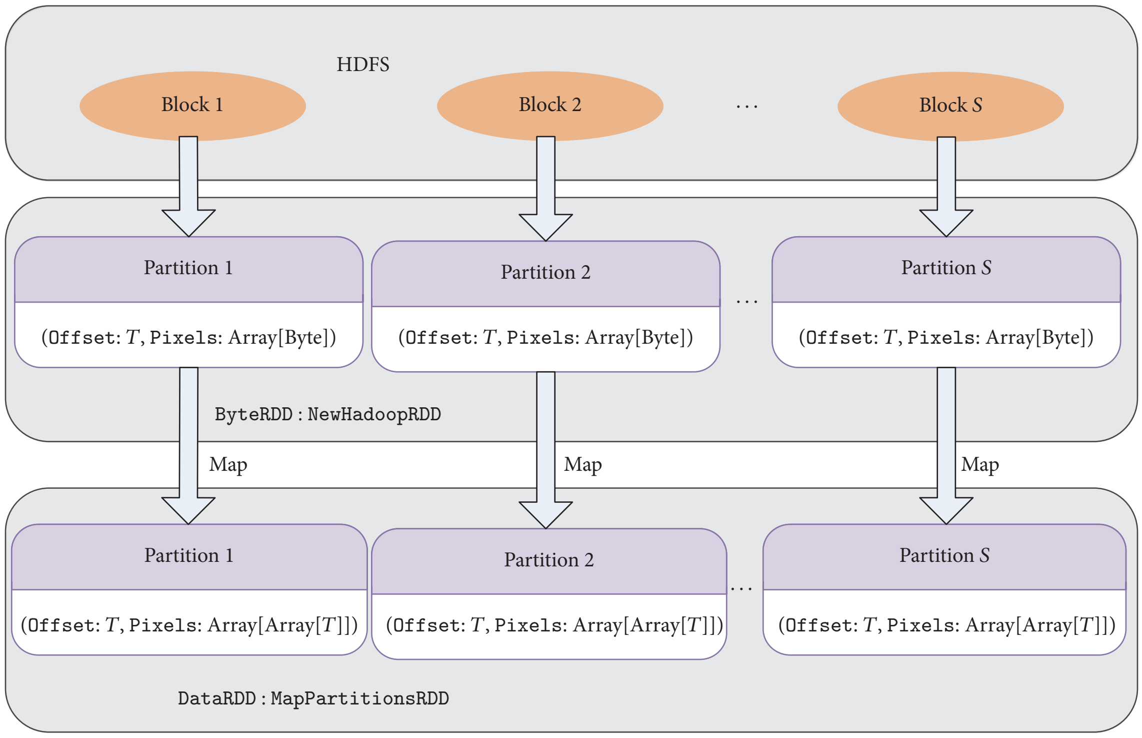
HDFS
Block 1
Block 2
Block S
Partition 1
Partition 2
Partition S
(Offset: T, Pixels: Array[Byte])
ByteRDD : NewHadoopRDD
Map
(Offset: T, Pixels: Array[Array[T]])
DataRDD : MapPartitionsRDD

Figure 2: Graphical illustration of the procedure used for reading hyperspectral datasets.

When computing (2), **ta** and **tb** parts are calculated, respectively, on driver. After that, **ta**, **tb,** and **E** are broadcasted to all workers. To avoid frequent garbage collection of Java virtual machine, pixels in the partition are processed successively by `map` operation. Take the pixel vector $\mathbf{Pixel}_i$ for example; its abundance coefficients are estimated by (2) and stored in vector $\mathbf{abu}_{1\times N}$. Then, its reconstruction error can be computed. When all the pixels in the partition have been processed, the maximum error *max*, as well as the corresponding position $pos_{\max}$, is selected. A tuple ($pos_{\max}$ + offset, max, $\mathbf{p}_{\max}$) is obtained as the output of `map` operation, where offset is the key of the certain partition that denotes the position of its first pixel in the whole hyperspectral image. The tuple with maximum max is afterwards selected by the `reduce` operation and is returned to driver.

In the next procedure, **ta**, **tb**, **E**, **pos**, and R are transmitted to every worker as broadcast variables. The `map` operations are performed on `DataRDD` to find the R pixels with the largest errors, and the subset of the set of R vectors is extracted, which consists of all those vectors which fall within the spectral angle θ of the maximum error vector. After the `reduce` operation is performed to merge sort, the vectors in the subset is averaged as $\mathbf{p}_i$ to decrease the effects of outliers and noise, and endmember matrix is updated ($\mathbf{U} = [\mathbf{U}; \mathbf{p}_1]$) on driver side.

The algorithmic details of IEA_DP are graphically illustrated in Figure 3. The storage of hyperspectral data is well organized to reduce the correlation among partitions as well as to avoid the data skew. Moreover, the logical procedure of IEA algorithm is optimized by reducing the intermediate data generated by every execution node and avoiding transitional large data. Accordingly, the computation of the IEA algorithm can be greatly accelerated.

5. Experimental Evaluation

To evaluate the proposed distributed parallel implementation of IEA algorithm, experiments were performed on a Spark equipped cluster with 1 master node and 8 slave nodes. Master is also the NameNode of HDFS and slaves are the DataNodes of HDFS. Master is a virtual machine created on the host with an Intel Xeon E5630 CPUs at 2.53 GHz with 8 cores by the VMware vSphere. The slave nodes are implemented by virtual machines created based on the virtualization of a 4-blade IBM Blade Center HX5 equipped with 2 Intel Xeon E7-4807 CPUs at 1.86 GHz and connected to a 12 TB disk array by SAS bus. Each slave is configured with 6 CPU cores. Master and slaves are all installed with Ubuntu 12.04, Hadoop 1.2.1, Spark 1.4.1, and Java 1.6.45. In addition, all nodes are connected by a gigabit switch. Figure 4 illustrates the architecture of the experimental platform.

The hyperspectral dataset used in our experiments is a subset of the well-known Airborne Visible Infrared Imaging Spectrometer (AVIRIS) Cuprite (http://aviris.jpl.nasa.gov/data/free_data.html) image with 224 spectral bands, which was collected over the Cuprite mining site, Nevada, in 1995. This scene has been widely used to validate the performance of endmember extraction algorithms. Some bands 1–3, 107–114, 159–169, and 221–224 have been removed prior to

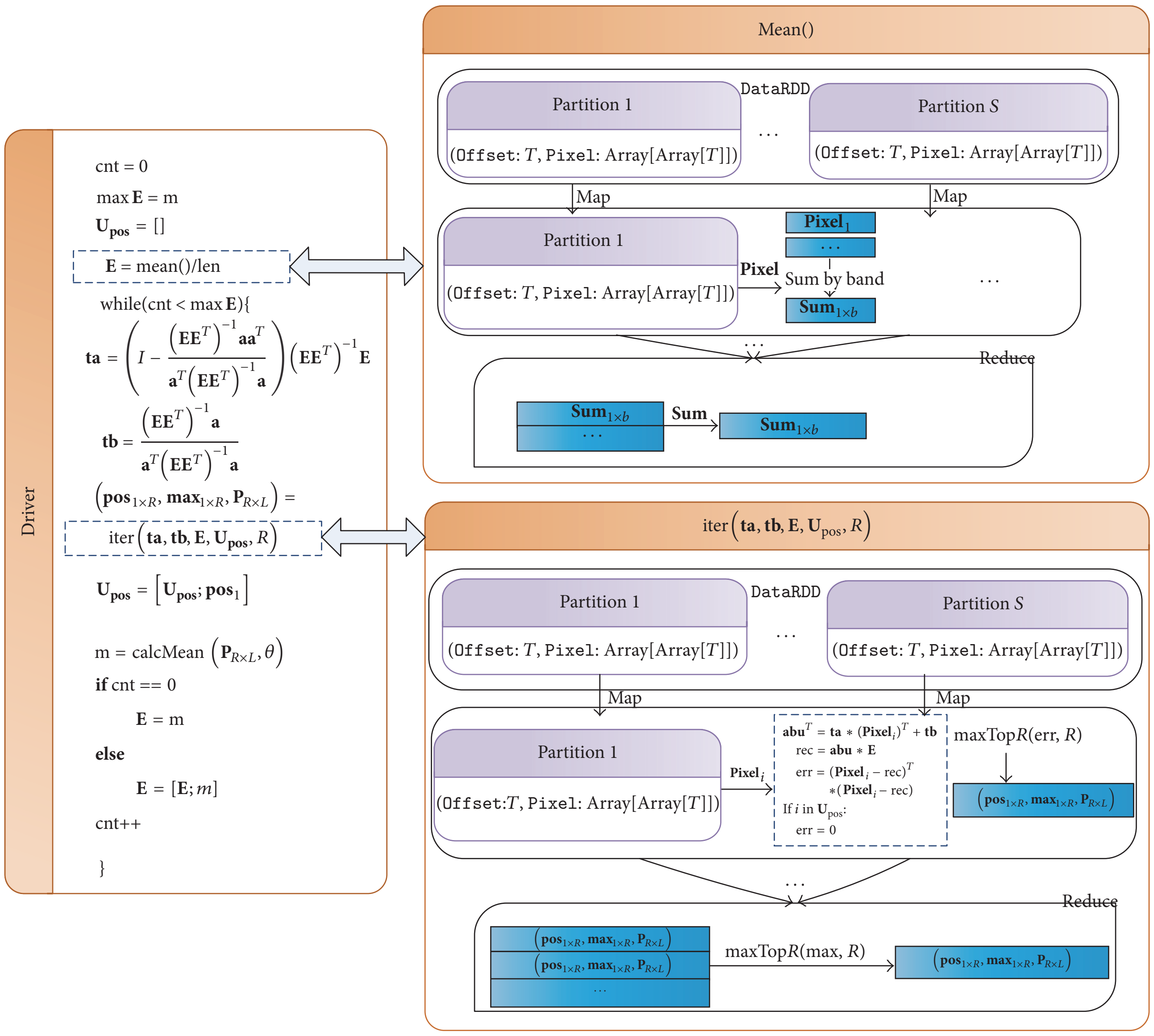

Figure 3: Design of IEA_DP algorithm based on Spark.

the analysis due to water absorption bands and bands with low SNR. The selected dataset consists of 350 × 350 pixels (as shown in Figure 5), 192 bands, and a total size of about 44.86 MB. In order to evaluate the algorithm's performance on data of different sizes, we use the Mosaicking function of ENVI software to generate 3 datasets with different sizes (including Dataset 1: 8400 × 350 with the size of about 1.05 GB, Dataset 2: 16800 × 350 with the size of about 2.10 GB, and Dataset 3: 33600 × 350 with the size of about 4.21 GB) by mosaicking the original 44.86 MB dataset. Both computational performance and unmixing accuracy have been taken into consideration for evaluation.

At the beginning, we evaluate the accuracy of the considered endmember extraction algorithm on the AVIRIS Cuprite image, taking advantage of the availability of detailed laboratory measurements of endmembers contained in the scene (http://speclab.cr.usgs.gov/maps.html). According to the survey results by the US Geological Survey (USGS), the Cuprite mining site mainly contains five categories of minerals, that is, Alunite, Buddingtonite, Calcite, Kaolinite, and Muscovite. Their reference ground signatures are available in the form of a USGS library (http://speclab.cr.usgs.gov/spectral-lib.html). The most similar endmember signatures extracted by the IEA algorithm are selected and compared with the available 5 reference USGS spectral signatures in terms of spectral angle distance (SAD), measured in radians. According to Table 1, it can be concluded that the serial and distributed parallel versions of IEA obtain identical results, while the extracted endmembers are very similar, spectrally, to the reference USGS spectra.

The most important aspect in this work is to what extent the distributed parallel implementation improves the endmember extraction procedure in terms of computational performance. Before reporting our performance evaluation,

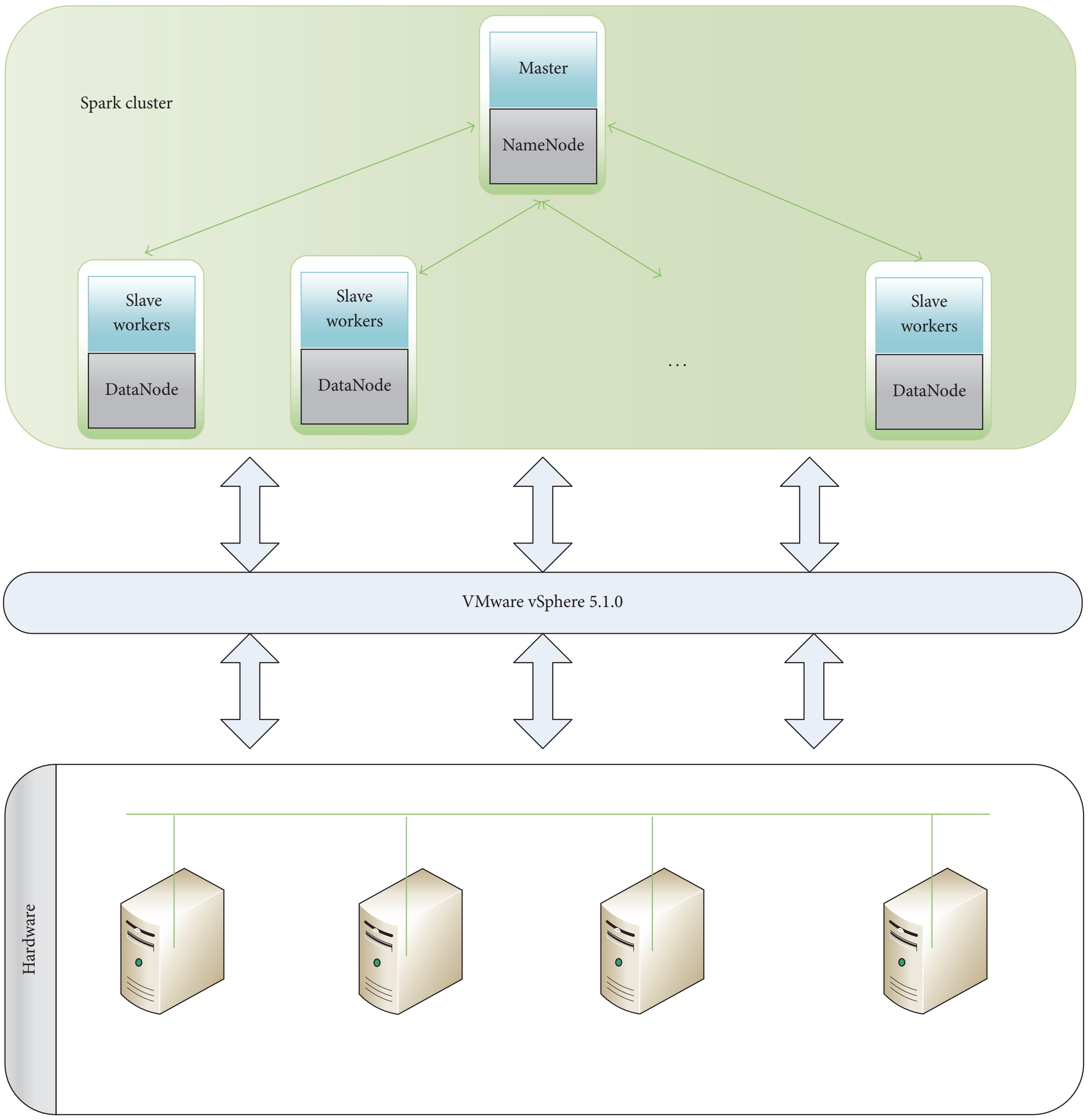

FIGURE 4: The architecture of the experimental platform.

it is worth emphasizing that our parallel implementation provides identical results as the serial version in terms of endmember spectra and fractional abundances. The key difference between the serial and parallel versions is the required runtime for completing the calculation. In subsequent part, we report the computational performance of the two versions executed on the datasets with different sizes, that is, Dataset 1, Dataset 2, and Dataset 3.

The execution time and speedups spent in processing Dataset 1 on the considered cloud computing architecture are listed in Table 2. In the first column of Table 2, "Parallel (x)" denotes the parallel version executed on a distributed platform consisting of 1 master node and x slave nodes (where $x = 1, 2, 4, 8$). As can be seen from Table 2, the execution time significantly decreases with regard to the increasing number of nodes. In terms of speedups, Figure 6 indicates approximately a linear growth with the number of nodes.

It is worth noting that the size of partition has great effects on the performance of the parallel versions. In other words, a good tuning of partition size helps improve the computational performance. In this paper, we empirically set the partition sizes (as Table 2 shows) to guarantee over 80%

Table 1: Spectral angle distance comparison between the endmembers extracted by IEA from the AVIRIS Cuprite scene and the reference USGS mineral spectral signatures.

Mineral	SAD (in radians)	
	Serial version	Parallel version
Alunite	0.0645	0.0645
Buddingtonite	0.0717	0.0717
Calcite	0.0898	0.0898
Kaolinite	0.0899	0.0899
Muscovite	0.0736	0.0736

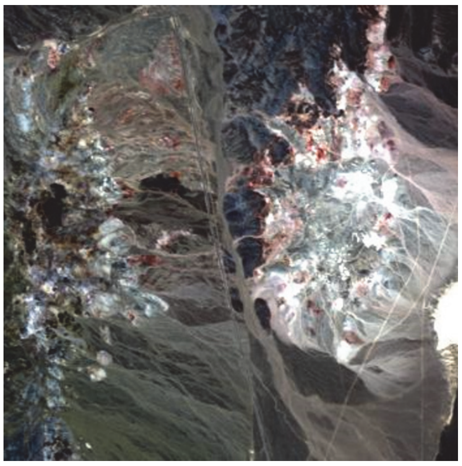
Figure 5: A 350 × 350-pixel subset of the AVIRIS Cuprite scene.

CPU utilization for every slave while running, thus leading to promising performance and speedup.

In a similar manner, experiments were performed on Dataset 2 and Dataset 3, using one master node and 8 slaves, to evaluate the efficiency of IEA_DP algorithm on larger datasets, as well as to compare IEA_DP algorithm with a Hadoop based IEA algorithm. It can be concluded from Figure 7 that the execution time of the proposed IEA_DP algorithm scales roughly linearly with the size of the dataset. Moreover, it is computationally efficient for large datasets (e.g., only 8.1%, and 6.2% of the time is consumed by initialization for the execution on Dataset 2 and Dataset 3, resp.). This conclusion is important, as it indicates the better scalability of the proposed IEA_DP as the data size increases. Moreover, Figure 8 demonstrates that the IEA_DP algorithm executed on the Spark platform processes the hyperspectral data much faster than on the Hadoop platform.

To summarize, the proposed IEA_DP performs and scales well under massive hyperspectral data. In particular, the availability of additional computing resources leads to more significant speedups. When using the cloud platform consisting of 1 master node and 8 slave nodes, the IEA_DP algorithm achieves a speedup of 33x in the hyperspectral endmember extraction. An endmember extraction task involving about 1 GB hyperspectral dataset, which took more than half an hour to be completed by the serial version, can now be completed in around 1.1 minutes by using our distributed parallel algorithm. This achievement is very promising for

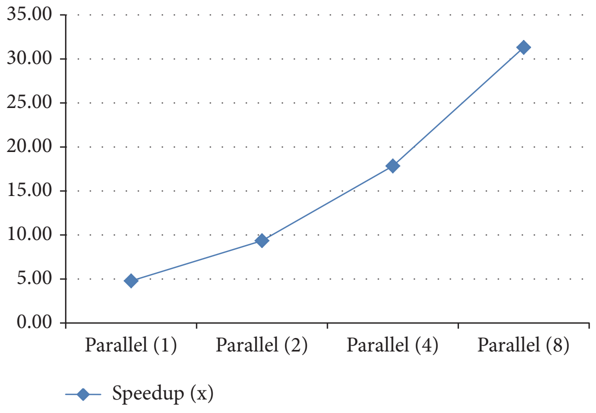

Figure 6: Speedup of the IEA_DP with Dataset 1.

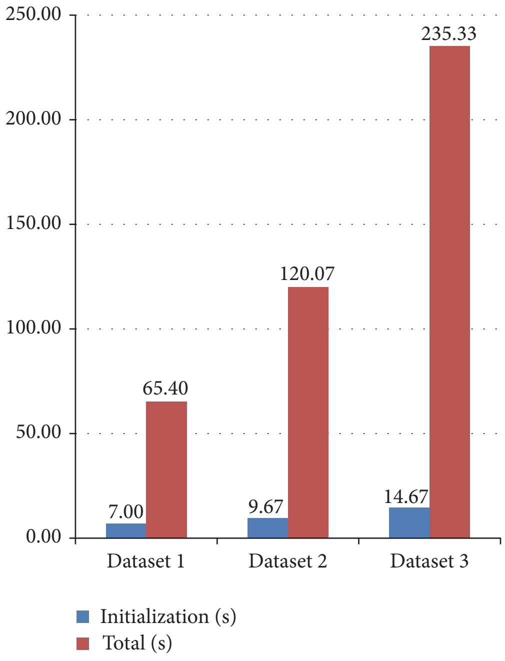

Figure 7: Execution time of the IEA_DP algorithm with different datasets.

a highly complex task, such as endmember extraction of a hyperspectral image with high volume and dimensionality.

6. Conclusions

The increased availability of high dimensional hyperspectral datasets is becoming an important challenge for hyperspectral image processing. This paper proposes a novel distributed parallel endmember extraction method based on iterative error analysis that utilizes advanced cloud computing technologies such as HDFS, Apache Spark, and the MapReduce model, to efficiently process massive hyperspectral data. Our experimental results indicate that the proposed method can be used to effectively process distributed collections of hyperspectral data of large scale. This contribution leads to the conclusion that hyperspectral image processing can greatly benefit from the efficient utilization of cloud computing

Table 2: Execution time of the serial and parallel versions of the IEA method with Dataset 1.

	Partition size (MB)	Initialization (s)	IEA (s)	Total (s)	Speedup (x)	Percentage of initialization
Serial	—	30.17	2017.68	2047.85	—	1.47%
Parallel (1)	269.17	12.33	415.80	428.13	4.78	2.88%
Parallel (2)	134.58	9.00	210.00	219.00	9.35	4.11%
Parallel (4)	67.29	8.67	106.13	114.80	17.84	7.55%
Parallel (8)	33.65	7.00	58.40	65.40	31.31	10.70%

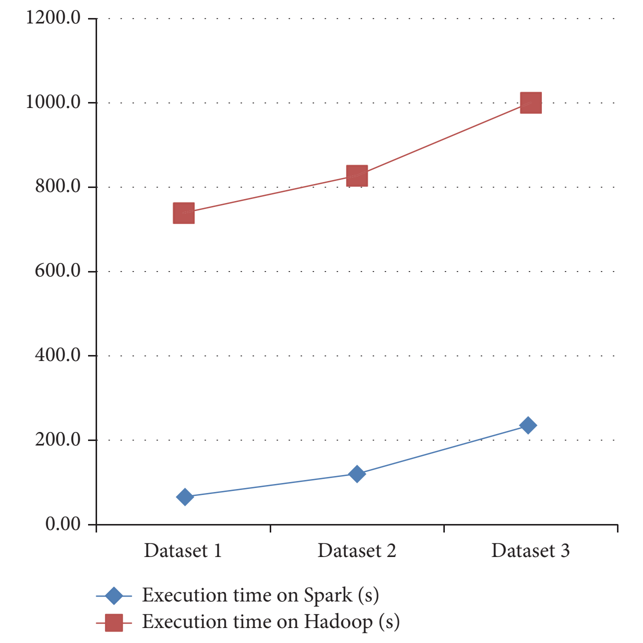

Figure 8: Execution time comparison between the Spark platform and the Hadoop platform.

architectures. Future work will focus on optimizing more complicated algorithms and applications for remotely sensed hyperspectral images.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

Financial support for this work, provided by the National Natural Science Foundation of China (Grant nos. 61471199, 91538108, and 11431015), the Research Fund of Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks under Grant no. WSNLBKF201507, the Jiangsu Province Six Top Talents Project of China under Grant no. WLW-011, is gratefully acknowledged.

References

[1] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon et al., “Hyperspectral unmixing overview: geometrical, statistical, and sparse regression-based approaches,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354–379, 2012.

[2] Z. Wu, S. Ye, J. Liu, L. Sun, and Z. Wei, “Sparse non-negative matrix factorization on GPUs for hyperspectral unmixing,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 8, pp. 3640–3649, 2014.

[3] R. A. Neville, K. Staenz, T. Szeredi et al., “Automatic endmember extraction from hyperspectral data for mineral exploration,” in *Proceeding of 21st Canadian Symposium Remote Sensing*, pp. 21–24, Ottawa, Canada, June 1999.

[4] J. M. P. Nascimento and J. M. B. Dias, “Does independent component analysis play a role in unmixing hyperspectral data?” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 1, pp. 175–187, 2005.

[5] J. M. P. Nascimento and J. M. Bioucas-Dias, “Hyperspectral unmixing algorithm via dependent component analysis,” in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS ’07)*, pp. 4033–4036, Barcelona, Spain, June 2007.

[6] J. M. P. Nascimento and J. M. B. Dias, “Vertex component analysis: a fast algorithm to unmix hyperspectral data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 4, pp. 898–910, 2005.

[7] C.-I. Chang, C.-C. Wu, W.-M. Liu, and Y.-C. Ouyang, “A new growing method for simplex-based endmember extraction algorithm,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2804–2819, 2006.

[8] J. Li and J. B. Dias, “Minimum volume simplex analysis: a fast algorithm to unmixhyperspectral data,” in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pp. 250–253, Boston, Mass, USA, 2008.

[9] A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, “High performance computing for hyperspectral remote sensing,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 528–544, 2011.

[10] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, “Recent developments in high performance computing for remote sensing: a review,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 508–527, 2011.

[11] C. González, D. Mozos, J. Resano, and A. Plaza, “FPGA implementation of the N-FINDR algorithm for remotely sensed hyperspectral image analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 2, pp. 374–388, 2012.

[12] A. Remón, S. Sánchez, A. Paz, E. S. Quintana-Ortí, and A. Plaza, “Real-time endmember extraction on multi-core processors,” *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 5, pp. 924–928, 2011.

[13] S. Bernabe, S. Sanchez, A. Plaza, S. Lopez, J. A. Benediktsson, and R. Sarmiento, “Hyperspectral unmixing on GPUs and multi-core processors: a comparison,” *IEEE Journal of Selected*

Topics in Applied Earth Observations and Remote Sensing, vol. 6, no. 3, pp. 1386–1398, 2013.

[14] A. Barberis, G. Danese, F. Leporati, A. Plaza, and E. Torti, "Real-time implementation of the vertex component analysis algorithm on GPUs," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 2, pp. 251–255, 2013.

[15] J. M. P. Nascimento, J. M. Bioucas-Dias, J. M. Rodriguez Alves, V. Silva, and A. Plaza, "Parallel hyperspectral unmixing on GPUs," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 3, pp. 666–670, 2014.

[16] Z. Wu, Y. Li, A. Plaza, J. Li, F. Xiao, and Z. Wei, "Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 6, pp. 2270–2278, 2016.

[17] Z. Chen, N. Chen, C. Yang, and L. Di, "Cloud computing enabled web processing service for earth observation data processing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 6, pp. 1637–1649, 2012.

[18] K. Stanoevska-Slabeva, T. Wozniak, and S. Ristol, *Grid and Cloud Computing: A Business Perspective on Technology and Applications*, Springer, Heidelberg, Germany, 2010.

[19] D. C. Heinz and C.-I. Chang, "Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 3, pp. 529–545, 2001.

[20] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[21] D. Borthakur, "HDFS Architecture Guide," 2008, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf.

[22] M. Zaharia, M. Chowdhury, T. Das et al., "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI '12)*, p. 2, USENIX Association, San Jose, Calif, USA, April 2012.

[23] M. Zaharia, "An architecture for fast and general data processing on large clusters," Tech. Rep. UCB/EECS-2014-12, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2014.