A Parallel File System with Application-aware Data Layout Policies For Massive Remote Sensing Image Processing in Digital Earth

Lizhe Wang, *Senior member, IEEE*, Yan Ma, *Member, IEEE*, Albert Y. Zomaya, *Fellow, IEEE*, Rajiv Ranjan, *Member, IEEE,* and Dan Chen

Abstract—**Remote sensing applications in Digital Earth are overwhelmed with vast quantities of remote sensing (RS) image data. The intolerable I/O burden introduced by the massive amounts of RS data and the irregular RS data access patterns has made the traditional cluster based parallel I/O systems no longer applicable. We propose a RS data object-based parallel file system for remote sensing applications and implement it with the OrangeFS file system. It provides application-aware data layout policies, together with RS data object based data I/O interfaces, for efficiently support of various data access patterns of RS applications from server side. With the prior knowledge of the desired RS data access patterns, HPGFS could offer relevant space-filling curves to organize the sliced 3-D data bricks and distribute them over I/O from the servers. In this way, data layouts consistent with expected data access patterns could be created to explore data locality and achieve performance improvement. Moreover, the multi-band RS data with complex structured geographical metadata could be accessed and managed as a single data object. Through experiments on remote sensing applications with different access patterns, we have achieved performance improvement of about 30% for I/O and 20% for total.**

Key word: **Parallel File System; Parallel I/O; Data-intensive Computing; Digital Earth; Big Data; Remote Sensing Image Processing**

I. Introduction

Digital Earth[1], a global information model of earth's surface, plays a significant role in addressing challenges in global resources and environmental change. The digital earth approach is to exploit regional to global covered remote sensing data for processing. With recent advances in remote sensing technology, the latest-generation space-borne sensors are continuously producing massive quantities of high-dimensional remote sensing (RS) image data in Digital Earth at a rate of several terabytes per day [2,3]. Meanwhile, remote sensing data processing for digital earth is commonly recognized as typical data-intensive applications. The large-scale RS applications like continental mosaicking for Global Rain Forest Mapping [4] normally require processing thousands of imageries that adds up to nearly terabytes of data.

Incorporation of distributed cluster-based high-performance computing (HPC) with remote sensing applications in Digital Earth is an effective solution for addressing these computational challenges introduced by massive data. However, the performance gap between I/O and computing is gradually widening by further growth of computing capability, especially in cluster scenarios with thousands of cores. In this case, the data processing has to wait for several CPU cycles for data accessing [5]. Withal, most of the remote sensing applications are significantly challenged with the specific complex data access patterns that result from different correlations between computation and data. These complex data access patterns perform intensive small non-contiguous or irregular I/O over multiple data files and vary across different applications. The data I/O then turns out to be an expensive burden for remote sensing applications in digital earth. Remote sensing image data sets, in particular, are characterized by multi-dimensional image data structure and abundant geographical metadata. Consequently, offering direct and efficient storing and accessing of these special structured RS datasets in traditional file system could be rather cumbersome, although it remains paramount.

The intolerable I/O burden introduced by massive quantities of RS data and the complex data access patterns has rendered traditional I/O systems no longer applicable. Employing a parallel file system with Parallel I/O [6,7] is an effective challenges encountered in remote sensing applications. The I/O parallelism is commonly achieved by block-wise striping of large data across a great number of storage blocks with a fixed stripe size. This method allows applications to access data from multiple I/O servers or storage concurrently. However, despite the high I/O parallelism, parallel file systems continue to suffer from the relatively poor performance of non-contiguous I/O patterns. The reason for this is that most of the mainstream parallel file systems seldom directly support these data accesses patterns properly [8]. Furthermore, although the standard data striping method and round-robin data layout scheme can benefit some data access

Lizhe Wang and Yan Ma are with the Institute of Remote Sensing and Digital Earth, Chinese Academy of Science, Beijing, China 100094. Albert Y. Zomaya is with the School of Information Technologies, the University of Sydney, Australia. Dan Chen is with the School of Computer, China University of Geosciences. Rajiv Ranjan is with ICT/CSIRO, Australia. Corresponding author: Yan Ma, phone/fax: 86-10-82178970; e-mail: yanma@ceode.ac.cn;

patterns, this is not the case for others [9]. In fact, the mismatch between the data access pattern and the physical data layout over I/O servers would inevitably introduce a workload imbalance among I/O devices and result in I/O inefficiency [10]. Besides, general-purpose file systems could not directly support the special data structure of remote sensing image datasets consisting of multi-dimensional image data and various geographical metadata. Therefore, trying to get a parallel file system to offer efficient managing and high-performance non-contiguous or irregular parallel accessing of RS data remains quite a challenge for remote sensing applications.

OrangeFS [11,12] is a configurable open source research platform with high-performance parallel I/O and scalable physical data layout scheme interfaces. We address the issue described above by proposing HPGFS, a RS data object based parallel file system implemented with OrangeFS and designed for massive remote sensing image processing. The main contribution of HPGFS is that it introduces application aware data layout policies together with RS data object based data parallel I/O interfaces to provide direct and efficient accessing of RS data for various remote sensing applications. As a special-purpose file system, HPGFS uses a logical distributed RS data object to describe the remote sensing image datasets and their relevant basic data operations. The complex structured geographical metadata are stored to or indexed from distributed metadata servers in a key-value pair manner. While, the multi-dimensional image data are scattered across I/O servers using application aware data layout policies. With the prior knowledge of the different I/O characteristic of applications, HPGFS could provide suitable space-filling curves for organizing the sliced multi-dimensional data bricks and distributing them among I/O servers. This method would create a physical storage layout consistent with expected data access patterns in order to explore data locality and achieve performance improvement. Moreover, HPGFS proposed an RS data object based I/O interfaces to describe the direct non-contiguous or irregular access of RS data for remote sensing applications. This native application-oriented noncontiguous data access support in the file system is recognized as being critically important to I/O efficiency [13].

The rest of this paper is organized as follows. The next section overviews some related work, and section III discusses the data access patterns of remote sensing applications and defines the I/O problems encountered in massive remote sensing processing with a parallel file system. In section IV, we address the design and implementations of the HPGFS with application-aware data layout strategies. Section V discusses the experimental analysis of HPGFS's I/O performance for remote sensing applications with different data access patterns, and section VII concludes this paper.

II. Related Work

Parallel file systems with scalable parallel I/O are widely employed in modern clusters where I/O emerges as the main bottleneck [14], especially for data-intensive applications [15]. Significant efforts have focused on cluster-based file systems, such as OrangeFS [12], PVFS [16], Lustre [17], PanFS [18] and GPFS[19]. These file systems converge the direct-attached storage of each node with networking techniques [20] and strip the data across this storage, in order to offer high throughput concurrent I/O. However, only one tenth of the peak I/O performance is achieved by most scientific applications [14,21], that perform small non-contiguous I/O [8]. This is because most of the mainstream file systems are optimized for large contiguous data accessing. Overall, their parallel I/O interfaces and physical data layout over storages do not match the expected data access patterns of the applications [22,23].

A fair amount of research works on parallel I/O libraries has been devoted to the optimization of non-contiguous I/O. ROMIO [13] is a well-known implementation of portable MPI-IO [21] interfaces that has extended many optimization techniques to handle non-contiguous data accesses. Data sieving [24,25] is one of the most widely accepted optimization techniques for many applications. It reduces I/O calls by reading a large contiguous chunk of data and sieving out non-requested ones, but with the penalty of reading and extracting extra data. Two-phase I/O [21][26] introduces Gather-Scatter communication in an extra exchange phase to merge non-contiguous I/O requests for exploring data locality, but this approach inevitably brings in extra communication overhead. The combination of data sieving with two-phase I/O by collective I/O [24] allows collection of small non-contiguous requests into a large contiguous one. This approach could greatly reduce I/O requests and achieve a better ratio of actually accessed data. Therefore, these techniques achieve I/O performance improvement by merging or reducing small non-contiguous data requests.

The data layout policy of a parallel file system that organizes the physical data layout over I/O servers is a key factor in determining parallel I/O performance. Proper data layout policies would lead to data locality and workload balance among I/O servers. Instead of the standard simple data striping approach, OrangeFS offers interfaces for customizing data layout policies. The data replication [28,29] schemes reorganize data layout by creating data replicas across I/O servers. These concepts focus on amortizing I/O requests and exploring data locality, but with the penalty of sacrificing storage capacity. In addition, Song et al. [1] have proposed a segment-level adaptive data layout scheme for variable I/O patterns. This method uses different stripe sizes in different file segments for an overall I/O performance optimization.

A large body of research [30] has focused on the algorithm optimization to address the computational challenges of remote sensing applications. By contrast, the parallel I/O efficiency of remote sensing applications and incorporation of parallel file systems have received little attention. Some limited work has attempted to building cluster-based I/O systems for managing massive RS data [31][30]. Other works has opted for the data prefetching techniques [33] employed by many remote-sensing applications [30][34] with intensive non-contiguous I/O. Explicit overlapping of the I/O operation with computing can hide the I/O latency to a certain extent. Nevertheless, data prefetching cannot completely eliminate inefficient non-contiguous or irregular I/O requests. In addition, some data layout optimization work from server side in parallel file system accompanies it [35][36]. In any case, the

RS applications still remain extremely troubled by intensive and cumbersome RS data accessing.

The HPGFS put forward in this paper aims at coping with these I/O burden issues. It offers RS data object based distributed storing and parallel I/O interfaces for native and efficient support of RS data accessing. It relies on the application aware data layout policies and thereby provides optimal organization and storage layout of multi-dimensional RS data that are consistent with the specific I/O pattern of RS applications. Overall, HPGFS develops its implementation on the prototype of OrangeFS by leveraging its high configurability.

III. Problem Definition

This section demonstrates the primary issues related to the parallel I/O system for massive remote sensing image processing in a cluster scenario. The problem has three key aspects: massive remote sensing data with high dimensionality, complex data access patterns of remote sensing applications, and the parallel I/O performance. The first issue involves the proper support of the distributed storage of large amounts of complex structured RS data and offering an easy to use RS data object based data operations (Section III-A). The second issue concerns the intensive but miserable RS data access patterns of applications, which perform non-contiguous I/O across many data files (Section III-B). The third issue concerns the creation of an optimal storage data layout that will allow extensive exploration of data locality (Section III-C).

A. Massive RS Data with High Dimensionality

The remote sensing image processing applications are overwhelmed with incredible amounts of RS data. Generally, the data amount of a single RS dataset would be several gigabytes. Large-scale remote sensing applications like those for global climate change may require processing thousands of multi-temporal RS data that add up to several terabytes. Accordingly, the intensive I/O result from massive RS data and increasing real-time processing requirements have become the main factors that restrict the data processing speed of applications.

Normally, the RS imagery from hyper-spectral sensors would have hundreds of spectral bands. However, the high dimensionality of RS data complicates the storage and access of distributed data in parallel I/O systems. The difficulties lie in mapping multi-dimensional imageries to 1-D data array. For specific, it may include appropriate data partitioning and data organization by the choice of suitable space-filling curves. On the other hand, RS datasets usually include associated complex- structured geographical metadata, which makes support of efficient storage and indexing of geographical metadata rather cumbersome on parallel I/O systems. Moreover, geographical metadata are always involved in computation. In case of the RS data accessing, a recalculation of the geographical metadata (like latitude, longitude) would be necessary for requests.

B. Irregular Data Accessing Patterns of RS Applications

In most cases, remote sensing applications show different computation features: different degrees of dependency between computation of algorithm and RS data. This is because the computation of each pixel usually depends on its neighborhood or even the data from other spectral bands. These dependencies involve data independent computation and regional dependent computation, as well as band dependent computation.

Generally, the data dependencies, to various degrees, would result in different data access patterns that vary across applications. Except for the data independent computation featured applications, most of the RS applications usually perform non-contiguous or irregular I/O. The applications that feature regional dependent computation tend to request many small data blocks scattered throughout the file in one logical I/O. Overall, the applications with band dependent computation normally access small data regions from multiple band files simultaneously. Unfortunately, these data access patterns are seldom natively supported by most of the popular parallel file systems. Traditionally, the non-contiguous I/O is implemented by repeatedly calling a number of individual data requests, each of which accesses a small chunk 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.

MPI-IO library has provided derived MPI datatypes to reflect the desired non-contiguous I/O pattern in a uniform style. The <*length,stride*> tuples are used to describe the arrangement of the non-contiguous data. However, these requested non-contiguous RS data usually cover a rectangular or irregular shaped region of the image. Even worse, the requested RS data may be scattered over multiple band files. As a result, relying on MPI-IO to describe the specific I/O pattern of RS applications becomes quite clumsy and difficult. Therefore, a more intuitive, elegant and effective solution that is natively supported by PFS is clearly necessary.

C. Parallel I/O Performance

The existing I/O optimization techniques like data sieving and collective I/O do not overcome the poor parallel I/O performance of data-intensive RS applications that incorporate PFS. The main reason is the mismatch between the storage data layout and the I/O patterns of RS applications.

Historically, the parallel file systems and applications are designed separately for transparency. Hence, parallel file systems commonly adopt a simple data striping method, but have no idea of the data access patterns of the applications. In case where the data layout created with a certain stripe size is not consistent with the desired data accessing patterns, poor data locality and imbalance of I/O workloads arise, ultimately leading to an I/O bottleneck. Consequently, proper data layout policies especially designed for the data access patterns of remote sensing applications are critically important for elegant data locality and parallel I/O performance.

IV. RS Data and RS Data Access Patterns

In this section, we introduce the data structure of RS data and probe into the I/O characteristics of RS data access patterns performed by various remote sensing applications.

A. Data Structure of Multi-dimensional RS Data

RS images are generally acquired from different sensors, like such as optical, hyperspectral, and SAR images. Unlike normal imgages, RS data consist of geographical metadata that vary with different sensors and multi-band images whose pixels are of high dimensionality.

1) Image Data with High Dimensionality (T)

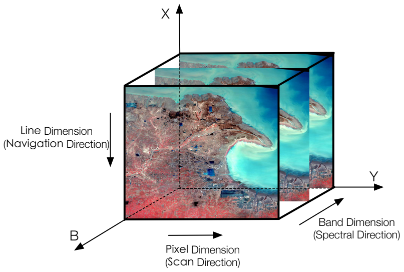


Fig. 1. 3-D Image of A RS Data

The image data T of the RS data records the spectral information and spatial location of the land surface features. For simplicity, we ignore the temporal dimension of RS data here. Then, a RS data object corresponds to a regular 3-D remote sensing image dataset as is showed in figure 1. Accordingly, each pixel of the RS image may be expressed as $(x, y, r(b_1, b_2, \dots, b_k))$, where (x,y) express the spatial location of each pixel and $r(b_1, b_2, \dots, b_k)$ is the DN value array of spectral reflectance across different spectral bands.

Mathematically, the RS image T with k bands of images in size of m lines and n pixels could be organized as a 3-D matrix (Equation 1). For different computation requests, the image T could be arranged as a band major order (BSQ), a band interleaved by pixel (BIP), or a band interleaved by line order (BIL).

$$T = \begin{bmatrix} r_{111} & \cdots & r_{11n} \\ \vdots & \ddots & \vdots \\ r_{1m1} & \cdots & r_{1mn} \end{bmatrix} \begin{bmatrix} r_{211} & \cdots & r_{21n} \\ \vdots & \ddots & \vdots \\ r_{2m1} & \cdots & r_{2mn} \end{bmatrix} \cdots \begin{bmatrix} r_{k11} & \cdots & r_{k1n} \\ \vdots & \ddots & \vdots \\ r_{km1} & \cdots & r_{kmn} \end{bmatrix} \quad (1)$$

2) Geographical Metadata (P)

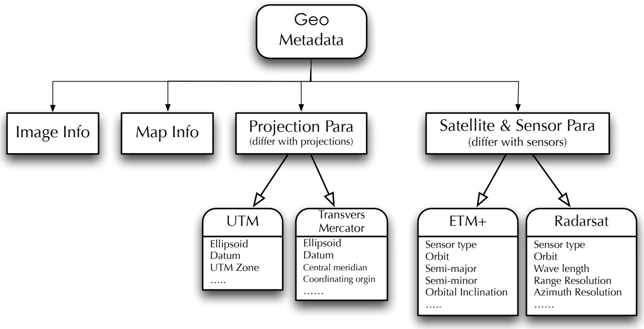


Fig. 2. The Data Structure of Geographical Metadata

The geographical data P are self-descriptive metadata and always participate in the computation of RS algorithms. As depicted in Figure 2, these includes image info, map info, satellite & sensor parameters and projection parameters. The image info describes the size, data type and the data organization of RS data. The map info records the spatial location of data, like latitude/longitude and x/y values of four corners respectively in geographical and projected coordinate system. However, the data items as well as the data structure of projection parameters vary with different projection methods. The same situation holds for satellite and sensor parameters. Therefore, expression of these metadata in a standard yet normalized fashion is critical for the storage of RS data in I/O systems.

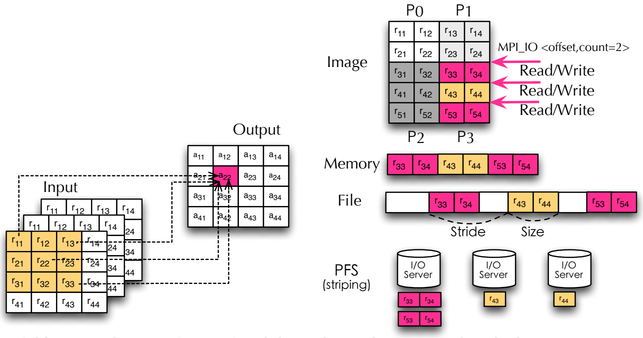
B. I/O Characteristics of RS Data Access Patterns

The different degree of dependency between computation and RS data means that remote sensing applications perform different RS data access patterns.

1) Consecutive-Lines Access Pattern for Pixel-Based Processing

Pixel-based processing refers to the data independent algorithms whereby each pixel can be computed without requesting for context. This category of algorithms, likes radiometric correction and SVM classifier, perform *consecutive-lines access pattern* with perfectly contiguous I/O. When these algorithms are implemented in parallel, each processor requests multiple consecutive image lines in a logical data access. These data correspond to a continuous stream of bytes in a file view, and as many contiguous data stripes distributed over storage in PFS. With MPI-IO, data could be requested in a <offset,count> tuple.

2) Rectangular-Block Access Pattern for Neighbor-Based Processing



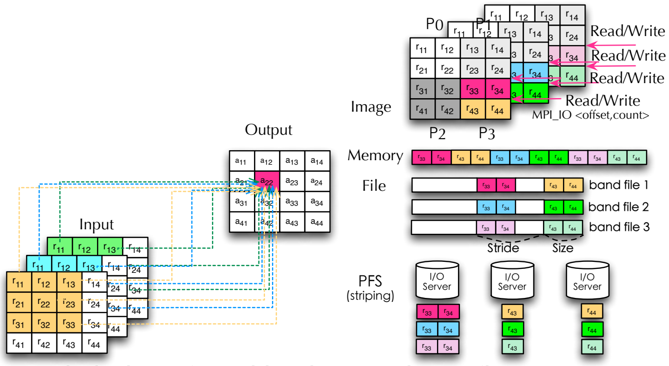
(a) Neighbor-Based Processing(Regional dependent) (b) Rectangular-Block Access Pattern

Fig. 3. Rectangular-Block Access Pattern for Neighbor-Based Processing

Neighbor-Based Processing refers to regionally dependent algorithms, where the calculation of each pixel depends on its corresponding close neighbors (rectangular window in figure 3(a)). This category of algorithms such as the convolution filter and image resampling perform a *rectangular-block access pattern*. This is a non-contiguous I/O pattern commonly seen in RS algorithms, but one that is not well supported by normal PFS. As drawn in figure 3(b), each processor requests one rectangular shaped data block at one time. In a file view, these data correspond to many small non-contiguous data fragments scattered throughout the file with same stride and size. These data are accessed by repeated read/write operations, each of which requests a small fragment of contiguous data. With MPI derived datatypes, this I/O pattern could be reflected uniformly but it has miserable performance.

The extreme case would be the *columns access pattern*, where each processor requests one column and a single I/O operation only accesses one pixel. As a result, the bytes used to describe data fragments in <offset,count> tuples would even exceed the actual requested image data.

3) Cross-File Access Pattern for Band-Related Processing

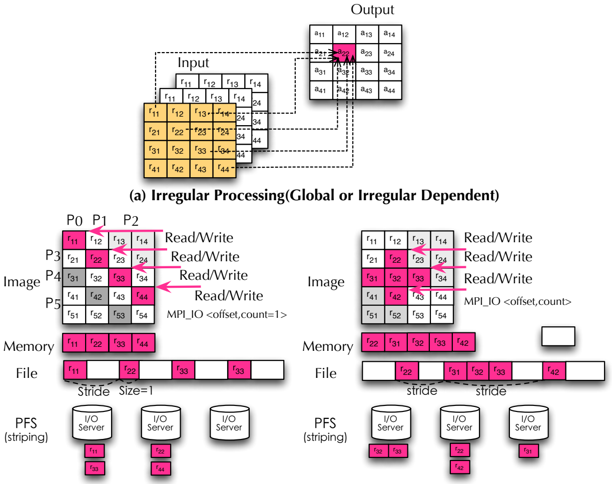


(a) Band-Related Processing(Band dependent) (b) Cross-File Access Pattern

Fig. 4. Cross-File Access Pattern for Band-Related Processing

Band-Related Processing refers to cases where the computation of a single pixel requires its context across several image band files. These algorithms, such as fusion and NDVI, perform a *cross-file data access pattern* (figure 5(a)). Each processer rquests small consecutive or rectangular shaped data from multiple band images simultaneously in a single logical data accessing. In the file view, these data are small non-contiguous fragments scattered across hundreds of image files in the same manner. This is generally translated into large numbers of read/write operations, each of which only request a small fragment of contiguous data located in a single image file. As a result, implementing this kind of data access pattern ends up being very cumbersome and time-consuming.

4) Irregular Access Pattern for Irregular Processing



(a) Irregular Processing(Global or Irregular Dependent)

(b) Diagnal Access Pattern (c) Polygon Access Pattern

Fig. 5. Irregular Access Pattern for Irregular Processing

Irregular Processing features algorithms whose single computation depends on the irregularly shaped data or even the entire image. Examples of this category are FFT, image warping and information extraction. These algorithms perform *irregular access patterns* including the *diagonal and polygon access patterns*. Each processor requests an irregularly shaped data region. For diagonal access patterns, the accessed data are located along the diagonal direction in the image view, while polygon access patterns request irregular polygon-shaped data regions in the image view. While, the polygon access pattern request a polygon shaped irregular data region in image. In a file view, these data are small non-contiguous data fragments with different size and they step over different strides. In some circumstances, parameters like offset, stride, and size cannot be determined in advance. The difficulties lie not only in the poor I/O performance but also the description of the irregular data regions.

V. Design And Implementation Of HPGFS

We propose HPGFS, a RS data object based file system for remote sensing applications. It offers a more efficient and easy-to-use solution from the server side that natively supports the direct distributed storage and concurrent accessing of massive RS data objects in different irregular I/O patterns. Based on the analysis of RS data along with the I/O characteristics of RS applications (Section IV), we design and implement HPGFS with an OrangeFS prototype. In HPGFS, a logical distributed RS data object model is adopted to organize the RS image datasets with complex data structure (Section V-A). Meanwhile, it also provides I/O interfaces with RS data operation semantics (Section V-C), together with application aware data layout policies (Section V-B) associated with expected data access patterns of applications.

The OrangeFS file system is an advanced branch of PVFS. It provides interfaces for data layout policy customization and distributed, distributed metadata management, as well as the native list-I/O interfaces for non-contiguous I/O description. Therefore, we adopt this research oriented open source platform as a desired parallel file system prototype for HPGFS implementation.

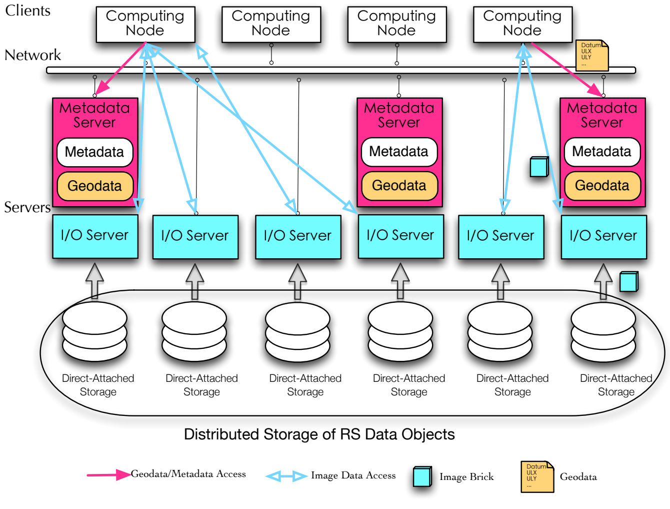


Fig. 6. Architecture of HPGFS Implementation

As is depicted in Fig.6, the HPGFS is implemented in a cluster scenario where each node is equipped with a direct-attached storage. This storage could be a single disk or a local disk array with different RAID levels. By virtue of OrangeFS, nodes in a cluster, with direct-attached storage, act as I/O servers and are converged as a single file system mirror for remote sensing image datasets. In OrangeFS, the nodes may serve three different roles including I/O server, metadata server, and computing node (client). Actually, a single node could perform all tree roles at the same time. Instead of introducing a new node role "Geodata Server," we extend the metadata server of OrangeFS to support the management of the geographical metadata of RS data object. This is because the introduction of a centralized "Geodata Server" would inevitably ruin scalability and give rise to single point of failure (SPOF).

The issues mentions in Section 3.1 are resolved here by modeling the RS image data set (consisting of multi-dimensional images and geographical metadata together with data operations) as a logically distributed RS data object, where the light-weight Berkely DB in distributed metadata servers is adopted for storing and managing the complex structured geographical metadata in a key-value fashion. This method allows simple retrieval of the data items of the geographical metadata.

We address the problem discussed in Section 3.2 by designing and implementing a set of RS data object oriented I/O interfaces. For a more intuitive and simple description of the RS data access patterns semantics (section IV-B), we provide some RS data structures to reflect the requested data region instead of uniformed <offset,length> tuples. These data structures include RSBlock, RSRect, RSDiagonal. Accordingly, the required data consist of many small non-contiguous data slices or even reside across band files could be easily requested in a one single logical I/O.

We tackle the I/O performance issue in Section 3.3 by proposing application aware data layout policies to support the specific data access patterns of RS applications from server side. Except for the data striping method with fixed or variable stripe size, we adopt a RS brick based data slicing policy, where the sliced data bricks are also multi-dimensional data scattered non-contiguously in file. With the awareness of the expected data access pattern of RS applications, multiple choice of space-filling curves like Z-order curve and Hibert curve are offered for access-pattern aware image data organization. Moreover, except for the standard round-robin policy, we have implemented different data distribution policies by extending policy-customizing interfaces, in order to create data layouts that are consistent with the desired I/O patterns.

A. *Organization and Management of RS Data Object*

1) Logical Model of Distributed RS Data Object

In the HPGFS parallel I/O system, we adopt a logical distributed RS data object model to describe the multi-band RS image and geographical metadata, as well as relevant basic RS data operations.

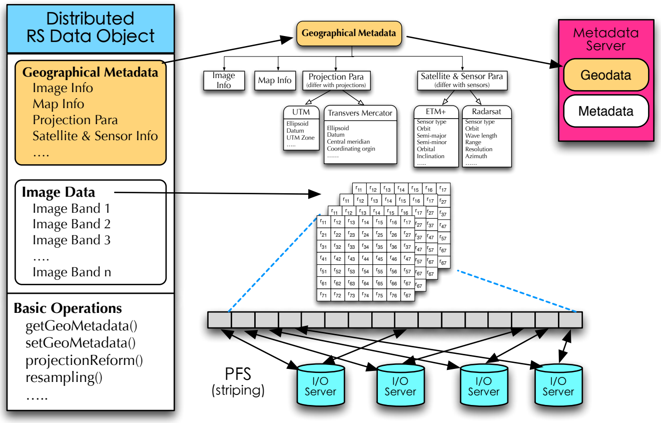


Fig. 7. Logical Model of Distributed RS Data Object

As is depicted in figure 7, the data items of geographical metadata are normalized in a uniform fashion and stored in the distributed metadata server using a <key,value> pairs, while the 3-D multi-band images are sliced and mapped into a 1-D data array using space-filling curves. The arranged data array is then scattered over a number of I/O servers with application aware data layout policies. A proper data layout consistent with expected RS data access pattern is then created for exploiting data locality and high throughput parallel I/O. Overall, the basic RS data operation includes some metadata inquiry interfaces and geographical operations like projection reform and resampling.

2) Management of Geographical Metadata

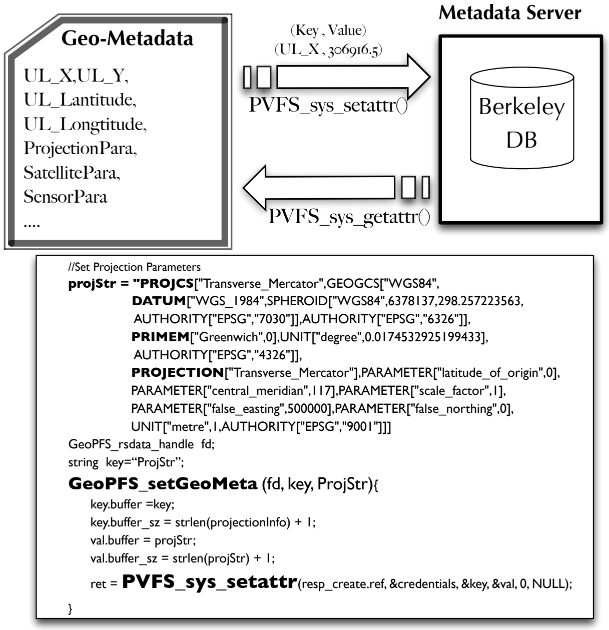
As showed in figure 2, the geographical data of a RS data object involves some metadata with variable data items and data structure. However, the metadata server has difficulty in supporting tree structured metadata in a simple <key,value> manner. Accordingly, we adopt a normalized XML string to express the projection parameters and satellite & sensor parameters with variable data items and tree style hierarchical structures. For a standard expression, we adopt a WKT (Well-known Text) string (projStr in figure 8) from the OGC (Open Geospatial Consortium) to normalize the projection parameters. The organization of the geographical metadata is shown in table I.

TABLE I

DATA STRUCTURE OF GEOGRAPHICAL METADATA)

Data Items	KeyWord	Data Type
ImageInfo	ImageLine	integer
	ImageCol	integer
	ImageType	integer
	ImageBands	integer
	ImageDataOrder	string
	Compression	string
SatSensorPara	SatSensorStr	String (xml)
ProjectionPara	ProjStr	String (WKT)
MapInfo	ULLatitude	string
	ULLongitude	string
	LRLatitude	string
	LRLongitude	string
	CenterPointLat	string
	CenterPointLon	string
	XResolution	decimal
	YResolution	decimal
	ULXCoordinate	decimal
	URXCoordinate	decimal
	LLXCoordinate	decimal
	LRXCoordinate	decimal
	Units	string
StatInfo	HistGram	string
	LastModified	date

OrangeFS provides metadata accessing interfaces including PVFS_sys_setattr() and PVFS_sys_getattr(). As portrayed in figure 8, by virtue of OrangeFS meatadata operation interfaces, we normalize the geographical metadata into <key,value> pairs, and store them into the Berkeley DB. In this way, these metadata can be easily accessed and inquired.



```
//Set Projection Parameters
projStr = "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"]]]
GeoPFS_rsdata_handle  fd;
string  key="ProjStr";
GeoPFS_setGeoMeta (fd, key, ProjStr){
    key.buffer =key;
    key.buffer_sz = strlen(projectionInfo) + 1;
    val.buffer = projStr;
    val.buffer_sz = strlen(projStr) + 1;
    ret = PVFS_sys_setattr(resp_create.ref, &credentials, &key, &val, 0, NULL);
}
```

Fig. 8. Storing Geographical Metadata into Metadata Server

B. *Application Aware Data Layout Polices for RS Data*

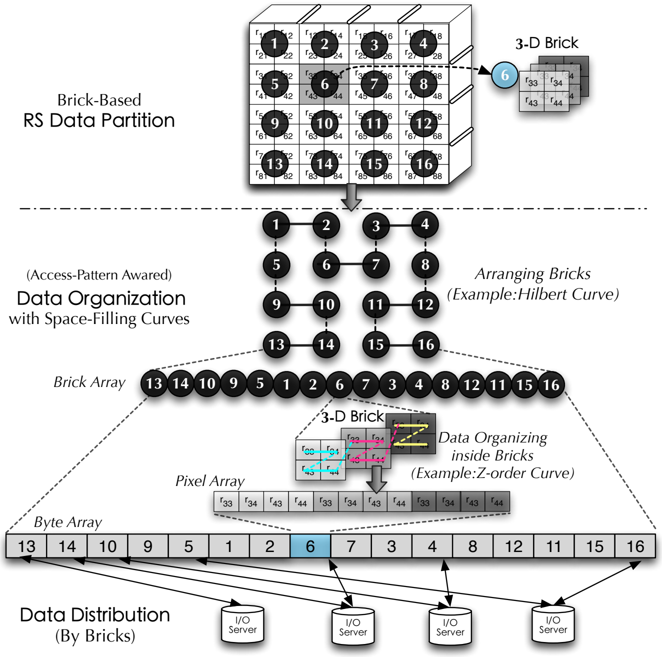


Fig. 9. Building Data Layout with Application Aware Policies

We provided efficient support of the specific data patterns performed by a variety of remote sensing applications by proposing a set of application. These polices could create a desirable RS data layout over I/O servers for some expected data access patterns. The construction of the RS data layout with application aware policies shown in figure 9 could also be described as follows:

--Firstly, divide the entire 3-D multi-band RS images into small data fragment with brick-based 2-D data slicing method.

--Secondly, arrange the sliced data bricks in a 2-D plane into an ordered 1-D data brick array, and also map the 3-D brick to a 1-D data array by space-filling curves. These space-filling curves are appropriately chosen according to the requested RS data access patterns.

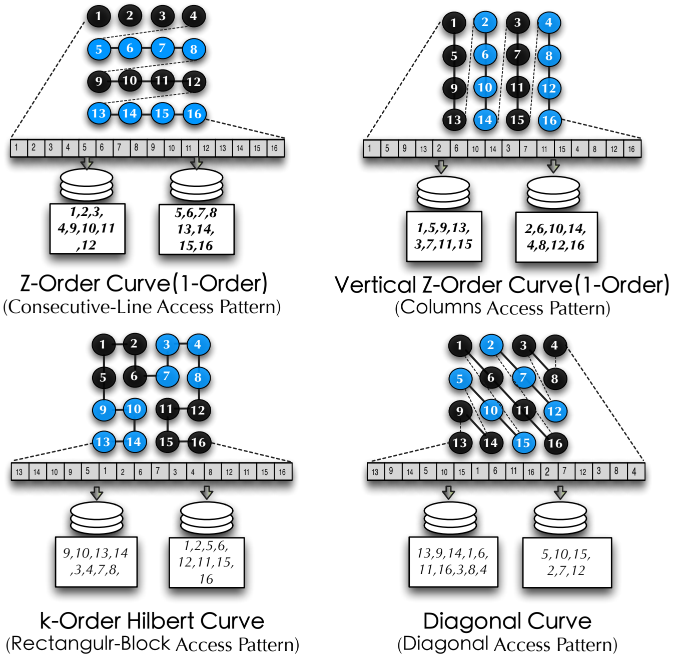
--Finally, scatter the stream of arranged image data with lower dimensionalities across multiple I/O servers with different data distribution methods.

In this way, HPGFS could natively create an efficient RS data layout over I/O servers that is suitable for many desired I/O patterns.

1) *Brick-Based Data Slicing Method*

Taking advantage of the regular geometric characteristic of RS image data, the RS applications normally divide the image with different data partition methods that exploit parallelism, since data partitions like line, row or diagonal partition could be created by rearranging the brick-based 2-D partition, where a data brick is a 3-D cubic data fragment with multiple bands that would normally be accessed together by remote sensing algorithms in one logical I/O. We considered the access patterns of RS applications by adopting a brick-based RS image partition (figure 9) as a basic data partition method in this paper. The brick-based 2-D partition is essentially a row-col division where all the multi-band images are first divided in a row direction, followed by division in column direction. In contrast to the data stripes in normal PFS, the most distinguishing thing is that the data bricks desired by RS applications are non-contiguous data scattered throughout the file with fixed stride and size.

2) Access-Pattern Aware Data Layout with Space-Filling Curves



Z-Order Curve(1-Order)
(Consecutive-Line Access Pattern)

Vertical Z-Order Curve(1-Order)
(Columns Access Pattern)

k-Order Hilbert Curve
(Rectangulr-Block Access Pattern)

Diagonal Curve
(Diagonal Access Pattern)

Fig. 10. Different Space-Filling Curves for Different Access Patterns

We provide proper support for the different RS data access patterns of the applications by employing different choices of space-filling curves for optimal data organization, like the Hilbert curve [37][38] and Z-order curve [39]. Firstly, in the *brick arranging* stage, many sliced data bricks are organized into an ordered array of data bricks with Hibert space-filling curve as well as modified Z-order curves. The next step is *data arranging inside bricks,* where the pixels inside each of the 3-D data bricks are mapped into a 1-D array of pixels with Z-order curve. Finally, all the data located in the entire multi-band image are arranged into a byte array ready for distribution over storage.

For *brick arranging*, different 2-dimension space-filling curves shown in figure 10 are adopted for creating the brick layout with the prior knowledge of the data access patterns.

The Z-Order is provided to organize the data bricks for the *consecutive-lines access pattern.* The data bricks located along the line direction would be arranged contiguously. These data, which are normally non-contiguous in a file view, would normally be accessed together in one logical I/O by some pixel-based processing featured applications.

Vertical Z-order Curve is adopted for the *columns access pattern* of RS application. This kind of space-filling curve is a modified 1-order curve, which could be constructed by rotating the normal Z-order curve counterclockwise by 90 degrees. With this curve, a brick layout that bricks located along same column direction would be ordered nearby. As a result, this brick layout will greatly facilitate the column access pattern, since the request non-contiguous bricks in one column are put in a contiguous layout that could be accessed in a single I/O.

The *Diagonal Curve* is provided for the *diagonal irregular data access pattern*. The diagonal curve is a non-recursive 2-dimensional curve. The data bricks located in a diagonal in the image space would be arranged as neighbors. Then, the non-contiguous data bricks in the diagonal could be accessed by applications with a single I/O for the contiguous brick layout created by curve.

The *Hilbert Curve (k-Order)* is used for the *rectangular-block access pattern*. Given the RS images that slices into *m*n* bricks, a k-order Hilbert curve should be used for brick mapping, where k equals to the mine ($\log_2^n$,$\log_2^m$). Benefiting from the excellent data clustering of the Hilbert curve, the neighbor brick located in both the line and column directions in the image space would be ordered contiguously in the arranged brick array. This brick layout is preferred by the applications performing rectangular-block access pattern that always request close neighbor in a single I/O.

When the sliced data bricks of multi-band images are mapped as a 1-D brick array, these data bricks would be scattered across I/O servers. For a consecutive-line access pattern, rearranged data bricks with Z-order curve would be then striped over I/O servers in a round-robin manner with a stripe size the length of a single image line. Then, the data bricks in the same image line would reside together for a better data locality. For a columns access pattern, the rearranged data bricks with Vertical Z-order curve would be then striped in a round-robin manner with a stripe size of one image column. Thus, the data bricks would be located together in one column so that computing nodes could simulta-neously and efficiently request their desired column of bricks in one single I/O. Similarly, the data bricks rearranged with the Hilbert curve should be distributed with a stripe size of bricks in order for the neighbor bricks to reside together. In a diagonal access pattern, the data bricks rearranged with the Hilbert curve should be distributed with a variable strip size.

Overall, the 3-D data bricks scattered to I/O servers should be mapped into a 1-D byte/pixel array in order to reside into the "dfile" data file in I/O servers. As depicted in figure 9, the pixels in each image plane of the 3-D brick are mapped into a pixel array with a 2-D Z-order curve. Then the pixel arrays of different image bands are arranged with band order. In this way, the same regions of different image bands are organized together as contiguous data. This kind of data layout would greatly benefit the *band-related data access pattern,* since the request data window across multiple band files could be laid out contiguously and be accessed in one single I/O operation.

3) Implementation of Data Layout Policies

OrangeFS as a highly configurable parallel file system that provides a set of PINT_dist_method interface for data layout customization. The main functional interfaces of the PINT_dist_method include logical_to_physical_offset, physical_to_logical_offset, and next_mapped_offset. With logical_to_physical_offset interface, we could define the spatial position mapping relationship from a multi-band image space to a physical datafile space in a certain I/O server, while logical_to_physical_offset could be used for define an inverse mapping of physical_to_logical_offset. In HPGFS, we defined four data layout methods including the line_brick_method for consecutive-lines access patterns, and the col_brick_method for columns access pattern, rect_brick_method for rectangular-block and band-related access patterns, as well as the diagonal_brick_method for diagonal irregular access patterns.

C. I/O Interfaces with The Semantics of RS Access Patterns

We facilitated the specific RS data access patterns of remote sensing applications by proposing a set of I/O interfaces with the semantics of these I/O patterns.

Enabled by generic RS data structures, the data region requested by different RS data access patterns that perform non-contiguous, cross-file or even irregular I/O pattern could be well reflected. These generic data structures offer a more intuitive and simpler method than that provided by a simple uniformed <offset,length> tuple.

VI. Experiments and Discussion

HPGFS implemented with OrangeFS have successfully applied to the Parallel Image Processing System for remote sensing (PIPS) which works on top on MPI-enabled clusters. Lots of remote sensing applications in PIPS are benefiting from the RS I/O interfaces and application aware storage data layouts provided by HPGFS. For performance analysis, we would conduct two groups of experiments. Where one is the performance comparative experiment of different data access pattern conducted respectively on HPGFS and OrangeFS. The other one is the performance experiment on several remote sensing applications with different RS data access patterns on HPGFS. These algorithms were implemented on a big bj-1 image data with size of 483430×19058.

The performance comparative experiments were conducted on a cluster environment that equipment with 12 nodes. These computing nodes are connected by RDMA protocol supported Infiniband network with a bandwidth of 20 gigabyte. Each node is a blade server with dual Intel(R) Quad core CPU (3.0 GHz) and 8GB memory. The operating system was Cent OS5.0, the C++ compiler was the Intel C/C++ Compiler with O3 optimizing level, and the MPI implementation was Intel MPI.

A. Comparative Experiment on OrangeFS and HPGFS

The performance comparison experiment was carried out on OrangeFS and HPGFS. We tested different data access patterns of RS applications by adopting a popular, we adopt a popular I/O benchmark tool IOR [38], which is capable of using MPI_IO interfaces. The downside of IOR is that it is quite limited in its capabilities, only utilizing the sequential and strided I/O manners. Thus, we improved the IOR source code by integrating RS I/O interfaces of HPGFS, so as to meet the irregular I/O pattern requirements of RS applications.

The performance of several different parallel data accessing patterns, including rectangular-block (RECT), columns (COL), and the normal random (RANDOM) access patterns were tested on both Orange FS and HPGFS. For experiment on Orange FS, IOR utilized the MPI_IO to implement these I/O patterns, while for HPGFS, IOR utilized the RS I/O interfaces provided natively by HPGFS. The comparative performance curves are also illustrated in Figure 11 and 12, where the total requested data amount adds up to about 512 gigabytes.

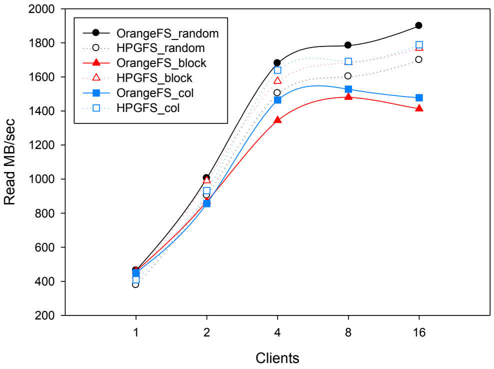


Fig. 11 Parallel Read Performance of OrangeFS and HPGFS (I/O patterns: random, rectangular-bock, col)

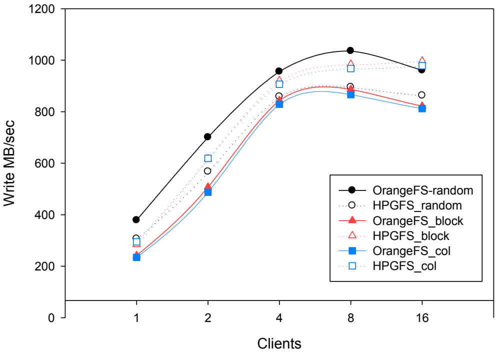


Fig. 12 Parallel Write Performance of OrangeFS and HPGFS (I/O patterns: random, rectangular-bock, col)

From the experimental results demonstrated in figure 11 and 12, we could tell that Orange FS had an excellent performance with random data access patterns, as its read performance scaled well with increasing computing nodes (clients). In contrast, when implementing rectangular-block (RECT) and columns (COL) access pattern, the performance of OrangeFS is relatively poor. With the number s of clients increasing to more than 4, the performance even goes down. In a word, OrangeFS shows its poor scalability for the RS data accessing patterns which performance non-contiguous I/O. The reason for that would probably be the mismatch between the required data access patterns and the native I/O interfaces together with the data layout of OrangeFS. This kind of mismatch increases actual I/O operations and also brings in more I/O penalty for exchanging data among I/O servers.

Compared to OrangeFS, HPGFS have got an apparent performance improvement for RECT and COL data access patterns. The parallel read performance has improved by 6% to 18%, while the parallel write performance has improved by 10% to 20%. With increasing scale of clients, HPGFS shows a better performance scalability. The reason for it goes with the RS I/O interfaces properly reflect the I/O pattern natively in file system as well as the appropriate data layout created by application aware data layout polices of HPGFS. The data layout consistent with desired I/O patterns gives rise to a better data locality of the file system. Benefiting from the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2322362, IEEE Transactions on Parallel and Distributed Systems

optimized data layout, the RS data requested by each computing node of these applications can be then accessed locally through one single direct I/O locally. However, the performance of the random data access pattern is generally poor. We could conclude that OrangeFS with striping method would more desirable for a random data access pattern.

B. Comparative Experiment on RS Algorithms

Three remote sensing image processing algorithms with different data access patterns are chosen for a comparative performance experiment on both HPGFS and Orange FS. These chosen algorithms are MTF, Band Registration (BR), and NDVI. This experiment analyzes both the total performance and the I/O and computation time penalties during the processing..

1) MTF Correction

The MTF algorithm is also an example of a rectangular block data access pattern. Each computation of MTF will require a data window with a fixed round size. This means that for each computation, the algorithm has to request a whole rectangular data block, these are scattered throughout the whole image file. The performance of MTF accelerated with HPGFS (MTF_HPGFS) and Orange FS(MTF) are demonstrated in Figure 13.

The total runtime and I/O curves illustrated in Fig. 13 tell us that the total runtimes of both MTF and MTF_HPGFS decrease linearly with increasing computing nodes (cores). An increase in node scale from 1processor(4 cores) to 7 processors (28 cores), gives an I/O time occupation of the total runtime of MTF on OrangeFS would be from 17 and 50 percent. By contrast, the I/O time occupation of MTF_HPGFS would be from 13% to 43%. By comparison with the OrangeFS accelerated MTF algorithm, an improvement of about 28 percent in I/O performance from HPGFS accelerated MTF_HPGFS would lead to a reduction in the total runtime of 19 percent (28 cores).

The experimental results show that the rectangular-block shaped RS data with multiple bands could be requested directly and locally in one single I/O, while in the normal parallel I/O system, this might require many I/O operations that each access a small fragment of continuous data. Therefore, I/O overhead improvements would be expected from data layout optimization.

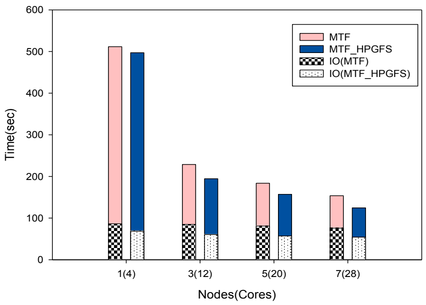


Fig. 13 The Runtime and I/O Time of MTF

2) Band Registration

The band registration algorithm always performs a rectangular-block data access pattern across two image files. Therefore, to some extent, band registration is also a simple example of a band-related data access pattern. Each computation of this algorithm will conduct a search operation for finding a matched pixel pair in a rectangular window of both two image bands. This means that for each computation, the algorithm must simultaneously request two rectangular data blocks, which are scattered throughout two image files. The performance of band registration accelerated with HPGFS (BR_HPGFS) and Orange FS(BR) is demonstrated Figure 14.

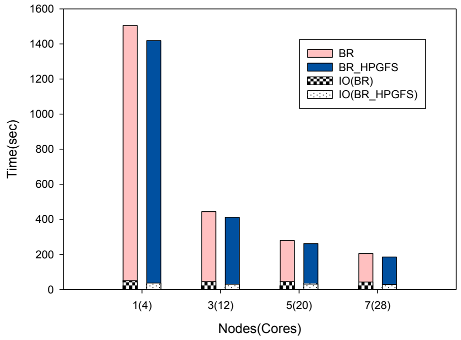


Fig. 14 The Runtime and I/O Time of Band Registration

The total runtime and I/O penalty curves of band registration algorithm illustrated in Fig. 14 reveal that the total runtimes of both MTF and MTF_HPGFS decrease linearly with increasing computing nodes (cores). These two algorithms both show excellent scalabilities. Since the BR algorithm also has relatively high degree of algorithm complexity, the occupation of I/O time in the total runtime is low. When the processor employed for implementation increases, the I/O time occupation of the BR algorithm on OrangeFS would be from 3%(1processor(4 cores)) to 20% (7 processor (28 cores)). By contrast, the I/O time occupation of BR_HPGFS algorithm on HPGFS would be from 2.5% to 15%. In other words, when compared with OrangeFS accelerated BR algorithm, the improvement in I/O performance of HPGFS accelerated BR_HPGFS would give rise to a reduction in the total runtime by 9.7% (28 cores).

3) NDVI

The NDVI algorithm conducts a serial of arithmetic operations among multiple image bands inside a remote sensing image. Normally, we would have at least 3 image bands involved in the computation. During each computation, the same data window from multiple image band files should be requested simultaneously, so NDVI is a typical example of a cross-file data access pattern. The performance of MTF accelerated with HPGFS (NDVI_HPGFS) and OrangeFS (NDVI) is demonstrated in Figure 15.

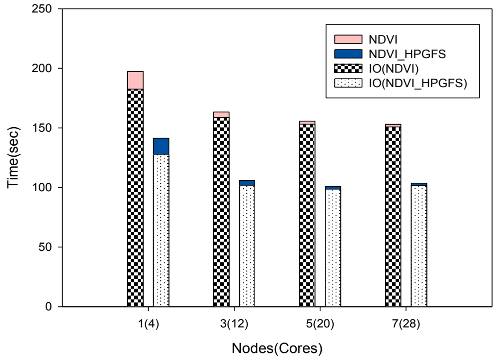


Fig. 15 The Runtime and I/O Time of Band Registration

The total runtime and I/O curves shown in Fig. 15 indicate that both algorithms have excellent scalability. Since the NDVI algorithm has relatively low degree of algorithm complexity, the occupation of I/O time in the total runtime is relatively high. When the processor employed for implementing increases, the I/O time occupation of BR algorithm on OrangeFS ranges from 92%(1processor(4 cores)) to 99% (7 processor (28 cores)). By contrast, the I/O time occupation of NDVI_HPGFS algorithm on HPGFS ranges from 90% to 98%. In other words, when compared to the OrangeFS accelerated NDVI algorithm, about a 32.6% improvement in I/O performance with HPGFS accelerated NDVI _HPGFS give rise to a reduction in the total runtime of 32% (28 cores). From this experiment, we could say that for some algorithms like NDVI with low algorithm complexity but intensive I/O, the majority of the tie is spent in data accessing with relatively high latency. Therefore, in this situation, effective parallel I/O with high throughput would be paramount.

VII. Conclusion

Remote sensing applications are normally featured with intensive concurrent non-contiguous or even irregular I/O, which give rise to the heavy I/O burden challenges. Even now, the most popular PFSs continue to suffer from low performance of RS data accessing. The main reason for this is the lack of I/O interfaces and optimized storage data layout consistent with RS data access patterns, as well as the capability of managing multi-dimensional RS data structures.

This study presents HPGFS implemented with OrangeFS as a way to provide: 1) I/O interfaces oriented to RS data accessing patterns; 2) application aware data layout policies , and 3) direct management of multi-band RS data objects with geographical metadata. This method offers a more efficient and easier to use solution from the server side for native support of directly distributed storage and concurrent accessing of massive RS data objects in different irregular I/O patterns. Its reliance on the application aware data layout policies allows HPGFS to provide efficient data organization of multi-dimensional RS data optimized distributed data layouts for multi-dimensional RS data that are consistent with specific I/O patterns of RS applications. In this way, a more desirable data locality would be exploited for improved parallel I/O performance.

The experimental results show that HPGFS has a better performance than OrangeFS and scales well when implementing some specific RS data access patterns of RS applications. A 20% to 30% I/O performance improvement was obtained for HPGFS compared to normal PFS, which would give rise to a total performance improvement of nearly 10% to 20%. We conclude that the RS data object based parallel I/O system HPGFS provided in this paper is efficient for managing remote sensing image data.

没有

References

[1] Guo H, Fan X, Wang C. A digital earth prototype system: DEPS/CAS[J]. International Journal of Digital Earth, 2(1): 3-15, 2009.

[2] Huang, Bormin, and Antonio J. Plaza. "High-performance computing in remote sensing." Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series. Vol. 8183. 2011.

[3] Liu, Y., B. Chen, et al. (2011). Applying GPU and POSIX thread technologies in massive remote sensing image data processing, IEEE.

[4] The Global Rain Forest Mapping project - A review A. Rosenqvist, M. Shimada, B. Chapman, A. Freeman, G. De Grandi, S. Saatchi, Y. Rauste Int'l J. of Remote Sensing Vol. 21, Iss. 6-7, 2000 .

[5] Xu R, Araya-Polo M, Chapman B. Filesystem Aware Scalable I/O Framework for Data-Intensive Parallel Applications[C]. Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum. IEEE Computer Society, 2013: 2007-2014.

[6] Yong Chen; Xian-He Sun; Thakur, R.; Huaiming Song; Hui Jin, "Improving Parallel I/O Performance with Data Layout Awareness," Cluster Computing (CLUSTER), 2010 IEEE Int'l Conf. on , pp.302-311, Sept. 2010.

[7] J. M. May, Parallel I/O for High Performance Computing. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[8] Worringen, Joachim, Jesper Larsson Traff, and Hubert Ritzdorf. "Fast parallel non-contiguous file access." Proc. of the 2003 ACM/IEEE Conf. on Supercomputing. ACM, 2003.

[9] Träff J L, Hempel R, Ritzdorf H, et al. Flattening on the fly: Efficient handling of MPI derived datatypes[M]//Recent Advances in Parallel Virtual Machine and Message Passing Interface. Springer Berlin Heidelberg,109-116,1999.

[10] Song H, Yin Y, Sun X H, et al. A segment-level adaptive data layout scheme for improved load balance in parallel file systems[C]//Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE Computer Society, 414-423,2011..

[11] Micheal Moore and et al. OrangeFS: Advancing PVFS.FAST Poster Session.2011.

[12] OrangeFS: Orange File System Project. http://www.orangefs.org.

[13] Thakur, Rajeev, William Gropp, and Ewing Lusk. "On implementing MPI-IO portably and with high performance." Proc. of the sixth workshop on I/O in parallel and distributed systems. ACM, 23-32,1999.

[14] Wu J, Wyckoff P, Panda D. Supporting efficient noncontiguous access in PVFS over InfiniBand[C].Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on. IEEE, 344-351,2003.

[15] Furht, Borko, and Armando Escalante, eds. Handbook of data intensive computing. Springer, 2011

[16] I. F. Haddad, "PVFS: A Parallel Virtual File System for Linux Clusters," Linux Journal, p. 5, 2000.

[17] "High-performance Storage Architecture and Scalable Cluster File System," White Paper, December 2007.

[18] D. Nagle, D. Serenyi, and A. Matthews, "The Panasas Activescale Storage Cluster: Delivering Scalable High Bandwidth Storage," Proc. of the 2004 ACM/IEEE Conf. on Supercomputing. Washington, DC, USA: IEEE Computer Society, 2004, p. 53.

[19] Schmuck F B, Haskin R L. GPFS: A Shared-Disk File System for Large Computing Clusters[C].FAST. 2: 19, 2002.

[20] Jain R, Sarkar P, Subhraveti D. Gpfs-snc: An enterprise cluster file system for big data[J]. IBM Journal of Research and Development, 57(3/4): 5: 1-5: 10, 2013.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2014.2322362, IEEE Transactions on Parallel and Distributed Systems

[21] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. S. Ellis, and M. Best. File-Access Characteristics of Parallel Scientific Workloads. IEEE Tran. on Parallel and Distributed Systems, 7(10):1075–1089, 1996.

[22] Narayan, S.; Chandy, J.A., "I/O characterization on a parallel file system," Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2010 Int'l Sym. on , vol., no., pp.133,140, 11-14 July 2010.

[23] Thakur, R.; Gropp, W.; Lusk, E., "Data sieving and collective I/O in ROMIO," Frontiers of Massively Parallel Computation, 1999. Frontiers '99. The Seventh Sym. on the , vol., no., pp.182,189, 21-25 Feb 1999.

[24] Lu, Yin, et al. "A New Data Sieving Approach for High Performance I/O." Future Information Technology, Application, and Service. Springer Netherlands, 111-121. 2012.

[25] R. Thakur and A. Choudhary, "An Extended Two-phase Method for Accessing Sections of Out-of-core Arrays," Sci. Program., vol. 5, no. 4, pp. 301–317, 1996.

[26] P. M. Dickens and R. Thakur. Evaluation of collective I/O implementations on parallel architectures. J. of Par. and Dist. Comp., 61(8):1052–1076, 2001.

[27] Koller, Ricardo, and Raju Rangaswami. "I/O deduplication: Utilizing content similarity to improve I/O performance." ACM Tran. on Storage (TOS). vol. 6, no.3 ,pp. 13, September 2010.

[28] Zhang, Jiaying, and Peter Honeyman. "Replication Control in Distributed File Systems." Technical Report, Center for Information Technology Integration, University of Michigan, 2004.

[29] Y. Ma, L. Zhao, and D. Liu. "An asynchronous parallelized and scalable image resampling algorithm with parallel I/O." Computational Science–ICCS 2009. Springer Berlin Heidelberg, 357-366, 2009.

[30] Keying Huang, Guoqing Li, Dingsheng Liu, Wenyi Zhang. "A parallel file system based on spatial information object." Network and Parallel Computing. Springer Berlin Heidelberg, 153-162. 2005.

[31] Yu, Zhanwu, Zhongmin Li, and Sheng Zheng. "An object-based storage model for distributed remote sensing images." Geoinformatics 2006: GNSS and Integrated Geospatial Applications. Int'l Society for Optics and Photonics, 64180A--64180A. 2006.

[32] Rafique, M. Mustafa, Ali R. Butt, and Dimitrios S. Nikolopoulos. "Dma-based prefetching for I/O-intensive workloads on the cell architecture." Proc. of the 5th Conf. on Computing frontiers. ACM, 23-32, 2008.

[33] Yanying Wang; Yan Ma; Peng Liu; Dingsheng Liu; Jibo Xie; , "An Optimized Image Mosaic Algorithm with Parallel IO and Dynamic Grouped Parallel Strategy Based on Minimal Spanning Tree," Grid and Cooperative Computing (GCC), 2010 9th International Conference on. IEEE, 501-506,2010.

[34] Lawder, Jonathan K. "Calculation of mappings between one and n-dimensional values using the Hilbert space-filling curve." School of Computer Science and Information Systems, Birkbeck College, University of London, London Research Report BBKCS-00-01 August 2000.

[35] Sun X H, Chen Y, Yin Y. Data layout optimization for petascale file systems[C]. Proceedings of the 4th Annual Workshop on Petascale Data Storage. ACM,11-15, 2009.

[36] Song H, Jin H, He J, et al. A server-level adaptive data layout strategy for parallel file systems[C]. Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. IEEE, 2095-2103, 2012.

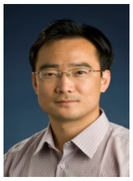
[37] Moon, Bongki, et al. "Analysis of the clustering properties of the Hilbert space-filling curve." Knowledge and Data Engineering, IEEE Tran. on 13.1. 124-141. 2001.

[38] Lawder, Jonathan. "The application of space-filling curves to the storage and retrieval of multi-dimensional data". Diss. Birkbeck College, 2000.

[39] Shan, Hongzhang, and John Shalf. "Using IOR to Analyze the I/O performance for HPC Platforms". 2007.

[40] F. Zhang, Q. M. Malluhi, T. Elsayed, "ConMR: Concurrent MapReduce Programming Model for Large Scale Shared-Data Applications", 42nd International Conference on Parallel Processing (ICPP), Lyon, France, October, 2013.

[41] F. Zhang, J. Cao, K. Hwang, and C. Wu, "Ordinal Optimized Scheduling of Scientific Workflows in Elastic Compute Clouds", Proc. 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011), 9-17, Athens, Greece, 2011



Prof. 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. Prof. Wang is a Fellow of IET and Fellow of BCS.

Dr. 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.

Albert Y. Zomaya is 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.

Prof. Dan Chen is currently a Professor, Head of the Department of Network Engineering, and the Director of the Scientific Computing lab with School of Computer Science, China University of Geosciences, Wuhan, China. His research interests include computer-based modelling and simulation, high performance computing, and neuroinformatics.

Dr. Rajiv Ranjan is a Senior Research Scientist, Julius Fellow, and Project Leader in the CSIRO Computational Informatics, Canberra, where he is working on projects related to cloud and service computing. Previously, he was a Senior Research Associate (Lecturer level B) in the School of Computer Science and Engineering, University of New South Wales (UNSW). Dr. Ranjan has a PhD (March 2009) in Computer Science and Software Engineering from the University of Melbourne. He completed Bachelor of Computer Engineering from North Gujarat University, India, in 2002. Dr. Ranjan is broadly interested in the emerging areas of cloud, grid, and service computing. The main goal of his current research is to advance the fundamental understanding and state of the art of provisioning and delivery of application services in large, heterogeneous, uncertain, and evolving distributed systems(cloud, grids, data center, and web services). Rajiv has 84 (37 journal papers, 31 conference papers, 9 book chapters, 7 books) scientific publications, approximately 70% of his journal papers and 60% of conference papers have been A

/A ranked publication, according to the ARC's Excellence in Research for Australia (ERA).