Contents lists available at ScienceDirect
Future Generation Computer Systems
journal homepage: www.elsevier.com/locate/fgcs

Remote sensing big data computing: Challenges and opportunities

Yan Ma[a], Haiping Wu[b], Lizhe Wang[a,*], Bormin Huang[c], Rajiv Ranjan[d], Albert Zomaya[e], Wei Jie[f]

[a] *Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, PR China*
[b] *Ministry of Education Key Laboratory for Earth System Modeling and Center for Earth System Science, Tsinghua University, PR China*
[c] *Space Science and Engineering Center, University of Wisconsin-Madison, USA*
[d] *Computational Informatics, CSIRO, Australia*
[e] *School of Information Technologies, University of Sydney, Australia*
[f] *School of Computing and Technology, West London University, UK*

HIGHLIGHTS

- This paper identifies the properties and features of remote sensing big data.
- This paper reviews the stat-of-the-arts of remote sensing big data computing.
- This paper discusses the “data-intensive computing” issues in remote sensing big data processing.

ARTICLE INFO

Article history:
Received 30 July 2014
Received in revised form
19 September 2014
Accepted 25 October 2014
Available online xxxx

Keywords:
Remote sensing data processing
Big data
Data-intensive computing

ABSTRACT

As we have entered an era of high resolution earth observation, the RS data are undergoing an explosive growth. The proliferation of data also give rise to the increasing complexity of RS data, like the diversity and higher dimensionality characteristic of the data. RS data are regarded as RS “Big Data”. Fortunately, we are witness the coming technological leapfrogging. In this paper, we give a brief overview on the Big Data and data-intensive problems, including the analysis of RS Big Data, Big Data challenges, current techniques and works for processing RS Big Data.

1. Introduction

The recent advances in remote sensing (RS) and computer techniques give birth to the explosive growth of remote sensing (RS) data. Momentary, the observation data streaming from the spacecrafts in current active missions of NASA would approximately be 1.73 GB gigabytes [1]. The RS data gathered by a single satellite data center are dramatically increasing by several terabytes per day [2]. According to the statistics of OGC [3], the global archived observation data would probably exceed one Exabyte. Especially, the advent of the high-resolution earth observation era (EOS-4) has also led to the high dimensionality of the RS image data. Remote sensing data are recognized as “Big Data” in some certain sense [4]. Meanwhile, the accurate and up-to-date information provided by the continual global earth observation are revolutionizing the way that earth system is interpreted. The large-scale environmental monitoring and researches [5–7] are exploiting regional to global covered multi-temporal and multi-sensor RS data for processing. Obviously, large remote sensing applications overwhelmed with massive remote sensing data are regarded as typical data-intensive issues [8].

The unprecedented proliferation of data has posed significant challenges in managing, processing and interpreting these RS “Big Data”. Great efforts have been towards the incorporation of the high-performance computing (HPC) paradigm in RS applications [9–11]. These HPC-based approaches have became the most dominant yet efficient way for addressing the enormous computational requirements introduced by massive RS data. However, the computation capability available is no longer the rate-limiting factor when in contrast to the traditional compute-intensive issues. Despite of the huge processing power, the cluster-based HPC systems still remain considerably challenging with the data-intensive issues [12] emerged in RS applications. There involves a number of

* Corresponding author.
E-mail address: Lizhe.Wang@gmail.com (L. Wang).

http://dx.doi.org/10.1016/j.future.2014.10.029
0167-739X/© 2014 Elsevier B.V. All rights reserved.

data availability [1,12] related challenging issues. Firstly, the enormous geographically distributed RS datasets with even higher dimensionality and significant metadata, on a scale that goes beyond the traditional data management practice both in memory and disk. Moreover, the intolerable I/O burden introduced by the intensive irregular RS data access patterns has made the common parallel file systems with stereotypical physical data layout no longer inapplicable. Secondly, the current task scheduling strategies [13,14] seeking load balancing among computational resources seldom take the data availability into account. To complicate the situation that some large-scale RS applications [10] could be structured as a large collection of data dependent small tasks with ordering constraint. The optimized scheduling of these bunch of tasks is a critical issue for achieving higher performance. In addition, the parallel programming for data-intensive RS applications on MPI-enabled (Message Passing Interface) cluster systems with multi-level hierarchy and increasing scale turn out to be rather trivial, difficult and error-prone.

In the near future, some new requirements together with problems will emerge with the further increasing amount and widespread application of RS data. Obviously, the increasing demand for real-time or near real-time processing capability by many time-critical RS applications [15,16] have definitely made the data-intensive issue even worse. The other thing is that it is anything but easy to offer an ease of use way for on-demand processing of massive RS data from almost every where. Especially, instead of the traditional order-request producing mode, the standard and serialized processing of RS data products are also arisen for an on-line data-triggered producing. The variation of data producing mode also means timely processing of even more amount of RS data. For the requirements list above both in current and near future, we go deep into the potential challenges introduced by the requirements in processing the RS "Big Data" in this paper. Meanwhile, through the discussion of limitation or problems with the state-of-the-art work of massive RS data processing, we demonstrate some possible solution and enabling techniques for these issues.

The rest of this paper is organized as follows. The following two sections respectively introduce what RS "Big Data" is and also discuss the challenges in the processing of these big data. In Section 4, we elaborate on the current work for processing RS big data. Finally, the conclusion section summarizes the whole paper.

2. What is RS "Big Data"

Remote sensing is generally defined as the technology of measuring the characteristics of an object or surface form a distance [17,18]. The RS data are the earth observing data continuously obtaining from space-borne and airborne sensors, as well as some other data acquisition measurements. With the exponential growth of data amount and increasing degree of diversity and complexity, the remotely sensed data are regarded as RS "Big Data". However, since the whole idea of big data is still remaining relatively new, most of the start off efforts are focusing on the definition and discussion of the realm of the big data. Big data [19,12,20] occurs when a large collection of data sets whose volume and rate of data is at a scale that is far beyond the state-of-the-art systems and revolutionize the way of seeking solutions. This is also the case for the remote sensing and earth sciences domain to offer the definition of what RS "Big Data" really is. The RS "Big Data" not merely refers to the volume and velocity of data that outstrip the storage and computing capacity, but also the variety and complexity of the RS data. There are several aspects and features of the RS "Big Data" that need to be discussed: the huge volume and rate of RS data, the diversity of RS data and also the complexity of RS data especially the higher dimensionality.

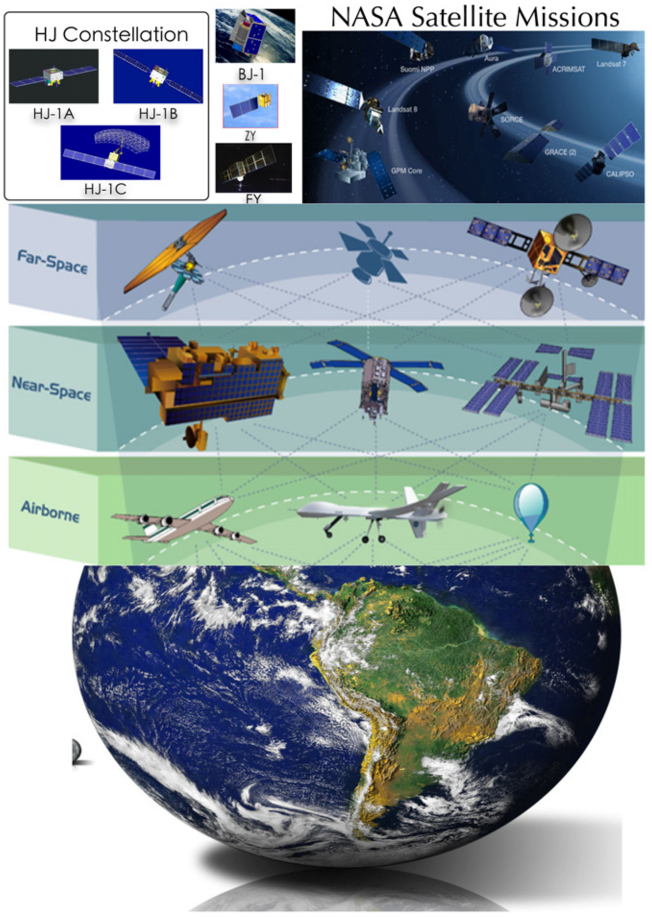


Fig. 1. The satellite network for earth observation.

2.1. The huge volume and velocity of RS data

With the recent advances in sensors and earth observation techniques, we are entering the high-resolution observation era (EOS-4) shown in Fig. 1. The satellite observation network technique is also employed to seek shorter re-visit cycle and larger coverage in the compensation for the limitations of a single sensor. Currently, more than 200 on-orbit satellite sensors are capturing multi-spatial, multi-temporal RS data from multi-sensors. These continual global observing data are capable of covering the global atmosphere, land surface as well as oceans. Wherein, the NASA's Earth Observing System Data and Information System (EOSDIS) [21,22] have successfully managed a growing archived RS data which would currently exceed 7.5 petabytes. During the year 2012, EOSDIS have already distributed more than 4.5 million gigabytes of data [23].

Considering the velocity of data, the RS data archives of EOSDIS are undergoing a growth of 4 TB daily. The data flow offers to the users around the world every day would be about 20 TB, which also means more than 630 million data files. Every unit time, the observing data gathered from nearly 100 active missions of NASA would be about 1.73 GB [1]. The increasing of remote sensing data also brings in the rapid growth of the metadata. For EOSDIS, the amount of data records in the metadata database would probably outstrip 129 millions and also dramatically increasing at a rate of more than 60 thousand every single day [23].

For a data center point of view, we take the statistics of satellite data center of CAS (Chinese Academy of Sciences) for case study. As is demonstrated in Table 1, the data streaming through downlink is around 0.9 Gbps would probably exceed 1.6 Gbps if high-resolution

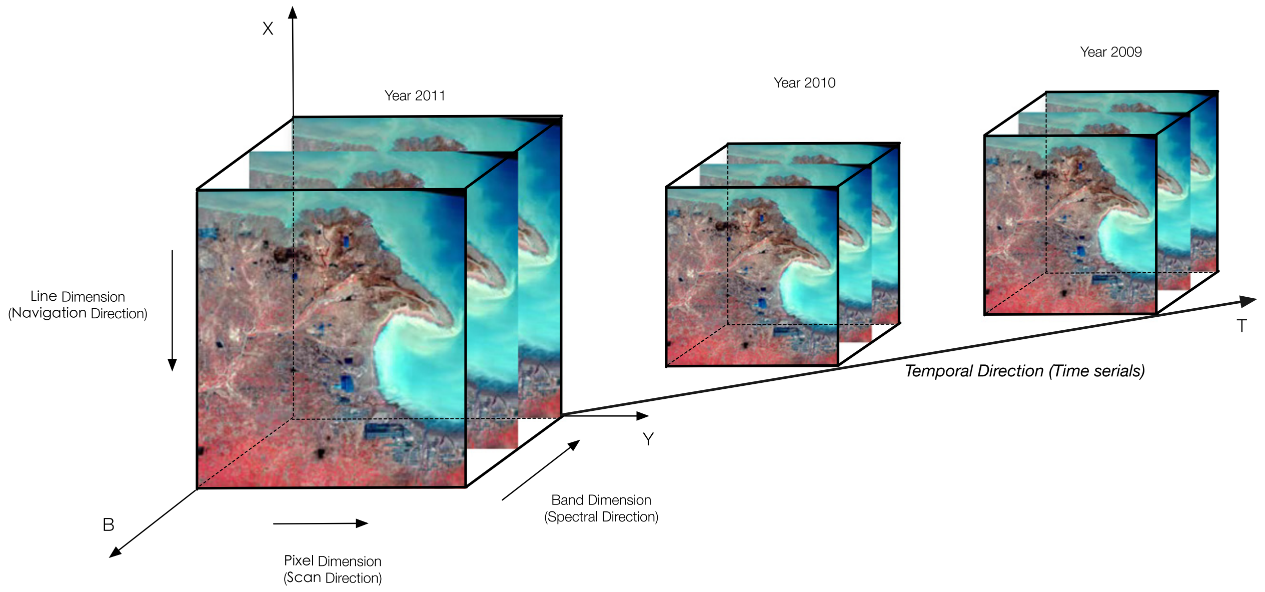

X
Year 2011
Year 2010
Year 2009
Line Dimension (Navigation Direction)
T
Temporal Direction (Time serials)
Y
Band Dimension (Spectral Direction)
B
Pixel Dimension (Scan Direction)

Fig. 2. The dimensionality of the multi-temporal RS data.

Table 1
Satellite data center: the volume and velocity of RS data.

Satellites	Velocity (Mbps)	Volumes (GB/Day)	Volumes (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	10.99	3.92
LANDSAT5	85.00	28.02	9.99
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	11.75
IRS-P6	210.00	46.14	16.45
LANDSAT8	440.00	241.70	86.15
Total	3712.98	2089.06	574.6

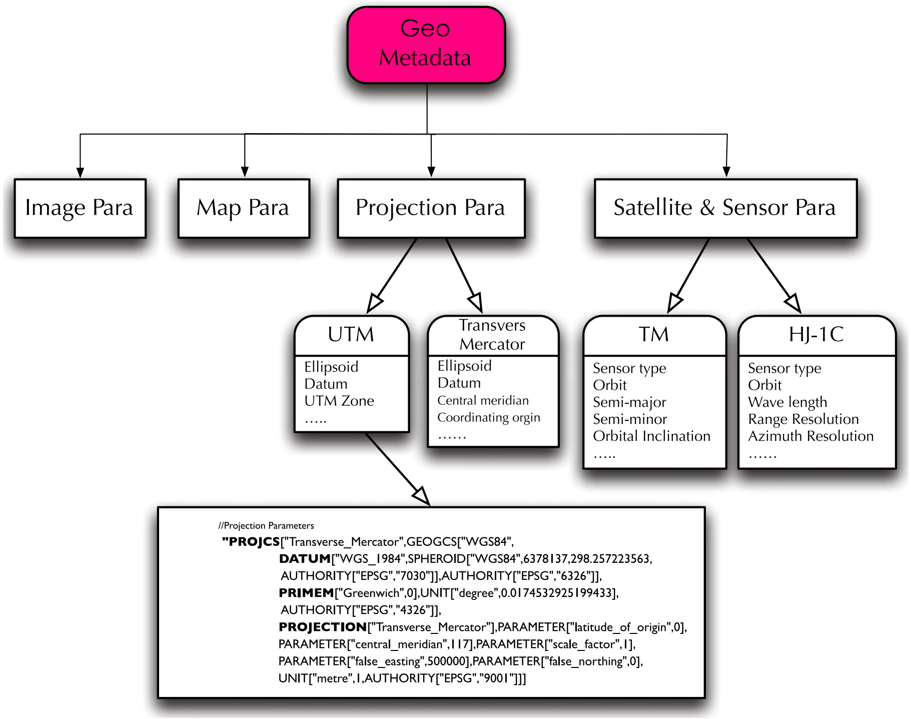
satellites are counted. The aggregate downlinking bandwidth of the multiple satellites is around 3.7 Gbps and would add up to be more than 10 Gbps. The volume of data acquired from a single satellite is about more than 500 GB every day. The total volume of data acquired by three ground stations connecting the data center sums up to about 2 TB per day, and exceeds 0.5 petabytes per year. For a single data center, the volume and data generating rate is considered as moderate "Big" and also manageable [23] from a physical infrastructure point of view. While, according to the statics of OGC [3], data archives of a nation-wide satellite data center would be several petabytes, and the global one would probably exceed one Exabyte. It is extremely massive data waiting for processing.

2.2. The complexity of RS data

The diversity and high dimensionality of data commonly lead to the complexity of the RS "Big Data". The remote sensing data normally serves many earth science disciplines, such as environmental monitoring, land processes, atmospheric, hydrology and oceanography. The wide range of applied disciplines give rise to the diversity of the RS data. As we know that the archives of NASA nearly includes 7000 types of data sets [24]. In most cases, the RS data sets are stored in structured files using various standard formats, including HDF, netCDF, GeoTIFF, FAST, ASCII and so on. Normally, these standard data formats have different data organizations, like different standards presenting metadata tags, different physical structures for organizing metadata and satellite image data. Accordingly, different data formats have its own format specific operation interfaces and libraries. The diversity of data makes the accessing and using of RS data significantly difficult especially for the non-experts.

With the advent of high resolution earth observation, we have sensors with even higher spatial resolution, temporal resolution and also spectral resolution. As a result, we have large numbers satellites and sensors with different resolutions. For a spatial resolution perspective, the high resolution satellites include HJ-1C (5 m), SPOT-5 (2.5 m), IKONOS (1 m), Quickbird (0.61) and Orbview-5 (0.41 m). The median-resolution satellites are LandSat serial satellites (LandSat-5 TM (30 m), LandSat-7 ETM+(15 m) and LandSat-8 OLI (15 m)). The low-resolution satellites involve NOAA (1 km), MODIS (250 m) and FY-1 to FY-3. Considering the temporal resolution, the re-visit cycle of SPOT would be 1 to 4 days, while NOAA would only take few hours. For the spectral point of view, the Hyperion sensor consists of 220 spectral bands with a resolution of 10 nm, the WIS instrument has 812 bands, also the hyperspectral imager equipped in HJ-1A has 128 bands. Following this way, the RS data with a wide variety of different spatial, spectral and even temporal resolutions would inevitably result in the complexity of the RS data. On the other hand, the regional to global remote sensing applications like global climate change or global estimation of grain yield are exploiting multi-temporal, multi-spatial RS data from different sensors for processing. Obviously, these advances always lead to the pixel's multi-dimensionality as shown in Fig. 2.

In addition, the complexity of RS data also lies in the metadata organized into complex data structures. As shown in Fig. 3, the RS data consist of abundant metadata for self-description, including image para for describing the basic size and type information of image data, map info that indicates the geographical location of the data, like latitude/longitude coordinate of image. The projection para includes some structured geographic projection parameters which vary with different projection methods. The complex data



```
//Projection Parameters
"PROJCS["Transverse_Mercator",GEOGCS["WGS84",
          DATUM["WGS_1984",SPHEROID["WGS84",6378137,298.257223563,
          AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],
          PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433],
          AUTHORITY["EPSG","4326"]],
          PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],
          PARAMETER["central_meridian",117],PARAMETER["scale_factor",1],
          PARAMETER["false_easting",500000],PARAMETER["false_northing",0],
          UNIT["metre",1,AUTHORITY["EPSG","9001"]]]
```

Fig. 3. The metadata organization of RS data.

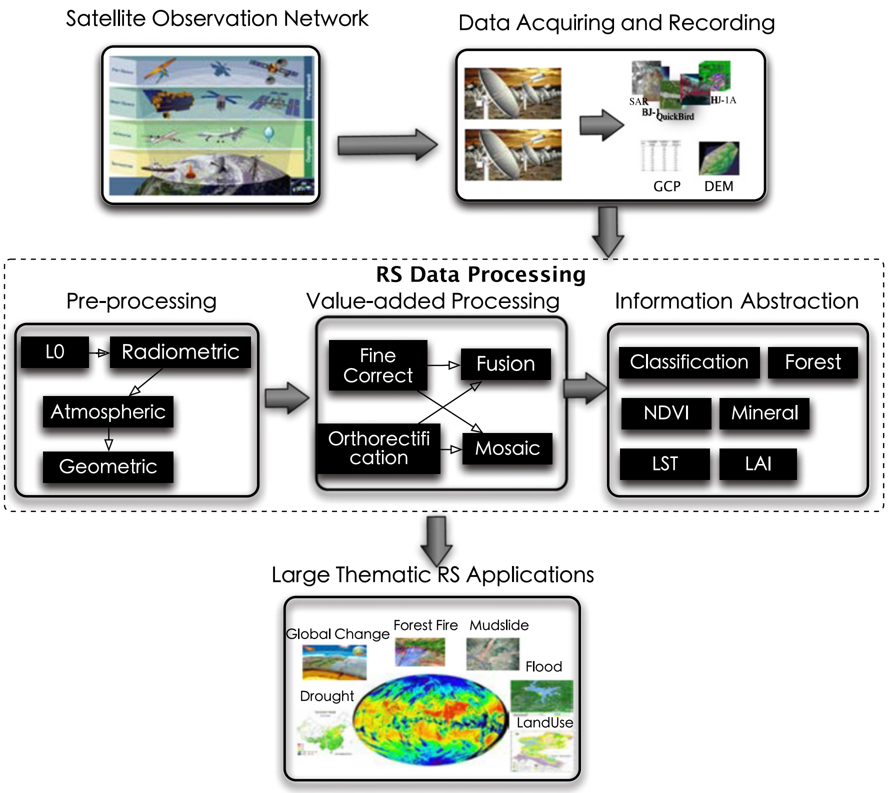


Fig. 4. Analysis of the entire remote sensing data processing flow.

structure for organizing the metadata also makes the easy use of the RS data rather trivial. The other factor to be concerned is that the RS data are naturally geographically distributed among data centers. This also poses complexity of managing and exploiting large region covered RS data which are distributed located for a long time-series analysis.

3. Research issues for processing RS Big Data

The RS "Big Data" is continually acquired for processing and knowledge discovering. Generally, the RS data processing is some what on-the-flow processing. The entire processing flow could be commonly segmented into several stages as depicted in Fig. 4.

Wherein, a number of preprocessing techniques are employed in the preprocessing stage, radiometric correction, geometric correction, image enhancement for removing noise and correcting inconsistencies. The value-add processing is responsible for processing georeferenced data, including the fine correction, orthorectification, fusion, mosaic and etc. The information abstraction stage produces various information products, such as LAI, LST, NDVI and so on. While, the thematic applications normally across various disciplines, examples are disaster monitoring, global change and etc. Take the large-scale spatial and temporal analysis of socio-economic phenomena for example, various types of data describing the same region are synthesized together for processing. These data could involve some large region covered fully structured data including GIS data, thematic maps (phenology and precipitation data) and even social statistical data (population, temperature and GDP). Accordingly, the processing of these massive RS data is quite a challenging issue. The difficulty lies in the data storing, memory loading, processing and also analyzing. If these challenges could not be properly surmount, the RS big data would become a treasure that we are not capable of exploring it. These research issues include:

- The difficulties lie in the efficient managing of the massive RS data;
- The intensive irregular data access patterns charge for the poor parallel I/O performance;
- The loading and transmission of RS "Big Data";
- Tons of data-dependent tasks for optimal scheduling;
- The efficient and productive programming of RS applications on hierarchical cluster-based parallel system.

3.1. *The difficulties of managing RS "Big Data"*

The explosion growth of the RS "Big Data" are changing the way remote sensing data are captured and managed. Naturally, the remote sensing data acquired by different data centers are geometrically distributed. However, these data centers are geometrically far away and normally connected by the Internet. It is critically important for a data managing system to manage these massive distributed RS data for a global data sharing and interoperation. This also means the demand for more storage devices and easy accessibilities of distributed located RS "Big Data". To meet these challenges, the innovations of traditional data storage devices and architectures are no doubt imperative [25]. The other difficulty also lies in the high dimensionality of the RS data, 3-D or even 4-D. The high dimensionality characteristic makes the distributed storing and accessing rather complicated. The main issue is how to organize and map the multi-dimensional remote sensing imageries to 1-D data array. To be specific, the suitable space-filling curves should be chosen for appropriate data partition and data organization with the purpose of better data availability. Besides, the RS data are characteristic with complex structured metadata, especially the geographical metadata. During the RS data processing, the geographical metadata would be also requested for recalculation in case of the RS data accessing. However, it is fairly cumbersome for the data managing system to support the storing and indexing of geographical metadata efficiently.

3.2. *The intensive irregular RS data access pattern*

For big data, the data availability is on top priority of the data processing and knowledge discovering [19]. There exists a restraint in common processing systems: CPU-heavy but I/O-poor. Mostly, the RS applications perform intensive but irregular data access patterns [26]. These irregular I/O patterns illustrated in Fig. 5 are introduced by the different degree of dependency between computation of algorithm and RS data during processing. The computation of each pixel usually depends on its neighborhood or even the data from other spectral bands. Generally, various degree of data dependencies would result in different data access patterns that vary across applications. For example, the regional dependent computation featured applications tend to request lots of small data blocks scattered throughout the file in one logical I/O. While, the band dependent computation normally involve accessing of small data region from multiple band files simultaneously. Unfortunately, these I/O patterns are seldom natively supported by most of the dominant parallel file systems (PFS). Traditionally, the non-contiguous I/O [27] is implemented by repeatedly calling a number of individual data requests, each of which accesses a small piece of consecutive data. While band related I/O is translated into many separate data requests each of which accesses from one band file. Obviously, this kind of implementation is rather time-consuming and tedious.

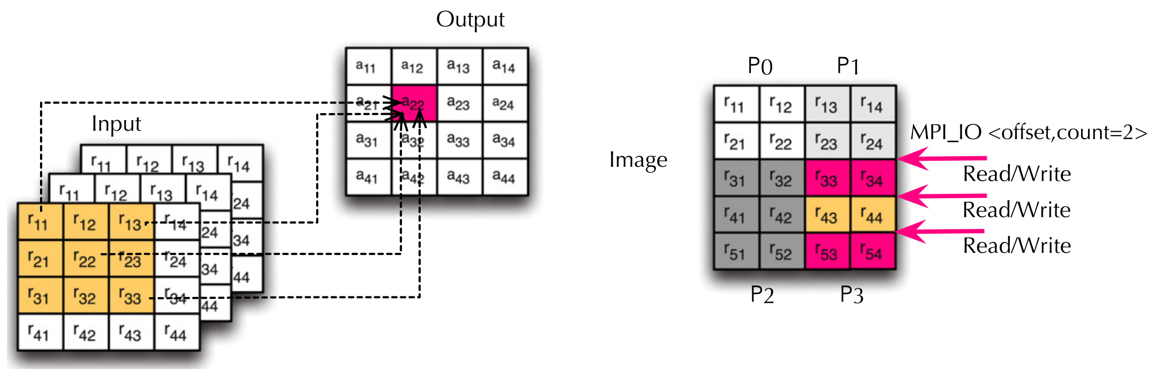
However, a large amount of scientific applications perform small non-contiguous I/O just like RS applications do [28]. But the situation is that, only one tenth of the peak I/O performance could be achieved by these kind of applications [29]. This is because most of the widespread PFSs are optimized for contiguous data accessing. Basically speaking, the parallel I/O interfaces and physical data layout over storages do not match to the expected data access patterns of RS applications [30]. Therefore, the intolerable I/O burden introduced by the intensive irregular RS data access patterns have made the traditional PFSs with stereotypical physical data layout no longer inapplicable.

3.3. *The data loading and transmission of RS "Big Data"*

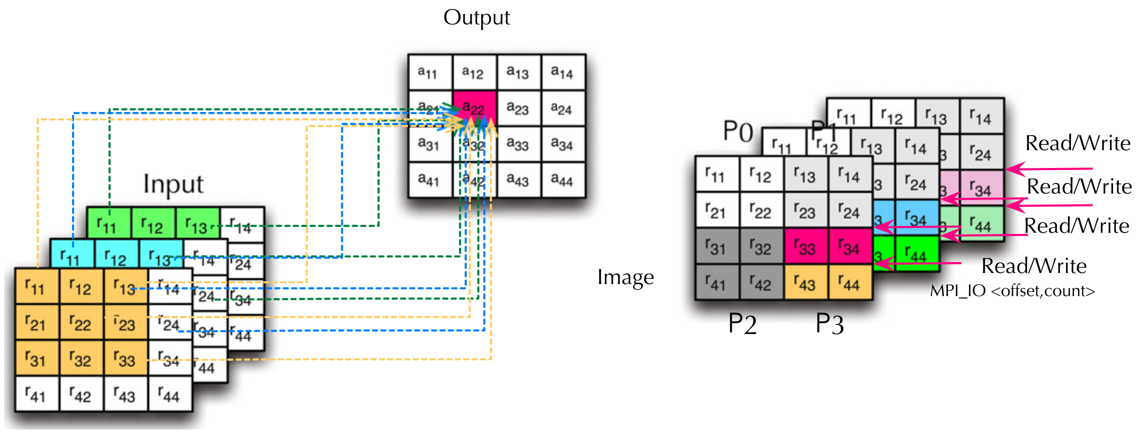
The massive high dimensional RS data make the data loading, memory residing as well as the data transmission among processing nodes during processing rather complicated and inefficient. Firstly, the enormous amount of RS data has far beyond the limited memory capacity of a single computer. The RS datasets generally consist of multi-dimensional images and complex structured metadata. It is rather complicated to offer a proper and large data structure for loading and residing these massive RS data into local memory. The other problems are that the communication of the RS data blocks are common during the parallel implementing of various RS applications. Due to the limitation of the network bandwidth, the communication of RS data would be time-consuming especially when the volume of communication is extremely large. However, the communication of complex RS data structure across processing nodes are not well supported by the current MPI implementations. As a result, repeated calling of low-level MPI send/receive communication APIs is required at the penalty of significant performance decline.

3.4. *Scheduling of large number of data-dependent tasks*

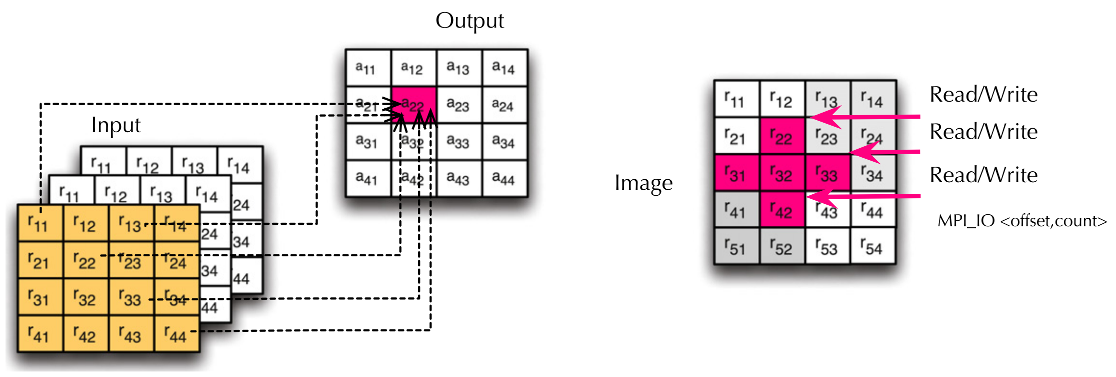
For many large RS applications, like large-scale RS data mosaicking [10] and hydrological modeling of large watershed [31] could be described as a large collection of data dependent small tasks (Fig. 6). The processing of these applications become extremely difficult because of the dependency among a large collection of tasks which give rise to ordering constraint. The succeeding tasks have to wait for the output data of preceding task to be available. The optimized scheduling of these bunch of tasks is critical to achieve higher performance. Therefore the problems lie in decoupling the dependence relationships among tasks so as to achieve a minimal execution length.



(a) Neighbor-based processing and rectangular-block access pattern.



(b) Band-related processing and cross-file access pattern.



(c) Irregular processing and irregular access pattern.

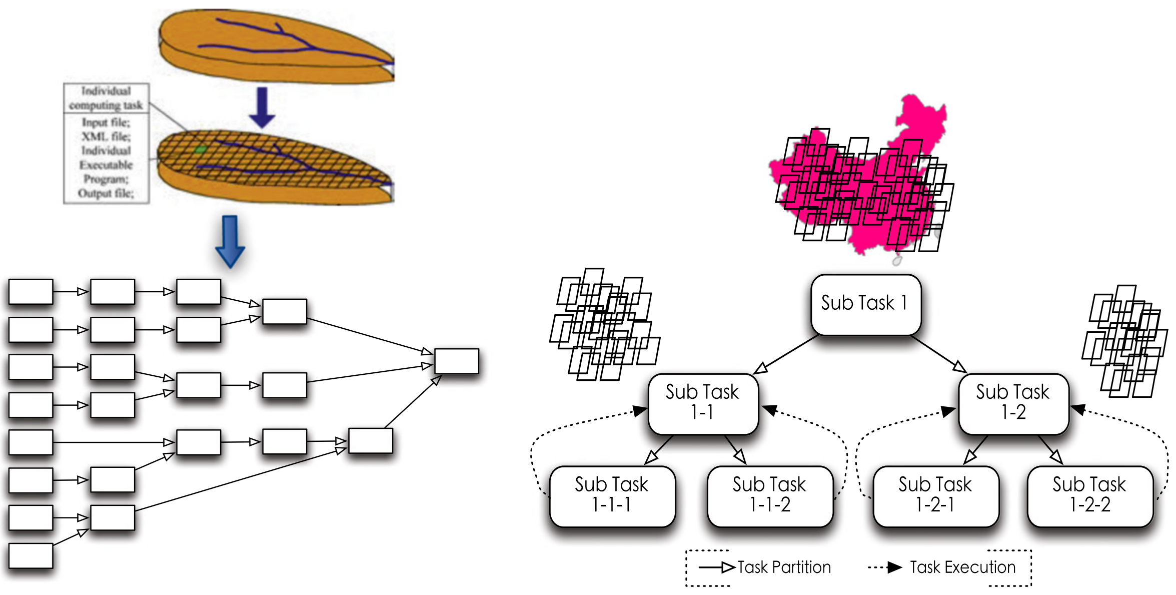
Fig. 5. The data access patterns of RS applications.

3.5. Efficient and productive parallel programming

The state-of-the-art cluster systems are featured with a multi-level hierarchical organization and increasing scale. Efficient and productive programming of these systems will be a challenge, especially in the context of data-intensive RS applications.

Recently, several low-level parallel paradigms are extensively employed for remote sensing image processing in those multi-level hierarchical clusters. OpenMP [32] paradigm is designed for shared-memory and could also be extended to the distributed shared memory clusters. MPI [33] is a message passing interface adopted for exploiting parallelism both within and across computing nodes. With MPI, programmers have to explicitly control the synchronization and communication among a set of processes using message-passing semantics, which is quite complicated. Typically, hybrid paradigms like MPI+OpenMP [34] is employed for exploiting multilevel of parallelism. Wherein, MPI is for message passing across nodes, while OpenMP is in charge of the parallelization inside a single node. In addition, the latest Partitioned Global Address Space (PGAS) [35] programming models could offer a global but partitioned memory address space across nodes, examples are Unified Parallel C (UPC) [36], Co-array Fortran [37], Chapel [38], X10 [39], and Global Arrays [40]. The novelty of PGAS is that the portions of the shared memory may have affinity to a local process or thread for locality of reference. Thereby, PGAS approaches attempt to incorporate the shared memory-programing model with the distributed-memory systems.

However, the knowledge of the detailed hierarchical architecture of parallel system turns is prominent to the efficient programming for the multi-level hierarchy. Generally, the non-experts have to handle both the message passing model for inter-nodes and the



(a) Data dependency among grid tasks.

(b) Relationships among mosaicking tasks.

Fig. 6. The data dependency among a collection of tasks.

shared memory model for intra-node communication. Therefore, these low-level programming paradigms that closely related to the parallel system architecture are bound to rather difficult and trivial. Moreover, in the RS data processing specific point of view, there are different degree of dependencies between the computation and associated RS data. While, these data dependences that vary with different algorithms would probably lead to different data parallelism scheme, data computation modes, data/task partition strategies. With no doubt these would make the parallel programming of RS algorithms much more difficult. Thereafter, a simple but efficient parallel programming for RS applications with less concern of architecture related implementing details is urgently desired.

4. State-of-the-art works of processing RS Big Data

The growing amount of RS datasets is outstripping the current capacity of exploring and interpreting them [12]. The processing of RS "Big Data" has poses both challenges and opportunities for several aspects of the state-of-the-art parallel systems, including system architecture, parallel file system and parallel I/O, programming models, data managing on multilevel memory hierarchy and task scheduling.

4.1. System architectures for data-intensive RS applications: clusters, and clouds

The emergence of RS "Big Data" is revolutionizing the techniques for analyzing and discovering the valuable information from it. From a system point of view, the rapid processing of RS big data with increasing volumes and complexity is greatly challenging the existing systems. Novel advances in system architecture are imperatively needed, especially the inherent scalability of the underlying hardware and software architecture. To be specific, these data-intensive systems could be scaled in a linear manner so as to accommodate processing of RS data at almost any volume. Possibly, for meeting the near real-time processing requirement of some RS applications, an easy configuration of adding extra computing resources is also demanded. For performance efficiency, it is critical for data-intensive platforms to abide by the Move the code to the data principle [41] so as to minimize data movements. Thus, a storage hierarchy of system optimized for data-intensive computing would probably reside data locally to reduce network and system overhead introduced by data transferring. Currently, several available high performance platforms are employed to make sense of these RS Big Data. The most dominant choices of platforms concentrate on, namely, Cluster-Based HPC systems or supercomputers as well as the Cloud platforms.

4.1.1. Cluster-based HPC systems

The cluster platform [9] normally performs a large computational problem by the collaborative work of multiple computers while offering a single-system image. At the present, the cluster platforms are the mainstream architecture for high performance computing. NASA gives the first shot of building a NEX system for global RS data processing on a Beowulf cluster with 16 computers [42]. InforTerra also adopts the cluster platform for the massive imaging auto-processing system "Pixel Factory" [43]. The Headwave company of American also takes advantage of the cluster systems for extensive analysis and interpretation of pre-stack seismic data. Google [44] has build a large cluster system consisting of nearly 1500 ordinary personal computers. This system could offer high-level of data capacity, throughput, and availability by virtue of the software fault tolerance, data backup and optimized system management. However, despite of the abundant computational capacities, to process RS Big Data on these existing cluster-based HPC system still remains quite challenging. The main reason for that is the current cluster systems are compute-intensive oriented. The system architecture and tools are not optimized for the data-intensive applications where data availability is the main concern.

The petascale supercomputers have become the mainstream platforms for large scale scientific computing. These supercomputing system are evolving towards hybrid or even accelerator-based architectures with millions of cores [45]. "Tianhe-2" [46,47] equipped with 2,736,000 Intel Xeon Phi has topped the TOP500 list of world's fastest supercomputers in 2014 at a speed of 33.86 petaflops. Tianhe-1A [48] with hybrid multicore CPU and GPU has ranked the top of the TOP500 at a peak of 2.507 petaflops. Other examples are CPU/GPU system "TITAN" built by Cray at Oak Ridge National Laboratory perform over 10 petaflops [49], the BlueGene/Q supercomputer "Sequoia" [50] with a LINPACK performance of

16.32 petaflops, and the "K computer" [51] of Japan with 10.51 petaflops. Moreover, All of these supercomputers are equipped with high performance Infiniband computing network with high bandwidth.

In spite of the tremendous computation capacity and outstanding scalability, the cluster systems or petascale supercomputers are not good at loading, transferring and processing extremely large volume of data. The main reason account for that is the data locality optimization of the data storage architecture is not under consideration in supercomputers where computation is the main concern. Actually, we are currently witnessing a transition to systems that are arranged in multiple hierarchical levels. These systems are trend to have a higher dimensional connection topology, and multilevel storage architecture as well. For example, the BlueGene/Q system is composed of several islands where non-blocking communication is only available within an island. The Gordon system [52] is designed for the data-centric applications. It has a five-level memory hierarchy (excluding caches) with a node-local shared memory, a virtual shared memory within a super node, a distributed memory, the flash memory (SSDs) and disk arrays. Wherein, with multi-level structured storage, the requested data would be much more possibly reside in local memory, local flash or even local disks. However, for performance efficiency it is critically important to take data locality into account [53]. Programming for these multiple levels of locality and routes for a certain dimensionality is anything but easy.

4.1.2. Cloud platforms

Cloud computing [54] has been one of the most robust Big Data techniques. By virtue of virtualization technologies, the supercomputing are made more accessible and affordable. The computing infrastructures are virtualized like true physical computer but with the flexible processors, memory and even disk size. The cloud computing not only delivers application and software as service (Saas), but also extend to the infrastructure and platform as service. For example, Amazon EC2 provides infrastructure as a service (Iaas), Google AppEngine and Microsoft Azure offer platform as a service (Paas). Following this way, the cloud computing has led to the pay-as-you-go computing and give the illusion of infinite resources. Another bonus of Cloud is the Cloud storage which provides a tool for storing Big Data with good scalability.

Hadoop and ecological environment [55] has become one of the most successful infrastructure for Cloud and also for Big Data computing. By open source implementation of Map/Reduce framework [56], Hadoop enable distributed, data-intensive and parallel applications. Inside the Hadoop ecology, the distributed File System (HDFS) is a large distributed file system with strategic layouts and data replication for fault tolerance and better accessing performance. Recently, Yahoo has deployed its search engine on the Hadoop cluster, Facebook and eBay also develop its large applications at a scale of exabyte with Hadoop. In addition, the Hadoop–GIS [57] system for large-scale spatial data processing, search and accessing is also build upon the Hadoop system.

4.2. Parallel file systems and database

Scaling from terabytes to petabytes or even exabytes, Big Data has posed unprecedented challenges for the efficient data storing and accessing. The storages for Big Data should provide not only huge storage capacity but also high I/O throughput and outstanding scalability as well. However, the performance gap between I/O and computing is gradually widening by further growth of computing capability, especially in cluster scenario with thousands of cores. In this case, the data processing has to wait a plenty of CPU cycles for data accessing [58]. In particular, in the scenario of massive RS data processing, the intensive irregular I/O patterns performed by RS data accessing are leading to even worse I/O situations and wider I/O gap. Obviously, the fast and easy access of the Big Data also means fully or partially break the restraint: CPU-heavy but I/O-poor. Currently, the existing mainstream techniques for Big Data storing and managing would include parallel file systems and NoSQL Databases.

4.2.1. Parallel file systems (PFSs) for big data storage

The traditional I/O systems may no longer be the wise choice to meet the intensive high performance I/O requirements in the modern large-scale cluster systems or even supercomputers. While, parallel file systems with scalable parallel I/O are widely employed in HPC systems where I/O emerges as the main bottleneck [29] especially when the amount of data is extremely large. Through data stripping, the extremely large amount of data are typically striped and distributed across hundreds of I/O devices (disks) or storage nodes blockwise. Taking advantage of this, parallel file systems are capable of storing and managing massive data ("big data") at a storage capacity of over petabytes. Meanwhile, the data-intensive applications like large-scale RS applications could access requested data from different physical storages or disks simultaneously. Following this way, the concurrent I/O through different data access connections would give rise to an extremely high bandwidth of aggregated I/O. The PFS approach is widely regarded as one of the good options for "Big Data" storing.

Recently, great efforts have been towards parallel file systems, such as OrangeFS [59], PVFS [29], Lustre [60], PanFS [61] and GPFS [62]. Lustre, the most widespread object-oriented parallel file system that adopted in the world's TOP 500 supercomputers. It offers huge data storage capacity up to petabytes, and a relatively high aggregated I/O bandwidth of hundreds of gigabytes per second. Benefiting from its outstanding scalability, Lustre could be easily scaled from hundreds to tens of thousands of computing nodes, which perform large amount of I/O connections and accessing concurrently. OrangeFS the advancing branch of PVFS [29] is an open source parallel file system for scientific computing. It provides non-contiguous I/O, distributed metadata management as well as configurable physical data layout. With the distributed metadata management, the metadata is separated from the actual data and spread across storage severs to eliminate the metadata-accessing bottleneck. In addition, OrangeFS also provides non-contiguous I/O patterns and the ability to integrate some application-specific data layout schemes. Taking advantages of these features, OrangeFS could easily obtain high throughput I/O. HDFS [63] is the primary distributed storage system for Hadoop with excellent portability and fault-tolerance. Facebook and Yahoo have already adopted it for storing and managing big Internet data. In those aforementioned PFSs, a great number of storage devices attached to computing nodes are virtually converged as a single big data image. Eventually, an extremely large storage capacity and also a high degree of I/O parallelism could be achieved.

However, only one tenth of the peak I/O performance of PFSs could be actually achieved by most of the scientific applications which perform small non-contiguous I/O [29,27]. The mismatch between the physical data layout over storages and the expected I/O characteristics or patterns of applications [30] would probably be the reason. Scientific computing like RS data processing often requires many non-contiguous accessing of small data fragments. Most of these non-contiguous or even irregular I/O patterns vary across applications in both the size and frequency of data requests. So, it is of significant importance for file systems to understand the I/O characterization of applications. Unfortunately, the applications and PFSs are traditionally designed separately in the consideration of better transparency. The thing is that PFSs typically adopt a simple data striping method with a fixed striping size, but almost know nothing about the I/O characterization of applications. Lacking information of each other would probably lead to the mismatch between data access patterns from client side and the physical data layouts inside PFSs. This mismatch would inevitably introduce workload imbalance among I/O servers and eventually negative effect of the overall I/O performance [64].

The physical data layout policies of PFSs are critical to the data locality and workload balance among I/O servers. PFSs like OrangeFS are trends to provide extra data striping or data layout options for some specific I/O workload. Even though, these fixed data layouts using simple and fixed data striping parameters may benefit some applications with a few specific I/O patterns, but may not be the cases for others [64]. Thereby, it is anything but easy to find a data layout optimized for all data-intensive scientific applications, especially for those with irregular or even variable I/O patterns. Accordingly, it is critically important for PFSs designers to understand the I/O characterization of large scientific computing. Some efforts are laid towards data layout optimization in current PFSs. Data replication [65] technique could reorganize the data layout through creating data replicas across I/O server for better locality but at the penalty of storage capacity. Song [64] put forward a segment-level adaptive data layout scheme for variable I/O pattern. Different stripe sizes are adopted for different data segments. Apart from the above works, Wang and Ma [26] have proposed a specific parallel file system with application-aware data layout for managing RS Big Data. These methods could offer the physical data layout exactly needed by RS applications for better locality and an overall I/O performance optimization.

4.2.2. NoSQL database for big data

NoSQL (Not Only SQL) [66] is the most popular but novel database designed for distributed managing of big unstructured or non-relational data. It is an inheritance but also development of the traditional database, since it breaks the rigidity of relational data model without avoiding SQL. The specific approaches that employed include the key-value storage for better scalability and schema-free for greater flexibility. Hbase [67] one of the most popular NoSQL databases, SQLstream a Big Data analyzing tool still adopts SQL for a more reliable and simple query. The other well known NoSQL databases are Apache Cassandra [68], Google BigTable [69], SimpleDB [70] and etc.

Essentially, it is a distributed database technique. Benefiting of the distributed data storing and managing architecture, NoSQL database could offer larger storage capacity, higher concurrency as well as better scalability and availability. There are two types of distributed architectures: Master–Slave [69] and P2P [68]. Within Master–Slave architecture, master node in charges of the whole distributed system and monitors the health of the distributed slave database nodes. While, the slave nodes are data storage nodes that hold a local data indexing table. These nodes are responsible for handling local data requests within the data storage region assigned by master node. This kind of architecture has simple designing but may give rise to single point of failure of master node. The typical examples are BigTable and Hbase. For P2P structured databases, there are no master nodes. The distributed storage nodes are logically organized in a ring structure by virtue of distributed hashing algorithm. Each of the distributed nodes is self-managing storage node. Comparatively, the P2P structure has much better scalability and self-coordination. Actually, Cassandra and Dynamo [71] are typical P2P structured databases.

In NoSQL databases, huge amount of data are organized with a simple key-value data model for simpler data operations and better concurrency of data accessing. With key-value model, the key based hashing mechanism is employed for data mapping and locating so as to conduct fast retrieval from tremendous data. In addition, some of these data are also organized as column families within key-column model (examples: Hbase and Cassandra), where data column is the basic data storage unit. Simply, it would adopt a multi-dimensional mapping method to locate physical data. Most of these data indexing approaches are used for local data mapping inside a data node. While, the global indexing is typically adopting hashing method to directly locate the exact node that data hosted. Following this way, NoSQL databases could be able to provide data retrieval and accessing with extremely high concurrency and parallelism.

Currently, this new NoSQL paradigm is also applied in remote sensing domain for the managing of unstructured massive RS image data as well as GIS (Geographical Information System) vector data. Generally, NoSQL databases like HBase are employed for the constructing and storing of distributed R tree or B$^+$ tree structured time-space indexing of these tremendous RS data, while the PFSs like HDFS are adopted for the actual physical data file storing. Following this approach, efficient data retrieval together with high throughput RS data accessing could be offered. Zhifeng [72] employs HBase to build database for distributed remote sensing image data.

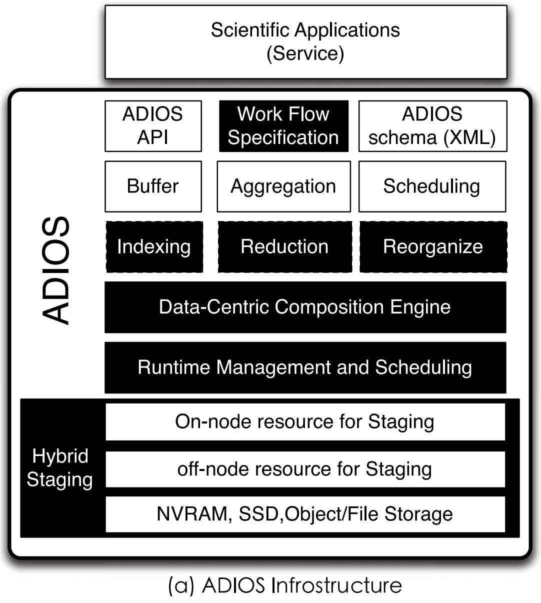
4.3. Parallel I/O optimizations

A fair number of parallel I/O libraries and optimization approaches have been exploited for supporting non-contiguous I/O. ROMIO [73] a famous and efficient implementation of MPI-IO [74] interface has add some optimizing strategy for non-contiguous data accesses. Two-phase I/O [75] introduces an extra exchanging phase for Gather–Scatter communication. The non-contiguous I/O requests could be rearranged into a contiguous one for exploring better data locality but also brings in extra network communication. Data sieving [76] normally loads a large chunk of data and sieves out non-requested ones in one single I/O. This approach could apparently reduce I/O calls but at the penalty of reading extra data. In particular, the collective I/O [77] merges small non-contiguous requests into a large contiguous one by combining the data sieving with two-phase I/O. This approach could greatly reduce the I/O requests while achieving a better ratio of actual data accessing. These approaches are also employed in the RS applications to reduce I/O operations for improved I/O performance, especially for the irregular RS data accessing.

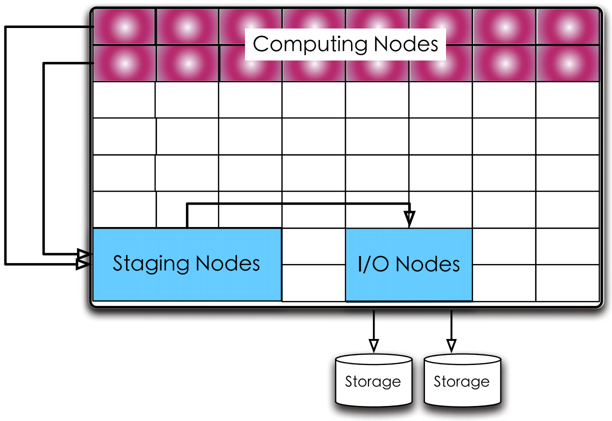
Apart from above approaches, ADIOS [78] depicted in Fig. 7 is essentially a componentization of different I/O transport methods, which is scalable and portable. By virtue of high-level I/O interfaces, ADIOS allows application scientists to choose the I/O methods fit the parallel infrastructures most with little modification and concerning for details of system I/O. The most attractive feature of ADIOS is data staging. The runtime of ADIOS could derive staging nodes and I/O nodes respectively dedicated to asynchronous memory-to-memory transferring between two applications and data I/O. Therefore, ADIOS would possibly benefit the multi-stage RS workflows where exist frequent data transferring among stages.

4.4. Programming models

Efficient and productive programming on large scale distributed systems with multilevel hierarchy will be extremely challenging, especially in the context of data-intensive applications just like RS data processing. The main reason is that the programming modes widely adopted in conventional cluster-based distributed platforms or supercomputers are typically machine dependent. With these low-level programming models, programmers have to manually control the machine related implementing details, include task partition, processing and node communications. Therefore, the data-intensive platforms require a machine independent programming model. With this approach, the applications are expressed in high-level of semantics. The runtime is


Scientific Applications (Service)
ADIOS API
Work Flow Specification
ADIOS schema (XML)
Buffer
Aggregation
Scheduling
ADIOS
Indexing
Reduction
Reorganize
Data-Centric Composition Engine
Runtime Management and Scheduling
Hybrid Staging
On-node resource for Staging
off-node resource for Staging
NVRAM, SSD,Object/File Storage
(a) ADIOS Infrostructure

(a) ADIOS infrostructure.



(b) ADIOS: Using staging as an burst buffer.

Fig. 7. The ADIOS adaptable parallel I/O system.

also demanded for transparent controls of load balancing, communications and data movements in large cluster system. Therefore, a programming model of high-level abstraction is urgently demanded for automatically taking care of the machine related parallel implementing details. The effective programming requirement in data-intensive systems is programmer could easily program a data-intensive application with no experience in parallel programming. There are three categories of programming models, including low-level programming models, skeletal parallel programming as well as generic parallel programming.

4.4.1. *Low-level programming models*

A number of low-level programming paradigms are developed for this hierarchical architecture. The OpenMP [79] model is used for parallelization of shared-memory and distributed shared-memory clusters. The Message Passing model (MPI)[1] is employed within and across the nodes of clusters. The hybrid programming paradigm MPI+OpenMP [80] exploits multiple levels of parallelism: the OpenMP model is used for parallelization inside the node and MPI is for message passing among nodes. The advantage of these programming models is that a relatively high parallelism and overall performance could be achieved through careful specific designing. Nevertheless, these low-level programming are machine dependent, manual control of computing and communication details as well as the significant tuning are required. Accordingly, to develop parallel programs for hierarchical cluster architectures with low-level programming models and APIs, for example MPI and OpenMP, are still difficult and error-prone.

4.4.2. *Skeletal parallel programming*

The skeletal parallel programming with higher-level patterns is adopted by researchers to simplify parallel programming. The parallel implementing details are already abstracted in the skeleton and would be automatically executed across hundreds to thousands of machines on large distributed systems. The eSkel library provides parallel skeletons for C language. It can generate efficient parallel programs but leave users to handle low-level API with many MPI-specific implementation details. Muesli [81] offers polymorphic C++ skeletons with high level of abstraction and simple APIs. The programs can be produced by construction of abstract skeleton classes and deal with data transmission via a distributed container. However, it suffers a large overhead paid for runtime polymorphic virtual function calls. Google's MapReduce model, which supports Map and Reduce operations for distributed data processing in a cluster, is a simple yet successful example of parallel skeletons. The programs could be automatically implemented across thousands of nodes in parallel while taking charge of the data partition, communications among nodes and other execution details. Following this way, the no-experts programmers of parallel computing could easily program efficient codes for large distributed platforms.

4.4.3. *Generic parallel programming*

The generic programming approach uses templates to program generically. This concept has been exploited efficiently in the Standard Template Library (STL) [82], which has been extensively applied due to its convenient generic features and efficient implementations. Furthermore, the polymorphism is resolved at compile time because of its usual type genericity. The QUAFF [83] skeleton-based library offers generic algorithms. It relies on C++ templates to resolve polymorphism by means of type definitions processed at the compile time. It reduces the runtime overhead of polymorphism to the strict minimum while keeping a high-level of expressivity and readability.

Low-level parallel programming models like MPI, OpenMP and MPI+OpenMP are extensively employed for remote sensing image processing. The aforementioned projects provide high-level pattern for parallel programming. However, these implementations did not develop their research for the massive remote sensing datasets with multi-band image data and complex structured metadata. Moreover, they did not handle the different dependences between computation and data of RS algorithms. In this situation, it remains a big challenge to program effective massive remote sensing data processing algorithms productively.

[1] MPI-3: A Message-Passing Interface Standard Version 3.0, http://www.mpi-forum.org, November 12, 2010.

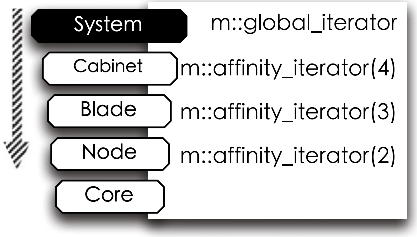


Fig. 8. The Hierarchical Arrays (HA) in DASH project.

4.5. *Data management across multilevel memory hierarchy*

For performance efficiency, it is imperative to take data locality into account, especially in data-intensive scenario of processing RS Big Data. However, the state-of-the-art parallel systems are arranged in multiple hierarchical levels, higher dimensional network topology, and a multilevel storage (memory hierarchy). The loading, managing of these Big Data on the multilevel memory hierarchy in multiple hierarchical parallel systems turn out to be rather challenging. Since the exploring of multiple levels of locality is anything but easy. On one hand, some skeletal parallel programming libraries offer distributed data structure for data managing on distributed memories. Wherein, SkeTo [84] and Muesli [81] provide multiple distributed data types like btree, array and list on top of MPI with both local and global view. On the other side, many latest programming models are capable of offering global data managing on multilevel distributed memories, like PGAS [35], UPC [85] and Chapel [86]. These PGAS-based approaches provide global memory view and explicit locality control. But there is no straightforward way to control multilevel data placement for exploring multilevel locality. In addition, the current DASH project[2] is developing the C++ template library named Hierarchical Arrays (HA) (Fig. 8) for distributed data structures with support for hierarchical locality for HPC and data-driven science. HA focuses on the data structures and their layout as well the efficient access and processing by optimized algorithms. In HA it will be possible to exploit multilevel locality by extending the iterator concept with iterators for multiple hierarchical levels. It is also important to note that HA is not intended to replace established programming mechanisms, such as MPI but rather to extend them with functionality for simplified handling large amounts of data (both regular as well as irregularly structured).

4.6. *Task scheduling*

Some RS applications could be modeled as parallel tasks. Data dependencies exist among tasks. List scheduling heuristic is the most frequently used scheduling algorithm of the research on traditional parallel task scheduling. List scheduling [87–89] is a class of scheduling heuristics in which tasks are assigned priorities and placed in a list ordered in decreasing magnitude of priority. Different task scheduling heuristics are proposed, e.g., DLS [87], ISH [90], DSH [89]. Conventional list scheduling heuristics do not consider resource reservation problems. The other choices of scheduling solutions with trade-offs between time complexity and quality, including clustering scheduling and the task-duplication based scheduling. The clustering scheduling approaches including EZ [91] and DCP [92] aim at reducing communication times and take advantages of more processor by merging cluster of task nodes.

[2] DASH project: under DFG programme “Software for Exascale Computing—SPPEXA”, http://www.dash-project.org.

The scheduler is responsible for mapping task to concrete processors, job execution and monitoring. Swift [93] integrates Karajan [93] to provide concise specification and scheduling for task graph, also incorporates Falkon [94] a lightweight execution framework for efficient dispatch of large numbers of tasks in Grids. DAGMan [95] offers workflow engine to schedule Condor jobs expressed as DAGs in distributed environment. In fact it is too closely tied to the Condor system and lacks generic programming APIs for easy use. Google’s Pregel [96] a distributed computing framework with super steps and message passing is suitable for large-scale graph processing.

5. Conclusion

With the advent of high resolution earth observation era give birth to the explosive growth of remote sensing (RS) data. The proliferation of data also gives rise to the increasing complexity of RS data, like the diversity and higher dimensionality characteristic of the data. RS data are regarded as RS “Big Data”. The large-scale environmental monitoring and researches are exploiting regional to global covered multi-temporal and multi-sensor RS data for processing. Large remote sensing applications overwhelmed with massive remote sensing data are regarded as typical data-intensive issues. In spite of the enormous computational power, the cluster-based HPC systems still remain considerably challenging with RS “Big Data” issues. These issues including the difficulties in storing massive complex RS data, intensive irregular data access patterns, managing RS “Big Data” on multilevel memory hierarchy, optimal scheduling of large amount of dependent tasks as well as the efficient programming of RS applications.

There is no doubt that the existing techniques and systems are so limited to solve the real RS Big Data problems completely. Fortunately, we are witnessing the coming technological leapfrogging. As we have discussed above, we do have drawn great effort for optimizing or even revolutionizing the existing technique, tools or system. These works include: (1) Some supercomputers and Cloud computing platforms optimized for data-intensive loads; (2) Parallel file systems and databases take the data availability and locality as the main concern; (3) The data managing tools for memory data placement controlling for multilevel data locality; (4) Task scheduling focusing on large amount of dependent tasks and considering data availability.

Opportunities are always followed by challenges. In the prospect of the further requirements in the near future, the demand for real-time processing, on-demand processing as well as the in-transit processing of standard RS data products also poses great opportunities.

Acknowledgments

Dr. Lizhe Wang’s work is supported by the National High Technology Research and Development Program of China (“863” Program) (No. 2013AA12A301) and the National Natural Science Foundation of China under Grant No. 61361120098. Yan Ma’s work is supported by the Preferred Foundation of Director of Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences, China.

References

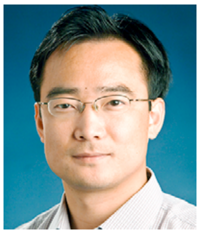
[1] N. Skytland, Big data: What is nasa doing with big data today, Open. Gov open access article 2012.
[2] P. Gamba, Peijun Du, C. Juergens, D. Maktav, Foreword to the special issue on human settlements: A global remote sensing challenge, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 4 (1) (2011) 5–7.
[3] OGC-OpenGIS Consortium et al. The opengis abstract specification-topic 7: The earth imagery case, 1999.

[4] Huadong Guo, Lizhe Wang, Fang Chen, Dong Liang, Scientific big data and digital earth, Chin. Sci. Bull. 59 (12) (2014) 1047–1054.
[5] A. Rosenqvist, M. Shimada, B. Chapman, K. McDonald, G. De Grandi, H. Jonsson, C. Williams, Y. Rauste, M. Nilsson, D. Sango, M. Matsumoto, An overview of the jers-1 sar global boreal forest mapping (gbfm) project, in: Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International, Vol. 2, Sept. 2004, pp. 1033–1036.
[6] Meixia Deng, Liping Di, Genong Yu, A. Yagci, Chunming Peng, Bei Zhang, Dayong Shen, Building an on-demand web service system for global agricultural drought monitoring and forecasting, in: Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International, July 2012, pp. 958–961.
[7] Van Zyl, Jakob, Application of satellite remote sensing data to the monitoring of global resources, in: Technology Time Machine Symposium (TTM), IEEEl, 2012, p. 1.
[8] Reagan Moore, Thomas A. Prince, Mark Ellisman, Data-intensive computing and digital libraries, Commun. ACM 41 (11) (1998) 56–62.
[9] Antonio J. Plaa, Chein-I. Chang, High Performance Computing in Remote Sensing, Chapman & Hall/CRC, 2007.
[10] Yan Ma, Lizhe Wang, Albert Y. Zomaya, Dan Chen, Rajiv Ranjan, Task-tree based large-scale mosaicking for remote sensed imageries with dynamic dag scheduling, IEEE Trans. Parallel Distrib. Syst. 99(PrePrints) (2013) 1.
[11] Yuehu Liu, Bin Chen, Hao Yu, Yong Zhao, Zhou Huang, Yu Fang, Applying gpu and posix thread technologies in massive remote sensing image data processing, in: Geoinformatics, 2011 19th International Conference on, June 2011, pp. 1–6.
[12] R.T. Kouzes, G.A. Anderson, S.T. Elbert, I. Gorton, D.K. Gracio, The changing paradigm of data-intensive computing, Computer 42 (1) (2009) 26–34.
[13] Hesham El-Rewini, Theodore G. Lewis, Hesham H. Ali, Task Scheduling in Parallel and Distributed Systems, Prentice-Hall, Inc., 1994.
[14] Behrooz A. Shirazi, Krishna M. Kavi, Ali R. Hurson (Eds.), Scheduling and Load Balancing in Parallel and Distributed Systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.
[15] Mathieu Fauvel, Jon Atli Benediktsson, John Boardman, John Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Paolo Gamba, A. Gualtieri, M. Marconcini, et al., Recent advances in techniques for hyperspectral image processing, Remote Sens. Environ. (2007) 1–45.
[16] Antonio J. Plaza, Special issue on architectures and techniques for real-time processing of remotely sensed images, J. Real-Time Image Process. 4 (3) (2009) 191–193.
[17] Russell G. Congalton, A review of assessing the accuracy of classifications of remotely sensed data, Remote Sens. Environ. 37 (1) (1991) 35–36.
[18] F.F. Sabins, Remote Sensing: Principles and Interpretation, Freeman, 1978.
[19] C.L. Philip Che, Chun-Yang Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, Inform. Sci. 275 (2014) 314–347.
[20] I. Gorton, Software architecture challenges for data intensive computing, in: Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on, Feb. 2008, pp. 4–6.
[21] R. Pfister, K. Wichmann, New paradigm for search and order in eosdis, in: Geoscience and Remote Sensing Symposium, 2000. Proceedings. IGARSS 2000. IEEE 2000 International, Vol. 3, 2000, pp. 1208–1210.
[22] M. Esfandiari, H. Ramapriyan, J. Behnke, E. Sofinowski, Evolution of the earth observing system (eos) data and information system (eosdis), in: Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on, July 2006, pp. 309–312.
[23] Hampapuram Ramapriyan, Jennifer Brennan, Jeanne Behnke, Managing Big Data: nasa Tackles Complex Data Challenges, 2013.
[24] NASA. Earth science measurements, 2013.
[25] S.F. Oliveira, K. Furlinger, D. Kranzlmuller, Trends in computation, communication and storage and the consequences for data-intensive science, in: High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on, June 2012, pp. 572–579.
[26] L. Wang, Y. Ma, A. Zomaya, R. Ranjan, D. Chen, A parallel file system with application-aware data layout policies in digital earth, IEEE Trans. Parallel Distrib. Syst. PP (99) (2014) 1–1.
[27] J. Worringen, J.L. Traff, H. Ritzdorf, Fast parallel non-contiguous file access, in: Supercomputing, 2003 ACM/IEEE Conference, Nov. 2003, p. 60.
[28] N. Nieuwejaar, D. Kotz, A. Purakayastha, C.S. Ellis, M.L. Best, File-access characteristics of parallel scientific workloads, IEEE Trans. Parallel Distrib. Syst. 7 (10) (1996) 1075–1089.
[29] J. Wu, P. Wyckoff, Dhabaleswar Panda, Supporting efficient noncontiguous access in pvfs over infiniband, in: Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on, Dec. 2003, pp. 344–351.
[30] S. Narayan, J.A Chandy, I/o characterization on a parallel file system, in: Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2010 International Symposium on, July 2010, pp. 133–140.
[31] Yuzhuo Wang, Wei Zhang, Xiuguo Liu, Hydrological watersheds model researching based on digital elevation model, in: Geoinformatics, 2010 18th International Conference on, June 2010, pp. 1–5.
[32] Yang-Suk Kee, Openmp extension to smp clusters, IEEE Potentials 25 (3) (2006) 37–42.
[33] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard Version 3.0, 09 2012. Chapter author for Collective Communication, Process Topologies, and One Sided Communications.
[34] R. Rabenseifner, G. Hager, G. Jost, Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes, in: Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on, Feb. 2009, pp. 427–436.
[35] Filip Blagojevi, Paul Hargrove, Costin Iancu, Katherine Yelick, Hybrid pgas runtime support for multicore nodes, in: Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, PGAS'10, ACM, New York, NY, USA, 2010, pp. 3:1–3:10.
[36] Wei-Yu Chen, C. Iancu, K. Yelick, Communication optimizations for fine-grained upc applications, in: Parallel Architectures and Compilation Techniques, 2005. PACT 2005. 14th International Conference on, Sept. 2005, pp. 267–278.
[37] Robert W. Numrich, John Reid, Co-array fortran for parallel programming, SIGPLAN Fortran Forum 17 (2) (1998) 1–31.
[38] Nan Dun, K. Taura, An empirical performance study of chapel programming language, in: Parallel and Distributed Processing Symposium Workshops Ph.D. Forum (IPDPSW), 2012 IEEE 26th International, 2012, pp. 497–506.
[39] J. Milthorpe, V. Ganesh, A.P. Rendell, D. Grove, X10 as a parallel language for scientific computation: Practice and experience, in: Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International, May, 2011, pp. 1080–1088.
[40] Vinod Tipparaju, Manoj Krishnan, Bruce Palmer, Fabrizio Petrini, Jarek Nieplocha, Towards fault resilient global arrays, Parallel Comput. Arch. Algorithms. Appl. (2008).
[41] Jim Gray, Distributed computing economics, Queue 6 (3) (2008) 63–68.
[42] C.A. Lee, S.D. Gasster, A. Plaza, Chein-I. Chang, Bormin Huang, Recent developments in high performance computing for remote sensing: A review, IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens. 4 (3) (2011) 508–527.
[43] Min CAO, Zhao-liang SHI, Primary study of massive imaging auto-processing system pixel factory, Bull. Surv. Mapp. 10 (2006) 55–58.
[44] Luiz André Barroso, Jeffrey Dean, Urs Holzle, Web search for a planet: The google cluster architecture, IEEE Micro 23 (2) (2003) 22–28.
[45] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, K. Heitmann, Hacc: Extreme scaling and performance across diverse architectures, in: High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for, Nov. 2013, pp. 1–10.
[46] J. Dongarra, Visit to the National University for Defense Technology Changsha, 2013.
[47] Wei Xue, Chao Yang, Haohuan Fu, Xinliang Wang, Yangtong Xu, Lin Gan, Yutong Lu, Xiaoqian Zhu, Enabling and scaling a global shallow-water atmospheric model on tianhe-2, in: Parallel and Distributed Processing Symposium, 2014 IEEE 28th International, 2014, pp. 745–754.
[48] Min Xie, Yutong Lu, Kefei Wang, Lu Liu, Hongjia Cao, Xuejun Yang, Tianhe-1a interconnect and message-passing services, IEEE Micro 32 (1) (2012) 8–20.
[49] Lal Shimpi, Anand, Inside the Titan Supercomputer, 2013.
[50] S. Habib, V. Morozov, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. Insley, D. Daniel, P. Fasel, N. Frontiere, Z. Lukic, The universe at extreme scale: Multi-petaflop sky simulation on the bg/q, in: High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for, Nov. 2012, pp. 1–11.
[51] M. Yokokawa, The k computer and its application, in: Networking and Computing (ICNC), 2012 Third International Conference on, Dec. 2012, pp. 21–22.
[52] AM. Caulfield, L.M. Grupp, S. Swanson, Gordon: An improved architecture for data-intensive applications, IEEE Micro 30 (1) (2010) 121–130.
[53] B. Dally, Power, programmability, and granularity: The challenges of exascale computing, in: Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International, 2011, p. 878.
[54] M.N.O. Sadiku, S.M. Musa, O.D. Momoh, Cloud computing: Opportunities and challenges, IEEE Potentials 33 (1) (2014) 34–36.
[55] A. Kala Karun, K. Chitharanjan, A review on hadoop: Hdfs infrastructure extensions, in: Information Communication Technologies (ICT), 2013 IEEE Conference on, 2013, pp. 132–137.
[56] Jeffrey Dean, Sanjay Ghemawat, Mapreduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[57] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, Joel Saltz, Hadoop gis: a high performance spatial data warehousing system over mapreduce, Proc. VLDB Endow. 6 (11) (2013) 1009–1020.
[58] Rengan Xu, M. Araya-Polo, B. Chapman, Filesystem aware scalable i/o framework for data-intensive parallel applications, in: Parallel and Distributed Processing Symposium Workshops Ph.D. Forum (IPDPSW), 2013 IEEE 27th International, 2013, pp. 2007–2014.
[59] Shuangyang Yang, Walter B. Ligon III, Elaine C. Quarles, Scalable distributed directory implementation on orange file system, in: The USENIX Conference on File and Storage Technologies (FAST 2011), the 9th Conference on, 2011.
[60] Tiezhu Zhao, V. March, Shoubin Dong, S. See, Evaluation of a performance model of lustre file system, in: ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual, 2010, pp. 191–196.
[61] Hong Tang, A. Gulbeden, Jingyu Zhou, W. Strathearn, Tao Yang, Lingkun Chu, The panasas activescale storage cluster - delivering scalable high bandwidth storage, in: Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference, Nov. 2004, pp. 53–53.
[62] R. Jain, P. Sarkar, D. Subhraveti, Gpfs-snc: An enterprise cluster file system for big data, IBM J. Res. Dev. 57 (3/4) (2013) 5:1–5:10.
[63] N.S. Islam, M.W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, High performance rdma-based design of hdfs over infiniband, in: High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for, Nov. 2012, pp. 1–12.

[64] Huaiming Song, Yanlong Yin, Xian-He Sun, R. Thakur, S. Lang, A segment-level adaptive data layout scheme for improved load balance in parallel file systems, in: Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, 2011, pp. 414–423.
[65] Jiaying Zhang, Peter Honeyman, Replication control in distributed file systems, CITI Technical Report 04-01, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, USA, Apr. 2004.
[66] Jaroslav Pokorny, Nosql databases: A step to database scalability in web environment, in: Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, iiWAS'11, ACM, New York, NY, USA, 2011, pp. 278–283.
[67] M.N. Vora, Hadoop-hbase for large-scale data, in: Computer Science and Network Technology (ICCSNT), 2011 International Conference on, Vol. 1, Dec. 2011, pp. 601–605.
[68] Avinash Lakshman, Prashant Malik, Cassandra: A decentralized structured storage system, SIGOPS Oper. Syst. Rev. 44 (2) (2010) 35–40.
[69] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, Bigtable: A distributed storage system for structured data, in: USENIX Symposium on Operating Systems Design and Implementation (OSDI 06), 7th, 2006, pp. 205–218.
[70] Andre Calil, Ronaldo dos Santos Mello, Simplesql: A relational layer for simpledb, in: Proceedings of the 16th East European Conference on Advances in Databases and Information Systems, ADBIS'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 99–110.
[71] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Kakulapati, Dynamo: Amazon's highly available key-value store, in: Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles, SOSP'07, ACM, New York, NY, USA, 2007, pp. 205–220.
[72] Zhifeng Xiao, Yimin Liu, Remote sensing image database based on nosql database, in: Geoinformatics, 2011 19th International Conference on, June 2011, pp. 1–5.
[73] Rajeev Thakur, William Gropp, Ewing Lusk, On implementing mpi-io portably and with high performance, in: Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems, IOPADS'99, ACM, New York, NY, USA, 1999, pp. 23–32.
[74] Ding Yong Hong, Ching-Wen You, Yeh Ching Chung, An efficient mpi-io for noncontiguous data access over infiniband, in: Parallel Architectures,Algorithms and Networks, 2005. ISPAN 2005. Proceedings. 8th International Symposium on, Dec. 2005, p. 6.
[75] Y. Tsujita, K. Yoshinaga, A. Hori, M. Sato, M. Namiki, Y. Ishikawa, Multithreaded two-phase i/o: Improving collective mpi-io performance on a lustre file system, in: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on, Feb. 2014, pp. 232–235.
[76] Yin Lu, Yong Chen, Prathamesh Amritkar, Rajeev Thakur, Yu Zhuang, A new data sieving approach for high performance i/o, Future Inf. Technol. Appl. Serv. (2012) 111–121.
[77] Phillip M. Dickens, Rajeev Thakur, Evaluation of collective i/o implementations on parallel architectures, J. Parallel Distrib. Comput. 61 (8) (2001) 1052–1076.
[78] Jay F. Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, Chen Jin, Flexible io and integration for scientific codes through the adaptable io system (adios), in: Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments, CLADE'08, ACM, New York, NY, USA, 2008, pp. 15–24.
[79] Yang-Suk Kee, Openmp extension to smp clusters, IEEE Potentials 25 (3) (2006) 37–42.
[80] R. Rabenseifner, G. Hager, G. Jost, Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes, in: Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on, Feb. 2009, pp. 427–436.
[81] P. Ciechanowicz, H. Kuchen, Enhancing muesli's data parallel skeletons for multi-core computer architectures, in: High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on, Sept. 2010, pp. 108–113.
[82] N. Pataki, Z. Porkolab, Extension of iterator traits in the standard template library, in: Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on, Sept. 2011, pp. 911–914.
[83] J. Falcou, J. Srot, T. Chateau, J.T. Laprest, Quaff: Efficient c++ design for parallel skeletons, Parallel Comput. 32 (7) (2006) 604–615.
[84] Y. Karasawa, H. Iwasaki, A parallel skeleton library for multi-core clusters, in: Parallel Processing, 2009. ICPP'09. International Conference on, Sept. 2009, pp. 84–91.
[85] Max T. Billingsley III, Beth R. Tibbitts, Alan D. George, Improving upc productivity via integrated development tools, in: Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, PGAS'10, ACM, New York, NY, USA, 2010, pp. 8:1–8:9.
[86] B.L. Chamberlain, D. Callahan, H.P. Zima, Parallel programmability and the chapel language, Int. J. High Perform. Comput. Appl. 21 (3) (2007) 291–312.
[87] G.C. Sih, E.A Lee, A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures, IEEE Trans. Parallel Distrib. Syst. 4 (2) (1993) 175–187.
[88] Jing-Jang Hwang, Yuan-Chieh Chow, Frank D. Anger, Chung-Yee Lee, Scheduling precedence graphs in systems with interprocessor communication times, SIAM J. Comput. 18 (2) (1989) 244–257.
[89] B. Kruatrachue, T. Lewis, Grain size determination for parallel processing, IEEE Softw. 5 (1) (1988) 23–32.
[90] Hesham El-Rewini, Theodore G. Lewis, Hesham H. Ali, Task Scheduling in Parallel and Distributed Systems, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
[91] Vivek Sarkar, Partitioning and Scheduling Parallel Programs for Multiprocessors, MIT Press, Cambridge, MA, USA, 1989.
[92] Yu-Kwong Kwok, I. Ahmad, Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors, IEEE Trans. Parallel Distrib. Syst. 7 (5) (1996) 506–521.
[93] Yong Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, M. Wilde, Swift: Fast, reliable, loosely coupled parallel computation, in: Services, 2007 IEEE Congress on, July 2007, pp. 199–206.
[94] I. Raicu, Yong Zhao, Catalin Dumitrescu, I. Foster, M. Wilde, Falkon: a fast and light-weight task execution framework, in: Supercomputing, 2007. SC'07. Proceedings of the 2007 ACM/IEEE Conference on, Nov. 2007, pp. 1–12.
[95] Thomas Tschager, Heiko A. Schmidt, Dagwoman: Enabling dagman-like workflows on non-condor platforms, in: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, SWEET'12, ACM, New York, NY, USA, 2012, pp. 3:1–3:6.
[96] Grzegorz Malewicz, Matthew H. Austern, Aart J.C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski, Pregel: A system for large-scale graph processing, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD'10, ACM, New York, NY, USA, 2010, pp. 135–146.

Yan Ma is an assistant Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS), Beijing, China. Her research interests include high performance geo-computing and parallel remote sensing image processing.

Haiping Wu received the B.S. in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 1994, respectively and the M.S. and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 1997 and 2006. He was an Assistant Professor with the computer center and department of computer science, Tsinghua University, from 1997 to 2007, where he became an associate professor in 2005. He was an associate professor with division of information, Graduate School of Tsinghua University at ShenZhen, from 2007 to 2009. He became the associate director of the Center for Earth System Science, Tsinghua University, Beijing, in 2010. He has authored/coauthored two books and over 20 publications in the field of high performance computing. His present interests include Earth System Modeling, high performance computing and global environmental change.



Lizhe Wang is a Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS), Beijing, China and a "ChuTian" Chair Professor at School of Computer Science, China University of Geosciences, Wuhan, China. He is a Fellow of IET and Fellow of BCS.

Bormin Huang received his M.S.E. in aerospace engineering from the University of Michigan, Ann Arbor, in 1992, and his Ph.D. in the area of satellite remote sensing from the University of Wisconsin-Madison in 1998. He is a Research Scientist of the Space Science and Engineering Center at the University of Wisconsin-Madison, a Honorary International Chair Professor at the National Taipei University of Technology (Taiwan), a Guest Professor at the University of Las Palmas de Gran Canaria (Spain), an Adjunct Professor at Harbin Institute of Technology (China), a Chair Professor at Xidian University (China), etc. He is a Fellow of SPIE - The International Society of Optics and Photonics.

Dr. Huang has been the Lead Chair for the SPIE Conference on Satellite Data Compression, Communications, and Processing since 2005, for the SPIE Europe Conference on High Performance Computing in Remote Sensing since 2011, and for the IEEE International Workshop on Parallel and Distributed Computing in Remote

Sensing, in conjunction with the IEEE International Conference on Parallel and Distributed Systems, since 2011. He is serving as an Associate Editor (in the areas including high performance computing) for the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, and for the Journal of Applied Remote Sensing (in the areas also including high performance computing). He also serves as a Guest Editor for Special Section on High Performance Computing in Remote Sensing, Journal of Applied Remote Sensing.

He was the PI of a NASA-funded project for accelerating the CRTM and RRTMG radiative transfer models on GPUs, the PI of a NOAA-funded project for accelerating the Weather Research and Forecasting (WRF) model on GPUs, etc. He has led his GPU team and guided numerous scholars and students at various universities to publish over 60 HPC-related publications in the areas of geoscience and remote sensing in the past three years. His GPU applications included weather forecast modeling, radiative transfer modeling, earthquake simulation, Tsunami simulation, microwave scattering of vegetation, hyperspectral unmixing, satellite data compression, communication, image processing, etc. One of the significant works by his GPU team was recently cited in NVIDIA's blog, titled "Three huge applications accelerated by the world's most powerful graphics chip".

Rajiv Ranjan is a Research Scientist and a Julius Fellow in CSIRO Computational Informatics Division (formerly known as CSIRO ICT Centre). His expertise is in datacenter cloud computing, application provisioning, and performance optimization. He has a Ph.D. (2009) in Engineering from the University of Melbourne. He has published 62 scientific, peer-reviewed papers (7 books, 25 journals, 25 conferences, and 5 book chapters). His hindex is 20, with a lifetime citation count of 1660+ (Google Scholar). His papers have also received 140+ ISI citations. 70% of his journal papers and 60% of conference papers have been A*/A ranked ERA publication. Dr. Ranjan has been invited to serve as the Guest Editor for leading distributed systems journals including IEEE Transactions on Cloud Computing, Future Generation Computing Systems, and Software Practice and Experience. One of his papers was in 2011's top computer science journal, IEEE Communication Surveys and Tutorials.

Prof. Albert Y. Zomaya is currently the Chair Professor of High Performance Computing & Networking and Australian Research Council Professorial Fellow in the School of Information Technologies, The University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing which was established in late 2009.

Professor Zomaya held the CISCO Systems Chair Professor of Internetworking during the period 2002–2007 and also was Head of school for 2006–2007 in the same school. Prior to his current appointment he was a Full Professor in the School of Electrical, Electronic and Computer Engineering at the University of Western Australia, where he also led the Parallel Computing Research Laboratory during the period 1990–2002. He served as Associate-, Deputy-, and Acting-Head in the same department, and held numerous visiting positions and has extensive industry involvement. Professor Zomaya received his Ph.D. from the Department of Automatic Control and Systems Engineering, Sheffield University in the United Kingdom.

Dr. Wei Jie is a Senior Lecturer at Computing, School of Computing and Technology, univeristy of west London. He is a Senior Member of The Institute of Electrical and Electronics Engineers (IEEE) and a Fellow of the Higher Education Academy (FHEA). Dr. Wei Jie has been active in a broad spectrum of areas in parallel and distributed computing, in particular, grid and cloud computing, computing security technologies, e-science and e-research. Dr. Jie has been actively involved in professional services. He is the General Chair of the IEEE workshop on Security in e-Science and e-Research, and has served as Program Committee member for more than 40 international conferences and workshops.

Dr. Wei Jie has published approximately 50 papers in international journal and conferences and has edited three books.