Chapter 5

Performance-Aware High-Performance Computing for Remote Sensing Big Data Analytics

Mustafa Kemal Pektürk and Muhammet Ünal

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75934

Abstract

The incredible increase in the volume of data emerging along with recent technological developments has made the analysis processes which use traditional approaches more difficult for many organizations. Especially applications involving subjects that require timely processing and big data such as satellite imagery, sensor data, bank operations, web servers, and social networks require efficient mechanisms for collecting, storing, processing, and analyzing these data. At this point, big data analytics, which contains data mining, machine learning, statistics, and similar techniques, comes to the help of organizations for end-to-end managing of the data. In this chapter, we introduce a novel high-performance computing system on the geo-distributed private cloud for remote sensing applications, which takes advantages of network topology, exploits utilization and workloads of CPU, storage, and memory resources in a distributed fashion, and optimizes resource allocation for realizing big data analytics efficiently.

Keywords: big data analytics, high-performance computing, real-time analytics, remote sensing, distributed computing, geo-distributed cloud, resource allocation

1. Introduction

The extreme increase in the amount of data produced daily by many organizations reveals big challenges in data storage and extracting information from timely data [1–3]. Many sensors designed in today's technology are used in observation centers and on the Earth to create a continuous stream of data [4]. Real-time, near-real-time geospatial data must be analyzed in a short time in order to be able to provide time-critical decision support in time-critical applications [5]. The development of efficient computing techniques for obtaining information from remote sensing (RS) big data is critical for Earth science [6, 7]. In particular, the recent developments in

remote sensing technologies have had a tremendous increase in remote sensor data [8]. The amount of remote sensing (RS) data collected from a single satellite data center has dramatically increased and has reached several terabyte values per day [9]. This is because sensors have high resolution and a large amount of band due to new camera technologies. Thus, RS data reaching high dimensions is defined as "Big data."

Large image files which consist of both voluminous pixel and multiple spectral bands (multi-spectral/hyperspectral) cause great difficulties to read and store in memory. Besides this, data mining algorithms which extract information from satellite and remote sensing data involve high computational complexity. With these features, remote sensor applications are both data intensive [10, 11] and compute intensive [12]. However, the computational complexity of many data mining algorithms is super linear to the number of samples and the size of the sample. Hence, optimization of algorithms is not enough to obtain better performance when those two variables continue to increase.

When talking about big data analysis, the main difficulties in computer architecture are CPU intensiveness and slow input/output (I/O) operations. According to Moore's Law, CPU and driver performance doubles every 18 months. On the other hand, when the trends in I/O interfaces are examined, the improvement is near-to-network speed improvement but still behind of it [13]. Although I/O interfaces operate at the same speed as the processor bus, I/O lags behind because peripheral hardware cards operate at low speeds. PCI cards are being used to increase I/O performance. Thus I/O performance increases by about 33% annually. In spite of these improvements in traditional approaches, when collected data exponentially increases, usage of relatively slow data processing techniques makes real-time analysis difficult particularly [2]. For this reason, the most important factor determining the performance of analytical processes is the limited structure of hardware resources inherently [14]. Therefore, modern technologies and high-performance computing (HPC) techniques, which have parallel processing capabilities such as multi-core programming, GPU programming, field programmable gate arrays (FPGA), cluster computer, and cloud computing are needed to perform analysis on large volumes of data, which is complex and time-consuming to extract information [14–16].

The HPC system usage attracts more attention in remote sensing applications because of a great deal of data recently [6, 7]. HPC integrates some of the computing environments and programming techniques to solve large-scale problems in the remote sensing era. Many applications of remote sensing such as environmental studies, military applications, tracking and monitoring of hazards, and so on require real-time or near-real-time processing capabilities for urgent intervention that is timely in the necessary situation. HPC systems such as multi- or many-integrated core, GPU/GPGPU, FPGA, cluster, and cloud have become inevitable to meet this requirement. Multi-core, GPU, FPGA, and so on technologies meet parallel computing needs for onboard image processing particularly. In such technologies, developed algorithms run on a single node only but with multiple cores. So they could just scale vertically with hardware upgrades—for example, more CPUs, better graphics cards, more memories [17]. When distributed computing is considered, the most conceivable technology is commercial off-the-shelf (COTS) computer equipment, which is called cluster [6, 7]. In this approach, a

http://dx.doi.org/10.5772/intechopen.75934

cluster is created from a number of computers to work together as a team [18]. These parallel systems, installed with a large number of CPUs, provide good results for both real-time and near-real-time applications that use both remote sensing and data streams, but these systems are both expensive, and scalability cannot exceed a certain capacity. Although much of parallel systems are homogeneous inherently, the recent trend in HPC systems is the use of heterogeneous computing resources, where the heterogeneity is generally the result of technological advancement in progress of time. With increasing heterogeneity, cloud computing has emerged as a technology which aimed at facilitating heterogeneous and distributed computing platforms. Cloud computing is an important choice for efficient distribution and management of big datasets that cannot be stored in a commodity computer's memory solitarily.

Not only the increasing data volume but also the difficulty of indexing, searching, and transferring the data exponentially increases depending on data explosion [19, 20]. Effective data storage, management, distribution, visualization, and especially multi-modal processing in real/near-real-time applications are challenged as open issues for RS [21].

2. Big data

Big data is characterized as 3 V by many studies: volume, velocity, and variety [22, 23]. Volume is the most important big data quality that expresses the size of the dataset. The velocity indicates the rate of production of big data, but the increase in the rate of production also reveals the need for faster processing of the data. Variety refers to the diversity of different sources of data.

Given the variety of the data, the majority of the data obtained is unstructured or semi-structured [21]. Considering the velocity of data, velocity requirements vary according to application areas. In general, velocity is addressed under the heading of processing at a specific time interval, such as batch processing, near-real-time requirement, continuous input–output requirement real time, and stream processing requirements. Application of critical and live analytics-based batches to improve data and analysis processes requires continuous and real-time analysis, and critical applications require immediate intervention, depending on the analysis of incoming data streams.

2.1. Remote sensing data

Remote sensing (RS) is defined as the ability to measure the quality of a surface or object from a distance [9]. RS data are obtained from various data acquisition technologies (lidar, hyperspectral camera, etc.) in airplanes and unmanned aerial vehicles. With the recent developments in RS technologies, the amount, production rate, and diversity of remote sensor data have increased exponentially (**Table 1**). Thus, the data received from the remote sensors are being treated as RS "Big Data."

Diversity and multidimensionality are the greatest factors in the complexity of RS big data. RS data is used in a variety of geosciences with environmental monitoring, underground, atmospheric, hydrological, and oceanographic content. Due to such different and wide range of

Satellites	Velocity	Volumes 1	Volumes 2
	(Mbps)	(GB/Day)	(TB/Year)
HJ-1B	60	57	20.32
HJ-1A	120	114	40.63
ZY-03	900	498.38	176.22
HJ-1C	320	187.5	66.83
ZY-02C	320.00	175.78	62.66
SPOT-4	50.00	36434	33664
LANDSAT5	85.00	43159	36404
RADASAT-2	105.00	57.68	20.56
RADASAT-1	105.00	57.68	20.56
SPOT-5	100.00	54.93	19.58
ENVISAT	100.00	32.96	27699
IRS-P6	210.00	46.14	16.45
LANDSAT8	440.00	241.70	86.15
Total	3712.98	2089.06	574.6

Table 1. Satellite data centers, data rates, and volumes.

application areas, the diversity of RS data has increased greatly. There are approximately 7000 different types of RS datasets in NASA archives, as far as is known [9]. Numerous satellites and sensors with different resolutions have emerged due to higher spatial resolution, temporal resolution, and even spectral resolution. As remote sensing data continues to increase and complex, a new architecture has become a necessity for existing algorithms and data management [24].

2.2. Remote sensing algorithms

The processing of RS data has an important role in Earth observation systems. The longest RS workflow starts with data acquisition and ends with a thematic application (**Figure 1**). There can be processes which are processed sequentially or concurrently in each step within the workflow. An RS application typically consists of the following stages, respectively [9]:

- Data Acquisition/Collection: Images obtained from satellite or hyperspectral cameras are captured on the downlink channel. With high spatial and temporal resolution, data volume is increasing.
- Data Transmission: The data stream obtained is sent simultaneously to the satellite center. The high bandwidth of the data transfer is critical for applications requiring real-time processing. In reality, it is impossible to transmit real-time data because the data increase due to satellite and camera technologies is faster than the data transmission rate.
- Preprocessing: Preprocessing related to image quality and geographical accuracy such as decomposition, radiometric verification, and geometric verification is performed.

http://dx.doi.org/10.5772/intechopen.75934

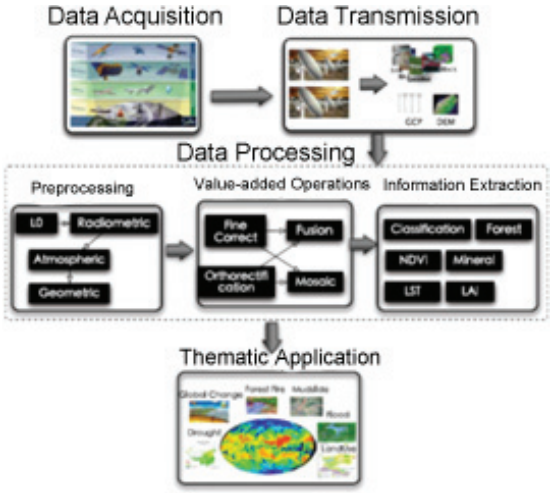

Figure 1. Remote sensing data-processing steps [9].

- Value-added Operations: Value-added operations such as fine-tuning, orthorectification, fusion, and mosaic are performed. It is possible to say that the mosaic process has a longer processing time than the others.
- Information Extraction/Thematic Implementation: In this step, classification, attribute extraction, and quantitative deductions are obtained with images subjected to preprocessing and value-added operations, for example, extracting information such as leaf area index, surface temperature, and snowy area on RS images.

2.3. Data access pattern in remote sensing algorithm

The data access patterns in the remote sensing algorithms vary according to the characteristics of the algorithms. There are four different access patterns for RS data. Depending on the size of the RS data, the unavoidable I/O load and irregular data access patterns make inapplicable the traditional cluster-based parallel I/O systems [25].

"Sequential row access pattern" is used by algorithms such as pixel-based processing-based radiometric verification, support vector machine (SVM) classifier. When algorithms are implemented in parallel, each processor needs logically multiple consecutive image rows.

"Rectangular block access pattern" is used by algorithms such as convolution filter and resample which require neighbor-based processing. Non-contiguous I/O patterns are visible in these algorithms and are not efficiently supported by normal parallel file systems.

"Cross-file access pattern" is used by algorithms like fusion and normalized difference vegetation index (NDVI) that require inter-band calculations. This data consists of small and non-contiguous

fragments in hundreds of image files. Thus, in this type of access, a large number of read/write operations take place, which is a time-consuming process.

"Irregular access pattern" is used by algorithms such as fast fourier transform (FFT), image distortion, and information extraction, which require scattered access or the entire image. These algorithms use diagonal and polygonal access patterns as irregular access. In these patterns, different sized parts can be in different nodes, even though they are small and non-contiguous pieces of data. In addition to I/O difficulties, the problem of identifying irregular data areas also arises.

3. Real-time big data architecture

Big data architects often need a distributed system structure for data analysis, which requires data storage. S. Tehranian et al. proposed an architectural model that provides performance, reliability, and scalability, consisting of candidate hardware and software for distributed real-time processing of satellite data at ground stations [26]. The resulting prototype system was implemented using the open source adaptive communication environment (ACE) framework and C ++ and tested on the cluster; real-time systems have achieved significant performance without sacrificing reliability and high availability. Structures and mechanisms for parallel programming are needed so that RS data can be analyzed in a distributed manner. In this context, Y. Ma et al. proposed a generic parallel programming framework for RS applications on high-performance clusters [27]. The proposed mechanism has programming templates that provide both distributed and generic parallel programming frameworks for RS algorithms. The proposed magHD for storage, analysis, and visualization of multidimensional data combines Hadoop and MapReduce technologies with various indexing techniques for use on clusters [28]. Some systems, such as [29], contain all of the different steps of addition, filtering, load balancing, processing, combining, and interpreting. In the related work, a real-time approach to continuous feature extraction and detection aimed at finding rivers, land, and rail from satellite images was proposed using the Hadoop ecosystem and MapReduce.

At a minimum, the components that the big data architecture should have in order to be able to perform real-time analysis are as follows: user interface, distributed file system, distributed database, high-performance computing (**Figure 2**).

A distributed file system is a virtual file system that allows distributed data to be stored on multiple computer clusters. In clusters that may consist of heterogeneous nodes, the application provides a common interface for accessing the data in isolation from the operating and file systems.

A distributed database consists of separate databases on each node/server in a multi-computer cluster connected by a computer network. The distributed database management system allows distribution and management of data on the server.

High-performance computing systems are technologies that provide an infrastructure that provides a sufficiently fast computing environment for parallel analysis of big data. It is crucial for the system to respond to the user within the reasonable time.

http://dx.doi.org/10.5772/intechopen.75934

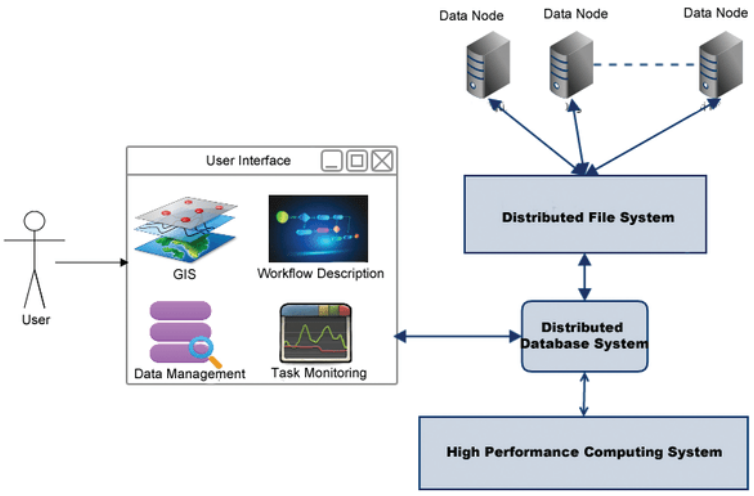

Figure 2. Generalized real-time big data architecture.

The user interface is a component that basically allows the user to query the server-based application through a visual interface, query, load, delete, update, request analysis, define workflow. The user interface has not been investigated as a major concern in this chapter.

3.1. Storage

3.1.1. Distributed file system

SSD and PCM devices that are being used instead of HDD as data storage are far from the I/O performance required for big data. Enterprise storage architectures, such as DAS, NAS, and SAN, have drawbacks and limitations when used as distributed systems [2].

Distributed File System (DFS) is a virtual file system that spans multiple nodes on the cloud [15]. It provides the abstraction of heterogeneity of data nodes in different centers. Thus, the distributed file system provides a common interface for applications to access data on heterogeneous nodes that use different operating systems and different file systems on different nodes individually. There are some capabilities that a distributed file system should generally provide. Location transparency is where the application can access data such as being held locally without actually having to hold it. Access transparency is a common interface for access to data independent of the operating system and the file system. Fault tolerance is the ability to keep a replica of a replica on more than one node so that in the event of an error the replica is preserved in the nodes holding the replica. Scalability means that the number of nodes the file system is running on can be increased to the required amount (without the error system hanging down) if needed.

The Hadoop distributed file system (HDFS) is an open-source distributed file system distributed with an Apache license (**Figure 3**). HDFS is designed especially for big datasets and high

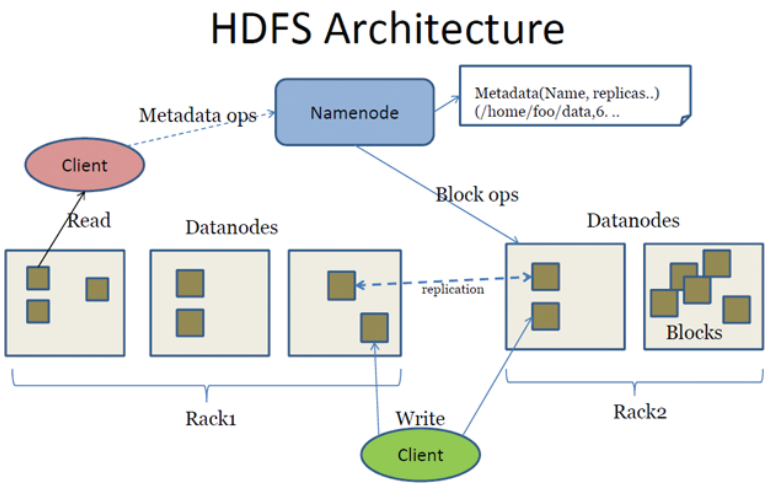

Figure 3. HDFS architecture.

availability, apart from the common abilities. It is also platform independent as it is implemented in Java. Applications are accessed via the HDFS API, which is maintained by any filing system. Thus, file access is isolated from local file systems. Compared to other distributed file systems (IRODS, Luster), it is stated that the performance is different in design and HDFS is the only DFS with automatic load balancing [15]. At the same time, because it is platform independent and the availability of MapReduce support makes it easy to use on many systems, it is the preferred choice.

The Google file system (GFS) is a proprietary distributed file system developed by Google for its own use [30]. The reason for the development is the need for a scalable distributed file system that emerges in big data-intensive applications. It is designed to enable reliable, efficient, and fault-tolerant use of data in a multitude of thousands of drives and machines, each with thousands of simultaneous users.

Storage systems such as amazon simple storage (S3), nirvanix cloud storage, openstack swift, and windows azure blob that are used in cloud systems do not fully meet the scalability and replication needs of cloud applications and the concurrency and performance requirements of analysis applications.

General parallel file system (GPFS) is a high-performance clustered file system developed by IBM. GPFS can be built on shared drives or shared-nothing distributed parallel nodes. Since it fully supports the POSIX-based file system, it removes the need to learn the new API set introduced by other storage systems. On the other hand, HDFS and GFS are not completely POSIX compliant and require new API definitions to provide analysis solutions in the cloud. In the study conducted by Schmuck et al., it is stated that GPFS is in terms of file-reading performance of HDFS with a meta-block concept [31]. A meta block is a set of consecutive

http://dx.doi.org/10.5772/intechopen.75934

data blocks located on the same disk. In the proposed approach, a trade-off was attempted between different block sizes.

Parallel virtual file system (PVFS) compared with HDFS [32] shows that PVFS has not shown a significant improvement in terms of completion time and throughput.

Rasdaman database is a database that supports large multidimensional arrays that conventional databases cannot handle and can store large remote sensing data by nature [33]. The architecture of Rasdaman is based on the sequence shredding process called "tiling." The Rasdaman parallel server architecture feature provides a scalable and distributed environment for efficient processing of large numbers of concurrent user requests. Thus it is possible to present distributed datasets over the web. In order to retrieve and process the dataset from Rasdaman, queries of data retrieval in the query language defined by open geospatial consortium's (OGC) web coverage processing service (WCPS) standards should be run. The PetaScope component, developed as a Java Servlet used at this point, provides queries for multidimensional data retrieval, retrieval filtering, and processing by implementing OGC standard interfaces. It also adds support for geographic and temporal coordinate systems.

Depending on the size of the RS data in the remote sensing applications, the unavoidable I/O load and irregular data access patterns are not applicable to traditional cluster-based parallel I/O systems [25]. In the study conducted by L. Wang et al., an RS file-based parallel file system for remote sensing applications was proposed and implemented using the OrangeFS file system. By providing an application-specific data placement policy, efficiency is achieved for different data access patterns. The improvement in the performance of the proposed system is seen as an average of 20%.

3.1.2. Distributed database

The classical approaches used in managing structured data have a schema for data storage and a relational database for retrieving data. Existing database management tools have been inadequate for processing large volumes that grow rapidly and become complex. Data warehouse and data-market approaches have gained popularity in systems with more than one structured data [2]. One of these approaches is the data warehouse, which is used to store, analyze, and report results to the user. The data market (March) approach is an approach that improves data access and analysis based on the data warehouse. The enterprise data warehouse (EDW), which is favored by large organizations, allows the data processing and analysis capability to be used on a very large and unified enterprise database [21]. Some cloud providers can offer a petabyte data and more scaling solution with EDW. For example, Amazon Redshift uses a massively parallel processing (MPP) architecture consisting of a large number of processors for high-performance interrogation, with columnar storage and data compression. In addition, the amount of I/O required by queries is reduced using local attached storage and zone maps.

For storing and managing unstructured or non-relational data, the NoSQL approach is divided into two independent parts: data storage and management [2]. With the key-value storage

model in storage, NoSQL's focal point is scalability and high performance of data storage. In the management section, data management tasks can be performed at the application layer through the lower-level access mechanism. The most important features of the NoSQL database are the ability to quickly change the data structure by providing schema freedom and the need to rewrite the data so that the structured data can be stored heterogeneously, providing flexibility. The most popular NoSQL database is the Cassandra database, which was first used by Facebook and published as open source in 2008. There are also NoSQL implementations such as SimpleDB, Google BigTable, MongoDB, and Voldemort. Social networking applications such as Twitter, LinkedIn, and Netflix also benefited from NoSQL capabilities.

According to the method proposed by L. Wang et al. for the management problem of conventional remote sensing data, the image data is divided into blocks based on the GeoSOT global discrete grid system and the data blocks are stored in HBase [34]. In this method, the data is first recorded in the MetaDataInfo table. The satellite-sensor acquisition time is used as the row ID. In the DataGridBlock table, the row ID is kept with the MetaDataInfo row ID as well as the geographic coordinate. HBase tables ensure that blocks that are geographically close to the ascending order of row numbers will be held in adjacent rows in the table. When a spatial query arrives, the GeoSOT codes are first calculated and the DataGridBlock table is filtered by these codes. In addition, a distributed processing method that uses MapReduce model to deal with image data blocks is also designed. When MapReduce starts the job, it splits the table into bounds of regions, each region containing a set of image data blocks. The map function then processes each data block from the region and sends the resulting results to the reduce function.

The analysis of ultra-big databases has attracted many researchers' interest, as traditional databases are inefficient for storing and analyzing large digital data. Apache HBase, the NoSQL distributed database developed on HDFS, is one of the results of these researches. A study by M.N. Vora evaluated a hybrid approach in which HDFS retains data such as non-textual images and HBase retains these data [35]. This hybrid architecture makes it possible to search and retrieve data faster.

3.2. HPC systems

When data analysis is considered, the most important difficulty is scalability, depending on the volume of data. In recent years, researchers have focused more on accelerating their analysis algorithms. However, the amount of data is much faster than CPU speed. This has led processors to come to a position to support parallel computing as multi-core. Timeliness for real-time applications comes first. Thus, many difficulties arise not only in hardware development but also in the direction of development of software architects. The most important trend at this point is to make distributed computing improvements using cloud computing technology.

The technologies used in remote sensing applications have difficulties in delivering, processing, and responding in time [36]. Web technologies, grid computing, data mining, and parallel computation on remote sensing data generated by R. Patrick and J. Karpjoo have been scanned. The size of the data volume, the data formats, and the download time are general difficulties.

http://dx.doi.org/10.5772/intechopen.75934

With the combination of betting technologies, the processing time in some applications can be reduced to as short a time as can be decided by the helpers in a timely manner. Although it is reasonable to process the remote sensing data as soon as possible, it does not seem possible to perform real-time processing automatically.

3.2.1. Onboard architecture

3.2.1.1. Multi-core processor

The multi-core processor is an integrated circuit which has two and more processors to process multiple tasks efficiently. After the frequency of processors reached limits due to the heating problem, processing capabilities of new CPUs continue to increase by multiplying the number of cores [37]. Recently new CPUs could handle 8–12 simultaneous threads. To benefit from the multi-core CPUs, the problem should be divided into partitions which can be processed simultaneously. Multi-core programming is needed to achieve this benefit and rewriting of application is needed affordably. Multi-core programming is the implementation of algorithms using a multi-core processor on a single computer to improve performance. Some APIs and standards such as OpenMP and MPI are needed to implement algorithms which could be run simultaneously on multi-core processors.

3.2.1.2. Graphic processing units

GPU is a specialized circuit dedicated to graphical processing preliminarily. After it has had a brilliant rise in manipulating computer graphics and image processing in recent years, this technology gets used for developing parallel algorithms on RS image data widely [38–40]. RS complex algorithms should be rewritten to benefit from GPU parallelism by using thousands of simultaneous threads.

3.2.1.3. Field programmable gate arrays

Some onboard remote sensing data processing scenarios require components that can operate with low weight and low power, especially in systems where air vehicles such as unmanned air vehicle and satellite are used. While these components reduce the amount of payload, they can produce real/near-real-time analysis results at the same time as data is being obtained from the sensor. For this purpose, programmable hardware devices such as FPGAs can be used [41, 42]. FPGAs are the digital integrated circuits which consist of an array of programmable logic blocks and reconfigurable interconnects that allow the blocks to be connected simply. But the need for FPGA programming and learning a new set of APIs is emerging.

3.2.2. Distributed architecture

3.2.2.1. Cluster

One of the most used approaches when considering hardware-based improvements is commercial off-the-shelf (COTS)-based computer-based solutions. In this approach, a cluster is created from a number of computers to work together as a team [43]. These parallel systems, installed with a large number of CPUs, provide good results for both real-time and near-real-time

applications using both remote sensors and data streams, but these systems are both expensive, and scalability does not exceed a certain capacity.

Cavallaro et al. have addressed the classification of land cover types over an image-based dataset as a concrete big data problem in their work [44]. In the scope of the study, PiSVM, an implementation based on LibSVM, was used for classification. The PiSVM code is stale and stable despite the I/O limits. While PiSVM is used in parallel, MPI is used for communication on multiple nodes. For the parallel analysis, the JUDGE cluster in Jülich Supercomputing Center in Germany was used. The training period has been reduced significantly in the PiSVM, which runs parallel to the running of the series MATLAB. In parallel operation, the accuracy of SVM remains the same as in serial operation (97%).

3.2.2.2. Cloud

Cloud computing is one of the most powerful big data techniques [45]. The ability to provide flexible processing, memory, and drivers by virtualizing computing resources on a physical computer made the supercomputing concept more affordable and easily accessible [46]. The use of the cloud concept, which provides a multi-computer infrastructure for data management and analysis, provides great ease in terms of high scalability and usability, fault tolerance, and performance. Especially considering the critical applications that need to extract information from the data that the next-generation remote sensors can produce near real time, it is very important to use cloud computing technologies for high-performance computing [21]. In addition, the cloud computing infrastructure has the ability to create an efficient platform for the storage of big data as well as for the performance of the analysis process. Thus, together with the use of this technology, expensive computing hardware such as cluster systems, allocated space and software requirements can be eliminated [22].

The Hadoop ecosystem has emerged as one of the most successful infrastructures for cloud and big data analysis [23, 45]. The platform brings together several tools for various purposes, with two major services: HDFS, a distributed file system, and MapReduce, a high-performance parallel-data processing engine. The MapReduce model is an open-source implementation of the Apache Hadoop framework. This model allows big datasets to be distributed concurrently on multiple computers. Remote sensing applications using the MapReduce model have become a research topic in order to improve the performance of the analysis process as a result of exponential growth within the latest developments in sensor technologies [15, 16]. The processing services on the cloud are accessed via the distributed file system. In order to reduce data access, it is reasonable to process the data in the central computer where the data is stored.

4. Performance-aware HPC

Processing of big geospatial data is vital for time-critical applications such as natural disasters, climate changes, and military/national security systems. Its challenges are related to not only massive data volume but also intrinsic complexity and high dimensions of the geospatial datasets [47]. Hadoop and similar technologies have attracted increasingly in geosciences

communities for handling big geospatial data. Many investigations were carried out for adopting those technologies to processing big geospatial data, but there are very few studies for optimizing the computing resources to handle the dynamic geo-processing workload efficiently.

In existing software systems, computing is seen as the most expensive part. After daily collected data amount is grown exponentially with recent technologies, data movement has a deep impact on performance with a bigger cost than computing which is cheap and massively parallel [48]. At that point, new high-performance systems need to update themselves to adapt to the data-centric paradigm. New systems must use data locality to succeed that adaptation. Current systems ignore the incurred cost of communication and rely on the hardware cache coherency to virtualize data movement. Increasing amount of data reveals more communication between the processing elements and that situation requires supporting data locality and affinity. With the upcoming new model, data locality should be succeeded with recent technologies which contain tiling, data layout, array views, task and thread affinity, and topology-aware communication libraries. Combination of the best of these technologies can help us develop a comprehensive model for managing data locality on a high-performance computing system [49].

4.1. Data partition strategy

Domain decomposition is an important subject in high-performance modeling and numerical simulations in the area such as intelligence and military, meteorology, agriculture, urbanism, and search and rescue [50]. It makes possible parallel computing by dividing a large computational task into smaller parts and distributing them to different computing resources. To increase the performance, high-performance computing is extensively used by dividing the entire problem into multiple subdomains and distributing each one to different computing nodes in a parallel fashion. Inconvenient allocation of resources induces imbalanced task loads and redundant communications among computing nodes. Hereby, resource allocation is a vital part which has a deep impact on the efficiency of the parallel process. Resource allocation algorithm should minimize total execution time by taking into consideration the communication and computing cost for each computing node and reduce total communication cost for the entire processing. In this chapter, a new data partitioning strategy is proposed to benefit from the current situation of resources on the cloud. At this new strategy, RS data should be partitioned based on performance metrics which is formulated by a combination of available resources of network, memory, CPU, and storage. After appropriate cloud site and resource nodes are found by stage 1 and 2 as described in Sections 6.2 and 6.3, the receiving data is divided into the selected resource nodes according to the number of available resources on that cloud. For this dividing operation, the system should determine which portion of data will be allocated to found nodes based on some performance metrics such as the below heuristic function:

$$p_i = \frac{t^i_{throughput} + t^i_{processing}}{\sum_{j=1}^{n} t^j_{throughput} + t^j_{processing}} \tag{1}$$

p_i is the portion of node i and $t^i_{throughput}$ are the transfer time needed for data with network throughput of i and processing time needed for data with CPU frequency of i. $\sum_{j=1}^{n} t^j_{throughput} + t^j_{processing}$ is the total sum of transfer and processing time for selected nodes. In the formula, transfer time can be computed with $t^i_{throughput} = data\_size/bandwidth_i$ and processing time can be computed with $t^i_{processing} = data\_size/CPU\_frequency_i$.

4.2. Geo-distributed cloud

Geo-distributed cloud is the application of cloud computing technologies which consist of multiple cloud sites distributed in different geographic locations to interconnect data and applications [51]. Cloud providers prefer the distributed cloud systems to enable lower latency and provide better performance for cloud consumers. Recently, most of the large online services have been geo-distributed toward the exponential increase in data [52]. The most important reason for realizing the services as geo-distributed is latency. Geo-distributed clouds provide a point of presence nearby clients with reference to reduce latency. In this chapter, we introduce a novel resource allocation technique for managing RS big data in the geo-distributed private cloud. This new approach will select the most appropriate cloud site which has minimum latency. It also finds efficient data layout for data which gives a higher performance for selected data nodes in the related cloud site. Within this context, resource management should match instantaneous application requirements with optimal CPU, storage, memory, and network resources [53]. Putting the data into a more appropriate node with enough resources and providing an efficient layout of the dependent data partitions on the nodes with minimum latency on the network should decrease processing time of algorithms and also minimize transferring time of dependent data which is needed by algorithm processing on different nodes.

In the proposed approach, acquired RS data should be stored on geo-distributed cloud within two stages (**Figure 4**). In the first stage, each cloud site determines a score based on latency, bandwidth capacity, CPU, memory, and storage workloads. At that point, each cloud site has a multi-criteria decision-making (MCDM) process. MCDM is a subdiscipline of operation research that evaluates multiple criteria in decision-making problems [54]. As a criteria value for the resource, workloads of resources could be computed by dividing the used amount of the resource to the total capacity of it. For CPU workload, the equation is given as follows:

$$W^i_{CPU} = \frac{\left(\sum_{j=1}^{n} c_{ij}\right)}{t^i_{CPU}} \tag{2}$$

Let t^i_{CPU} be the total number of physical CPU threads in the cloud site i and c_{ij} be the number of virtual CPUs that are allocated for virtual machines (VMs) in the cloud site. For storage workload, the equation is given as follows:

$$W^i_{STR} = \frac{\left(\sum_{j=1}^{n} s_{ij}\right)}{t^i_{STR}} \tag{3}$$

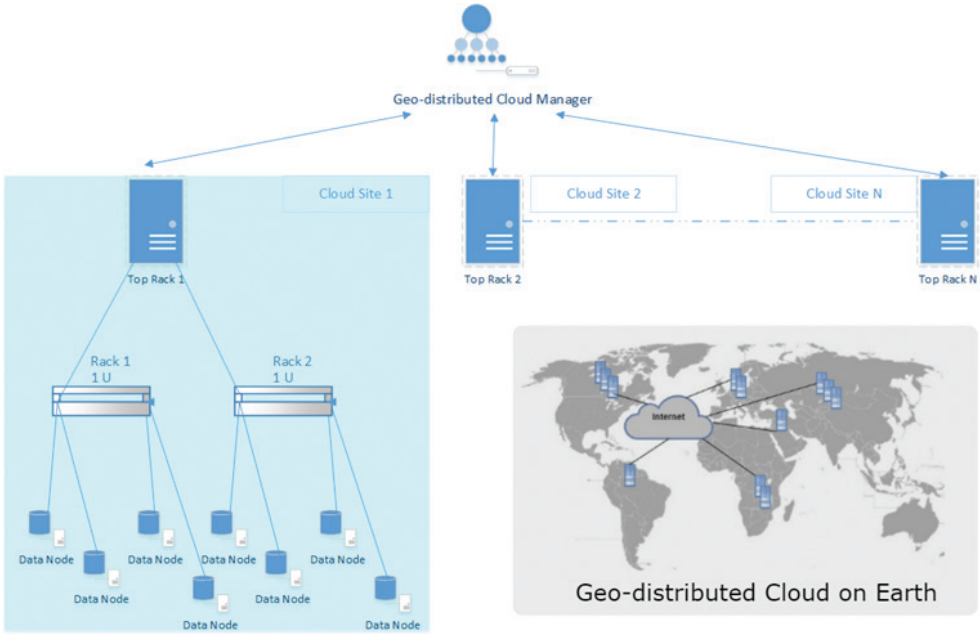

Figure 4. Geo-distributed cloud hierarchy and scene on earth.

Let t^i_{STR} be the total size of storage in the cloud site i and s_{ij} be the size of storage that is used for coming data in the cloud site. For memory workload, the equation is given as follows:

$$W^i_{MEM} = \frac{\left(\sum_{j=1}^{n} m_{ij}\right)}{t^i_{MEM}} \tag{4}$$

Let t^i_{MEM} be the total size of memory in the cloud site i and m_{ij} be the size of storage that is used for coming data in the cloud site.

Time of transferring data between consumer and cloud site is described as latencyL_i, bandwidth between them is B_i.

After determining criteria for decision-making, a processing method is needed to compute numerical values using the relative importance of the criteria to determine a ranking of each alternative [54]. Some of the well-known MCDM processing models are weighted sum model (WSP), weighted product model (WPM), and analytic hierarchy process (AHP). If we define the solution with AHP which is based on decomposing a complex problem into a system of hierarchies, the best alternative could be defined as the below relationship:

$$A_{AHP} = \min_i \sum_{j=1}^{N} w_{ij} C_j, \quad \text{for } i = 1, 2, 3, \ldots, M. \tag{5}$$

where $\sum_{j=1}^{N} w_{ij} = 1$, N is criteria amount, and *i* cloud site. If we write the model for earlier five criteria:

$$A_{AHP} = \min_i \left(w_1 L_i + w_2 \frac{1}{B_i} + w_3 W_{CPU}^i + w_4 W_{STR}^i + w_5 W_{MEM}^i \right) \text{ for } i = 1, 2, 3, \ldots, M \quad (6)$$

Although AHP is similar to WSM, it uses related values instead of actual values. This makes it possible to use the AHP in multidimensional decision-making problems by removing the problem of combining different dimensions in various units (similar to adding oranges and apples).

After the minimum valued alternative cloud site is found, the second stage takes place for evaluating which resources should be used optimally in the related cloud site and finding an optimal layout in the network for RS big data.

4.3. Resource optimization in cloud network with performance metrics

Large-scale networks and its applications lack centralized access to information [55]. When we interpret RS big data on the cloud as a large-scale network, optimization of resource allocation depends on local observation and information of each node. At this point, control and optimization algorithms should be deployed in a distributed manner for finding optimum resource allocation in such a network. Optimization algorithm should be robust against link or node failures and scalable horizontally.

To succeed distributed optimization of the system, each node should run the optimization algorithm locally which can be called as an agent. At this system which consists of multi-agents connected over a network, each agent has a local objective function and local constraint set which are known by just this agent. The agents try to decide on a global decision vector cooperatively based on their objective function and constraints (**Figure 5**).

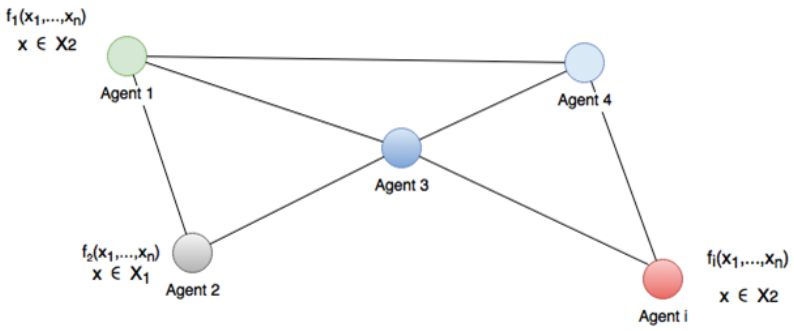

Figure 5. Multi-agent optimization problem for resource allocation on a network. $f_i(x)$: Local objective function where f: $\mathbb{R}^n \to \mathbb{R}$. X_i: Local constraint set where $X_i \subset \mathbb{R}^n$. x: Global decision vector which agents collectively try to decide on, where $x \in \mathbb{R}^n$.

http://dx.doi.org/10.5772/intechopen.75934

The agents cooperatively optimize a global objective function denoted by f(x), which is a combination of the local objective functions, that is:

$$f(x) = T(\, f_1(x), \ldots, f_i(x)\,) \tag{7}$$

where $T : \mathbb{R}^n \to \mathbb{R}$ and decision vector x is bounded by a constraint set, where $x \in C$, which consists of local constraints and global constraints that may be imposed by the network structure, that is:

$$C = \left(\bigcap_{i=1}^{n} X_i \right) \cap C_g \tag{8}$$

where C_g represent global constraints. This model leads to following optimization problem:

$$minimize\, f(x)\ subject\ to\ x \in C \tag{9}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and the set C is constraint set. The decision vector in Eq. (9) can be considered as resource vector whose component corresponds to resources allocated to each node or global decision vector which is estimated by the nodes on the network using local information.

After defining some basic notation and terminology for optimization of a function, continue with where we left off. The cloud site should decide which resources will be used for coming RS data when it is determined to receive it. Each node in the cloud site evaluates network latency and bandwidth between other nodes for optimum network communications. In addition to that, current amounts of CPU, memory, and storage are also taken into account together for finding the best-fitted solution to store and process RS data. Hence each node solves the defined formula:

$$\min f_i(x) = \sum_{j=1}^{n} x_j \left[\left(w_1 L_{ij} + w_2 \frac{1}{B_{ij}} + w_3 H_{ij} + w_4 W^j_{CPU} + w_5 W^j_{STR} + w_6 W^j_{MEM} \right) \right]$$

$$\text{for } i = 1, 2, 3, \ldots, N$$

$$\text{subject to} \begin{cases} \sum_{j=1}^{n} x_j A^j_{MEM} \geq I_{size} \\ \sum_{j=1}^{n} x_j A^j_{STR} \geq I_{size} \\ H_{ij} \leq Hop_{max} \\ B_{ij} \leq C_{max} \\ x_{i,j} \in \{0, 1\} \forall i, j \end{cases} \tag{10}$$

where x_j indicates that the jth node would be in the data-receiving group together with node i or not, L_{ij} is latency between i and j, B_{ij} is bandwidth between i and j, H_{ij} is hop count between i and j, and W^j_{CPU}, W^j_{STR}, and W^j_{MEM} are CPU, storage, and memory resources in node j. Each node should solve $f_i(x)$ to minimize decision vector with AHP model in a decentralized manner. The

individual optimization problem is a mix integer program for each node. Lagrangian relaxation is a heuristic method that stands for solving mix integer problems with decomposing constraints. The idea of the method is to decompose constraints which complicated the problem by adding them to the objective function with the associated vector μ called the Lagrange multiplier. After applying it to our problem, we derive the dual problem of Eq. (9) for an efficient solution by adding complicated constraints 1 and 2.

$$\min L(x,\mu,\lambda) = \sum_{j=1}^{n} x_j U_{ij} + \mu_j \left(I_{size} - \sum_{j=1}^{n} x_j A_{MEM}^{j} \right) + \lambda_j \left(I_{size} - \sum_{j=1}^{n} x_j A_{STR}^{j} \right)$$
$$\text{subject to} \begin{cases} H_{ij} \le Hop_{max} \\ B_{ij} \le C_{max} \\ x_{i,j} \in \{0,1\} \forall i,j \end{cases} \tag{11}$$

where utilization function for node:

$$U_{ij} = \left[\left(w_1 L_{ij} + w_2 \frac{1}{B_{ij}} + w_3 H_{ij} + w_4 W_{CPU}^{j} + w_5 W_{STR}^{j} + w_6 W_{MEM}^{j} \right) \right]$$

and $\mu_j \ge 0 \forall j$ and $\lambda_j \ge 0 \forall j$ are the dual variables.

After obtaining above optimization problem, it could be separable in the variables x_i and it also decomposes into sub-problems for each node i. Thus each node needs to solve the one-dimensional optimization problem Eq. (11). This optimization problem consists of its own utility function and Lagrangian multipliers which are available for node i. Generally, the subgradient method is used to solve the obtained dual problem because of simplicity of computations per iteration. First-order methods such as subgradient have slower convergence rate for high accuracy but they are very effective in large-scale multi-agent optimization problems where the aim is to find near-optimal approximate solutions [55].

When one optimal approximate solution is found, RS data should be divided into partitions and proportionally distributed to found nodes in the solution according to performance heuristic (Eq. (1)) which is given in Section 6.1.

5. Conclusion

As a result of technological developments, the amount of data produced by many organizations on a daily basis has increased to terabyte levels. Remotely sensed data, which is spatially and spectrally amplified and heterogeneous by means of different sensing techniques, causes great difficulties in storing, transferring, and analyzing with conventional methods. It has become a necessity to implement distributed approaches instead of conventional methods that are inadequate in critical applications when real/near-real-time analysis of relevant big data is needed. Existing distributed file systems, databases, and high-performance computing systems are experiencing difficulties in

optimizing workflows for analysis, in the storage, and retrieval of spatial big data of remote sensing and data streams.

According to investigated researches in this chapter, it is observed that existing techniques and systems cannot find a solution that covers the existing problems when analyzing the real and near-real-time big data analysis in remote sensing. Hadoop and similar technologies have attracted increasingly in geosciences communities for handling big geospatial data. Many investigations were carried out for adopting those technologies to processing big geospatial data, but there are very few studies for optimizing the computing resources to handle the dynamic geo-processing workload efficiently.

In this chapter, a two-stage innovative approach has been proposed to store RS big data on a suitable cloud site and to process them with optimizing resource allocation on a geo-distributed cloud. In the first stage, each cloud site determines a score based on latency, bandwidth capacity, CPU, memory, and storage workloads with an MCDM process. After minimum valued alternative cloud site is found, the second stage takes place for evaluating which resources should be used optimally in related cloud site and finding an optimal layout in the network for RS big data with respect to latency, bandwidth capacity, CPU, memory, and storage amount. Lastly, data should be divided into partitions based on a performance metric which could be computed with available network and processing resources of selected nodes in the cloud site.

As future work, optimal replication methods will be searched for preventing failure situations when transferring and processing RS data in a distributed manner. For succeeding that, a performance-based approach is considered to maintain high-performance computing.

Author details

Mustafa Kemal Pektürk[1]* and Muhammet Ünal[2]

*Address all correspondence to: mkpekturk@havelsan.com.tr

1 HAVELSAN A.Ş., Ankara, Turkey

2 Gazi University, Ankara, Turkey

References

[1] Fadiya SO, Saydam S, Zira VV. Advancing big data for humanitarian needs. Procedia Engineering. 2014;**78**:88-95

[2] Chen CLP, Zhang C-Y. Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences. 2014;**275**:314-347

[3] Özköse H, Arı ES, Gencer C. Yesterday, today and tomorrow of big data. Procedia-Social and Behavioral Sciences. 2015;**195**:1042-1050

[4] Rathore MMU et al. Real-Time Big Data Analytical Architecture for Remote Sensing Application. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2015;**8**(10):4610-4621

[5] Yue P et al. Sensor Web event detection and geoprocessing over big data. In: 2014 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE; 2014

[6] Lee CA et al. Recent developments in high performance computing for remote sensing: A review. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2011;**4**(3):508-527

[7] Toma AC et al. Computational challenges in processing large hyperspectral images. In: Tier 2 Federation Grid, Cloud & High Performance Computing Science (RO-LCG), 2012 5th Romania. IEEE. 2012. pp. 111-114

[8] Ma Y et al. Remote sensing big data computing: Challenges and opportunities. Future Generation Computer Systems. 2014;**51**:47-50

[9] Ma Y et al. Towards building a data-intensive index for big data computing–a case study of remote sensing data processing. Information Sciences. 2014;**319**:171-188

[10] Aji A et al. Hadoop GIS: A high performance spatial data warehousing system over mapreduce. Proceedings of the VLDB Endowment. 2013;**6**(11):1009-1020

[11] Zhong Y, Fang J, Zhao X. VegaIndexer: A distributed composite index scheme for big spatio-temporal sensor data on cloud. In: 2013 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE; 2013

[12] Wickramaarachchi C et al. Real-time Analytics for fast evolving social graphs. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). IEEE; 2015

[13] Oliveira SF, Fürlinger K, Kranzlmüller D. Trends in computation, communication and storage and the consequences for data-intensive science. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS). IEEE; 2012

[14] Fernández A et al. Pattern recognition in Latin America in the "big data" era. Pattern Recognition. 2015;**48**(4):1185-1196

[15] Lin F-C et al. The framework of cloud computing platform for massive remote sensing images. In: 2013 IEEE 27th International Conference onAdvanced Information Networking and Applications (AINA). IEEE; 2013. pp. 621-628

[16] Krämer M, Senner I. A modular software architecture for processing of big geospatial data in the cloud. Computers & Graphics. 2015;**49**:69-81

[17] Bernabe S et al. Hyperspectral unmixing on GPUs and multi-core processors: A comparison. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2013;**6**(3):1386-1398

[18] Chen J, Zheng G, Chen H. ELM-MapReduce: MapReduce accelerated extreme learning machine for big spatial data analysis. In: 2013 10th IEEE International Conference on Control and Automation (ICCA). IEEE; 2013

[19] Kambatla K et al. Trends in big data analytics. Journal of Parallel and Distributed Computing. 2014;**74**(7):2561-2573

[20] Hashem IAT et al. The rise of "big data" on cloud computing: Review and open research issues. Information Systems. 2015;**47**:98-115

[21] Song G et al. Constructing gazetteers from volunteered big geo-data based on Hadoop. Computers, Environment and Urban Systems; 2017;**61**:172-186

[22] Yang C et al. A spatiotemporal compression based approach for efficient big data processing on cloud. Journal of Computer and System Sciences. 2014;**80**(8):1563-1583

[23] Douglas CC. An open framework for dynamic big-data-driven application systems (DBDDAS) development. Procedia Computer Science. 2014;**29**:1246-1255

[24] Konstantinos K, Bliziotis D, Karmas A. A scalable geospatial web service for near real-time, high-resolution land cover mapping. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2015;**8**(10):4665-4674

[25] Wang L et al. A parallel file system with application-aware data layout policies for massive remote sensing image processing in digital earth. IEEE Transactions on Parallel and Distributed Systems. 2015;**26**(6):1497-1508

[26] Tehranian S et al. A robust framework for real-time distributed processing of satellite data. Journal of Parallel and Distributed Computing. 2006;**66**(3):403-418

[27] Ma Y et al. Generic parallel programming for massive remote sensing data processing. In: 2012 IEEE International Conference on Cluster Computing (CLUSTER). IEEE; 2012

[28] Angleraud C. magHD: a new approach to multi-dimensional data storage, analysis, display and exploitation. In: IOP Conference Series: Earth and Environmental Science. Vol. 20. No. 1. IOP Publishing; 2014

[29] Rathore MMU et al. Real-time continuous feature extraction in large size satellite images. Journal of Systems Architecture. 2016;**64**:122-132

[30] SUPPLY, POWER. Zhou et al. (45) Date of Patent: Aug 26, 2014

[31] Schmuck FB, Haskin RL. GPFS: A shared-disk file system for large computing clusters. FAST. 2002;**2**

[32] Tantisiriroj W et al. On the duality of data-intensive file system design: reconciling HDFS and PVFS. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM; 2011

[33] Assunção MD et al. Big data computing and clouds: Trends and future directions. Journal of Parallel and Distributed Computing. 2015;**79**:3-15

[34] Wang L et al. Massive remote sensing image data management based on HBase and GeoSOT. In: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE; 2015

[35] Vora MN. Hadoop-HBase for large-scale data. In: 2011 International Conference on Computer Science and Network Technology (ICCSNT). Vol. 1. IEEE; 2011

[36] Tumwizere RP, Karpjoo J. A survey on computing technology applications in remote sensing. In: 2012 8th International Conference on Computing and Networking Technology (ICCNT). IEEE; 2012

[37] Setoain J et al. GPU for parallel on-board hyperspectral image processing. The International Journal of High Performance Computing Applications. 2008;**22**(4):424-437

[38] Qu H et al. Parallel acceleration of SAM algorithm and performance analysis. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2013;**6**(3):1172-1178

[39] Qu H et al. Parallel implementation for SAM algorithm based on GPU and distributed computing. In: 2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE; 2012

[40] Wu Z et al. Parallel implementation of sparse representation classifiers for hyperspectral imagery on GPUs. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2015;**8**(6):2912-2925

[41] Shan N, Wang X-S, Wang Z-S. Efficient FPGA implementation of cloud detection for real-time remote sensing image processing. In: 2010 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia). IEEE; 2010

[42] Plaza A et al. Parallel implementation of hyperspectral image processing algorithms. In: IEEE International Conference on Geoscience and Remote Sensing Symposium. IGARSS 2006. IEEE; 2006

[43] Plaza A et al. High performance computing for hyperspectral remote sensing. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2011;**4**(3):528-544

[44] Cavallaro G et al. Scalable developments for big data analytics in remote sensing. In: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE; 2015

[45] O'Driscoll A, Daugelaite J, Sleator RD. 'Big data', Hadoop and cloud computing in genomics. Journal of Biomedical Informatics. 2013;**46**(5):774-781

[46] Vinay A et al. Cloud based big data analytics framework for face recognition in social networks using machine learning. Procedia Computer Science. 2015;**50**:623-630

[47] Li Z et al. Automatic scaling hadoop in the cloud for efficient process of big geospatial data. ISPRS International Journal of Geo-Information. 2016;**5**(10):173

[48] Tate A et al. Programming abstractions for data locality. In: PADAL Workshop 2014, April 28–29. Lugano: Swiss National Supercomputing Center (CSCS); 2014

[49] Pektürk MK, Ünal M. A review on real-time big data analysis in remote sensing applications. In: 2017 25th Signal Processing and Communications Applications Conference (SIU). IEEE; 2017

[50] Gui Z et al. Developing subdomain allocation algorithms based on spatial and communicational constraints to accelerate dust storm simulation. PLoS One. 2016;**11**(4):e0152250

[51] Wu Y et al. Scaling social media applications into geo-distributed clouds. IEEE/ACM Transactions on Networking (TON). 2015;**23**(3):689-702

[52] Narayanan I et al. Towards a leaner geo-distributed cloud infrastructure. In: HotCloud; 2014

[53] Tso FP, Jouet S, Pezaros DP. Network and server resource management strategies for data Centre infrastructures: A survey. Computer Networks. 2016;**106**:209-225

[54] Triantaphyllou E et al. Multi-criteria decision making: An operations research approach. Encyclopedia of Electrical and Electronics Engineering. 1998;**15**(1998):175-186

[55] Nedic A, Ozdaglar A. 10 cooperative distributed multi-agent. Convex Optimization in Signal Processing and Communications. 2010;**340**