A Parallel File System Based on Spatial Information Object

Keying Huang[1,2], Guoqing Li[2], Dingsheng Liu[2] , and Wenyi Zhang[2]

[1] Graduate School of the Chinese Academy of Sciences (GSCAS)
kyhuang@ne.rsgs.ac.cn
2 Key Laboratory, China Remote-Sensing Satellite Ground Station,
Chinese Academy of Sciences
{gqli, dsliu, wyzhang}@ne.rsgs.ac.cn

Abstract. In this paper we introduced a parallel file system based on the spatial information object storage, the PIPFS system. PIPFS is a special-purpose parallel file system which designed in view of the remote sensing image processing. It uses the server/client pattern and bases on the metadata mechanism. It simultaneously accesses disks on several nodes for application I/O operations, which improves the efficiency of the operation on large scale data. A high performance is shown on high-data-complexity application, such as remote sensing image processing.

1 Introduction

1.1 Spatial Information Data Characteristics

Remote sensing technology is developing on the filed of spatial resolution, spectrum resolution, time resolution and weather condition, model, rate of observation, with which the data scale is expanding rapidly. A single scene of TM image with 7 bands can reach 280MB. The large mosaic image can be several gigabytes[1]. Different from general file data, the image data structure of remote sensing is quite complicated. The data type which is used to save pixel of remote sensing image data may be 8bit integrated, 16bit integrated, 32bit integrated or complex number and the organizational form may be BIP, BIL or BSQ. Moreover, a group of remote sensing image data often contains the same spatial attribute information.

The traditional file system is unable to combine and save the remote sensing images with their attribute information. They can be only saved separately as different files. In the remote sensing image processing process, we should keep the maximum spatial information which the image contained in order to keep the high-usability of the spatial data. Because each pixel in the image represents some spatial information, the image processing is aim at the raw form image. This limits the use of image compression technology, especially the loss-compression technology. Therefore, the remote sensing image data characteristics and data processing force us to face the problems of computing and saving the magnanimous special structure data.

H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 153 – 162, 2005.

1.2 The Storage Pattern Used in High Performance Computing

Facing the computing and saving problems of magnanimous spatial information data, high performance computing has took one good way which applied in spatial information processing and service. In high performance computing, data storage pattern affects the overall performance directly. General high performance computing storage pattern mainly includes two kinds of network storage system: (1) parallel and distributed file system, (2) data and computation separated system. The architecture of parallel and distributed file system is mainly based on computing servers. In other words, the storage and the computing are both in the same group servers. The representatives of this storage pattern are[2] message sharing mechanism such as NFS[3], Coda[4], XFS[5] and storage sharing mechanism such as VMS[6] and SFS[7]. There are two kinds of mainstream network storage construction. They are distinguished by the command collection[8]. One kind is the high-bandwidth, low-detention but high-price and bad-extension SAN（Storage Area Network) structure. The other is good-extension, low-price, easy-manage but high-protocol-spending, low-bandwidth and heavy-delay NAS（Network Attached Storage) Structure.

In view of the insufficiency of above storage pattern, the research aim at a new Linux cluster file system, object storage file system, has been launched.

1.3 From the File System Angle to Accelerate the Remote Sensing Image Parallel Processing

When the remote sensing image parallel processing algorithm executes on traditional file system, the data operation model is distribution - computing - collection. With this model, data distribution and collection process is the bottleneck of entire procedure. For example, in the image rotate algorithm which using small buffer, data I/O cost takes almost 60% of the whole time used by application[9]. Using the traditional file system and existing parallel computing model can not solve this problem. Therefore, the key job is to study a file system which adapts the characteristics of remote sensing image parallel process. This new parallel file system saves the remote sensing image data in cluster according to some distribution rules and manages the relevant spatial information and physical data distribution information in unison. In the process, through the algorithm control, the majority of data which each computing node needs can be read from local hard disk, thus reduces the network transmit time which cased by using traditional file system. The new file system can effectively accelerate the data accessing speed and the application execution.

2 PIPFS: A Parallel File System Based on the Spatial Information Object Storage

PIPFS（Parallel Remote Sensing Image Processing File System) is a parallel file system on Linux cluster, which based on spatial information object storage. In the following, we will introduce PIPFS system from tow aspects: the system structure and its support to the remote sensing image parallel process.

2.1 System Structure

As figure 1, PIPFS adopts client-server pattern based on metadata. File metadata information is managed by metadata server. Physical data is stored with distributed mode. Physical files distributed on different nodes are looked as a whole logic file. This can shield the network transfers to developers and reduce the complexity of programming and file management. PIPFS system contains three parts: metadata server, storage servers and the clients.

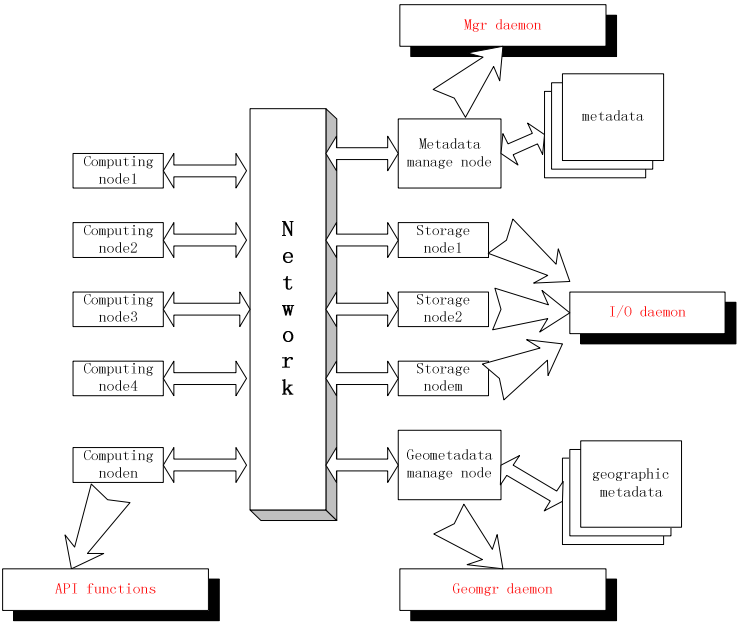

Fig. 1. Structure of PIPFS System

2.1.1 Metadata Server

The management daemon Mgrd （Metadata Manager daemon) is running on the metadata server. It is responsible for storing and managing the ordinary metadata and the spatial information metadata, doing any kind of operations on metadata, such as create, read or modify. Ordinary metadata includes the physical distributing information of image data which is distributed stored. Spatial information metadata includes the spatial attribute information that the remote sensing image has. In PIPFS, we adopt metadata centralized management--there is a unique metadata server in cluster. On one hand, it can apply a foundation for cluster to distributed store and read remote sensing data. On the other hand, it can apply convenience for users to unified manage the image data.

2.1.2 Storage Servers

In PIPFS, the physical data is distributed in storage servers which named I/O servers. The physical data management daemons Iod (I/O daemon) are running on these

servers. They are responsible for real read and write operations on local files and communication with clients. In fact, these servers create new files on local file system and access files with common operations, such as read(), write() and mmap(). It means that any local file system can store PIPFS files, such as ext2, ext3 and so on. Furthermore, we can realize fault tolerance by hard RAID or soft RAID which can create extern large file system.

2.1.3 Clients

The clients include the kernel interface module and the application program library (the application program interfaces APIs).

The kernel interface module is the interface between PIPFS system and file system manage module in Linux kernel. With this module, there are two advantages. One is that PIPFS system can be mounted as same directory in different clients. Then the users can use the files in the same directory at different clients. After installed the kernel interface module, the user may use the familiar command, such as ls, cd, rm, etc. to manage files. Most of present parallel file systems can not be visited via different operation system. Another advantage is that the directory which attached to PIPFS system can be visited by using samba protocol from windows system.

The application program library has provided the function interfaces which can be called by application programs. Application programs access all kinds of data in PIPFS through the APIs. The operations mainly consist of three kinds: operating on ordinary metadata, on spatial information metadata and accessing distributed physical data. Visiting files through the file system manage module increases the time expenses of kernel. But in PIPFS, the application program library provides a shortcut for programs to visit storage servers directly. It saves more resources for the computation. Application program library is analogous to the file system interface function library of UNIX\Linux system. This has facilitated users to develop application programs based on PIPFS system.

2.2 Support to Remote Sensing Image Parallel Processing

2.2.1 To Parallel Processing

In PIPFS, the storage server is also the computing server. In this way, the application programs can get distribution information of physical data through the metadata and control the parallel processing. As a result, the majority of data that each computing server needs can gain from local hard disk, little part of data gains from other storage servers via PIPFS. Thus, it reduces the time spend in data distributing and collecting, which caused by using traditional file system and the distribution – computing – collection pattern. Parallel programs furthest use the data exchange and manage protocol of PIPFS to improve the efficiency. The parallel disk I/O operations in different nodes accelerate accessing data in file system. Therefore, PIPFS can improve the application performance.

2.2.2 To Spatial Information Object

The spatial information object is PIPFS system fundamental unit. An object is a combination of some spatial attribute information and remote sensing image data files. But in traditional system, the file and the block are basic storage units. Users

should track the relevant spatial information while access the images. But in PIPFS system, the spatial information objects manage their attribute information via file system. And all spatial information objects have a unique object marking. Through the object marking, the users can access and operate the spatial information objects easily. Besides the I/O function interfaces which are analogous to UNIX\Linux system, PIPFS also provides some new functions which fit the remote sensing image processing:

1. Read-write data by block. Traditional file system read/write function only can read/write continual data that starts from an assigned address. A remote sensing image actually is a two-dimensional or multi-dimensional array. It frequently uses BIL or BSQ as its data organization form. Read and write data by block is the basic data accessing mode. In this mode, the efficiency of read function in traditional file system is very low. It leads to frequent I/O operations and memory redundancy. Using the read/write functions by block that provided by PIPFS system, users can access the assigned region data easily.

2. Distribution strategy control. In common parallel file system, the users almost can not control the distribution strategy of a file. They only can make some adjustments in the file distribution number or the partition size but unable to control the storage location of block data. In PIPFS system, there is a default distributed strategy, but users are able to control the distribution through APIs, too.

3. Sampling reading. For better supporting remote sensing image processing application, PIPFS system also add some commonly used functions, such as image sampling, to the file system. Users can gain the sampling date but need not to read the distributed date to local node. This facilitates developers and enhances the system's efficiency. It also avoids the frequency I/O operations and lightens the load of master node.

4. Support to save large size spatial information object. Because the using of advanced remote sensing technologies, for example high spectrum, high resolution and so on, now the single spatial information object size may amount to several hundred Megabyte. The size of spatial information object that has been processed is possible to reach several Gigabyte even dozens of Gigabyte. On the traditional file system, it is difficult to save or operate a file bigger than one Gigabyte. But in PIPFS system, the physical data are distributed, so it theoretically can save and operate arbitrary size spatial information object as will.

3 Experiments

The experimental environment is:

Meta data server is equipped with dual Xeon 2.4G processors, 2GByte ECC ram, 146GByte 1000RPM SCSI hard disk, and a 1000MBps Ethernet card.

Storage servers and computing servers constitute 8 nodes. Each node is equipped with dual Xeon 2.4G processors, 1GByte ECC ram, 160GByte 7200RPM Ultra IDE (ATA133) hard disk, and a 1000MBps Ethernet card.

The operating system is Redhat Linux 7.3.

3.1 Throughput

There is only one application program operated the data in this test. Parallel operation will be displayed in the expansibility test (see Sect. 3.3).
We compared the throughput of PIPFS with NFS in the testing environment.

Table 1. Throughout data ranged of PIPFS and NFS

Unit:MB/s

Filing system		Data scale				Read-write speed Mean value
		100MB	500MB	1GB	5GB	
NFS	Reading	21.76	23.50	21.12	20.05	21.61
	Reads in	43.52	36.83	37.05	34.81	38.05
PIPFS	Reading	21.38	22.84	22.31	21.20	21.93
	Reads in	53.91	52.45	51.62	52.03	52.50

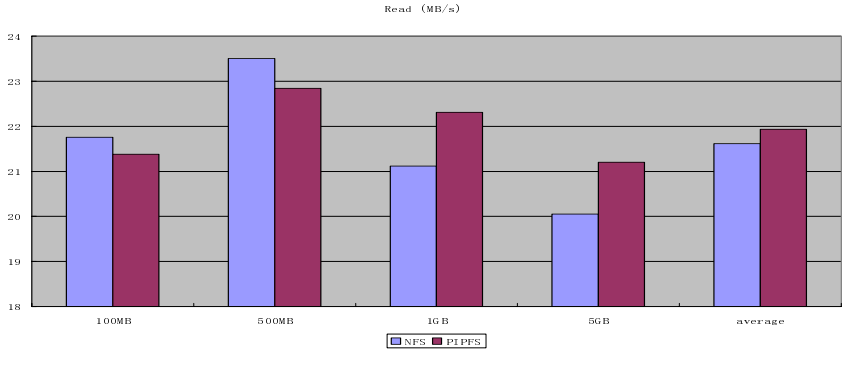

Fig. 2. Read speed of PIPFS and NFS

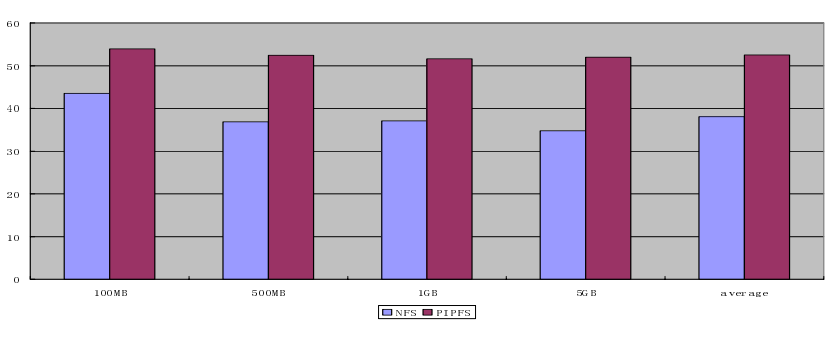

Fig. 3. Write speed of PIPFS and NFS

From the result we can see that NFS and PIPFS can perform the full performance on the Giga Ethernet. The result from different size of data indicated that when handles big data the throughput of PIPFS is much better. That is because NFS only can operate one file and PIPFS on a group of files. In the experiment we adopt NFS version3, which is an asynchronous write mode. PIPFS is built on local ext3 file system and applies asynchronous write mode in real write procedure. Therefore the write speed of NFS and PIPFS is both higher than read.

3.2 Combine PIPFS with Remote Sensing Image Processing Algorithm

The experiment explained the advantages of PIPFS combined with algorithms. The testing algorithms we chose are: (1) image rotate algorithm whose local hit rate is small in image processing algorithms, (2) unsupervised classify algorithm which the data needed by computing is stored in local storage.

3.2.1 Image Rotate Algorithm

In the experiment we used a single wave band TM image whose size is 5728*6920 pixels and rotated it 45 degree in the counterclockwise. The output image is an 8942*8941 pixels image. We use cubic convolution algorithm and gained two rotate program's run time in different parallel scales.

Table 2. Execute time of two rotate functions

Unit: Second

Function name	Parallel scale and running time			
	5 nodes	6 nodes	7 nodes	8 nodes
Rotate（MPI）*	79.5	71.0	64.9	61.7
Rotate（PIPFS）	25.2	23.6	19.8	18.1

* rotate（MPI) A function in PIPS system

In the experiment rotate algorithm is a reduced local retrieve algorithm. That algorithm's characteristic is that each computing node just deals with the local retrieve area. That brings the treatment simpler and avoids transferring the whole image to each computing node. But in PIPS system's rotate algorithm, the data is distributed by the master process through MPI. That consumes network bandwidth and computing node's memory. There is certain of resource waste. When dealing with pictures oversize there is some limits. Recur to PIPFS system, we can access the spatial information object via its global control ability. The master process's task is limited to little message transfer. The data operations are achieved by several storage servers in parallel. This can radically avoid block and improve the algorithm's efficiency. From table 2 we can see that the rotate algorithm combined with PIPFS is faster 2/3 running time than that with MPI to distribute data. We can conclude that the algorithms combined with PIPFS can obviously improve the performance.

3.2.2 Unsupervised Classify Algorithm

In the experiment, we used a single wave band TM image whose size is 5728*6920 pixels. We separated the image into 8 classes, iterative time is 10 and the threshold is 0.01. We compared algorithm using MPI with that combined with PIPFS in the main procedure's running time. Because the massage transfer time is microsecond, we ignored it.

Table 3. Execute time of tow class functions

Unit: Second

Function name		Class（MPI）	Class（PIPFS）
Master process	Read/ write	0.98	
	Transmission	4.17	
	Processing	10.32	13.38
	Receive	5.72	
slave process	Receive /Reads	3.89	3.60
	Processing	11.35	10.06
	Transmission /Writes	6.04	1.28
total		42.47	28.32

The parallel unsupervised classify algorithm's procedure is: The slave process deals with the clustering center of the image area which is distributed in each iterative. When accomplished the slaves send the result to the master. Then the master deals with the clustering center received from each slave and returns the new center to them. The slave deals with data according to the new cluster center. From table 3 we can see the algorithm with PIPFS can save the master's read/write and send/receive time compared with that using MPI. The slave's read/write time also smaller. Therefore in the whole procedure we can save above 40% time using PIPFS and the result will be much evident when dealing with bigger data.

3.3 File System Regarding Parallel Scale Extended Test

The experiment environment is:

Meta data server is equipped with two Xeon 2.4G processor, 2GByte ECC ram, 160GByte 7200RPM Ultra IDE (ATA 133) hard drive, and a 1000MB Ethernet card. The operating system is Redhat Linux 7.3.

Storage servers and compute servers constitute 8 nodes. Each node is equipped with two Xeon 2.4G processor, 1GByte ECC ram, 160GByte 7200RPM Ultra IDE (ATA133) hard drive, and a 1000MB Ethernet card. The operating system is Redhat Linux 7.3.

In remote sensing image processing, read operation is more than write. However in parallel algorithm, parallel reading the same file in the meantime is a basic operation. Therefore read performance, especially parallel read, is important to the

Table 4. Expansibility of NFS and PIPFS

Unit:MB/s

Filing system	Parallel scale			
	1 node	2 nodes	4 nodes	8 nodes
NFS	21.56	7.61	6.19	4.07
PIPFS	20.67	21.13	22.03	21.84

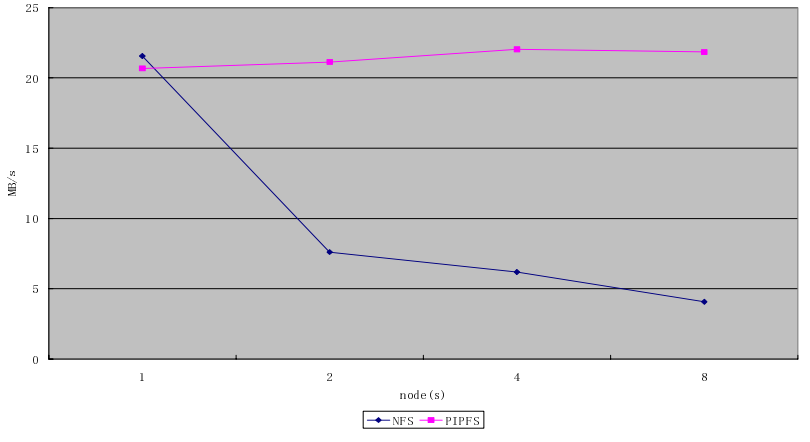

Fig. 4. Read speed of NFS and PIPFS while parallel scale increase

performance of a file system. This test assumes that the number of storage servers and computing servers is linearity grows. In this condition, compare the read speed of the bottleneck nodes in NFS and PIPFS.

From table 4 and figure 4, the test result showed that the support of NFS to the parallel scale extended was far inferior to PIPFS. PIPFS has used the multi-thread response mechanism, and the data distributed to different server. So concurrently reading can fully use the network band width. Therefore, when the parallel scale is growing, PIPFS had a higher reading speed than NFS. This test also showed that under the certainly parallel scale, PIPFS system has a good extension and it can provide an effective performance platform for the remote sensing image parallel processing.

4 Conclusion

From above tests, we can find that the efficiency of remote sensing image processing algorithm in PIPFS system is higher than the algorithm in traditional file system, because the algorithm used file system to control and operate the spatial information object. Storage based on spatial information object shield physical distribution detail and the network transmission to the developers. It greatly reduces the complexity of programming and file management. The developer need not to consider the distribution and parallel operations on data. The application program can execute the

parallel process just by calling PIPFS system interface functions and synchronizing the messages. So the entire development mode in PIPFS is even more similar to the development mentality of serial programming.

What's more, combined with Linux kernel, management of spatial information object in PIPFS is more convenient and direct-viewing. Users can operate the data but need not to enter each storage server. They can complete the operation through the metadata server. At the same time, this also increased data security on the storage server.

Reference

1. Guoqing Li, DingSheng Liu, "PIPS: A Cluster-based Parallel Remote Sensing Image Processing System", *Journal of Image and Graphics*, Vol.5 Supp. 2000
2. P. Valduriez, "Parallel Database Systems: the case for shared–something," *Proceedings of the Ninth International Conference on Data Engineering*, pp. 460-465, 1993.
3. R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh and B. Lyon, "Design and Implementation of the Sun Network File System", *Proceedings of the Summer USENIX Conference*Pp. 119 - 130, 1985.
4. M. Satyanarayanan, "Coda: A Highly Available File System for a Distributed Workstation Environment, " *Proceedings of the Second IEEEWorkshop onWorkstation Operating* SystemsSeptember 1989.
5. T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R.Wang, "Serverless Network File System," *ACM Operating Systems Review*Vol. 29, no. 5, December 1995.
6. *Digital Technical Journal*, VAXcluster Systems, September 1987. Special Issue - Number 5.
7. K. Matthews, "Implementing a Shared File System on a HIPPI Disk Array," *Fourteenth IEEE Symposium on Mass Storage Systems*, pp. 77-88, 1995.
8. Wu Qingbo, "Linux Object storage file system research",*Http://www-900.Ibm.Com/ developerWorks/cn/linux/l-ofs/index.Shtml*, 2004.10
9. Zhu Yaofei, " research and experiment in remote sensing data parallel processing file system ", *Master's degree paper,* Chinese remote sensing satellite earth station, 2002.7