# A Dynamic Extension and Data Migration Method Based on PVFS

Xiaoyu Zhang, Jie Tang[1(✉)], Heng Gao, and Gangshan Wu

State Key Laboratory for Novel Software Technology
Department of Computer Science And Technology
Nanjing University, Nanjing, China 210046

**Abstract.** With the development of the big data, The traditional file system can no longer meet the demand of High Performance Computing and Big Data. Parallel file systems are getting more and more popular in High Performance Computing. As a typical parallel file system, PVFS has been widely used in big data computing area in recent years. However with the increasing of computing scale, there exist the needs to dynamic extend data nodes, which PVFS does not support at present. This paper put forward a dynamic data node extension method as well as the subsequent data migration algorithm based on PVFS. The algorithm first adds a new data node automatically and transparently. After that, the algorithm finds out the most loaded data node in the original file system using a new load evaluation method and transfer the data into the newly added data node to mitigate the imbalance of the system. The experimental results show that our dynamic data node extension method could improve the performance of PVFS and reduce the probability of hot point effectively.

**Keywords:** Distributed File System; Parallel File System; PVFS; Dynamic Extension; Data Migration.

## 1 Introduction

With the continuous development of High Performance Computing and Distributed Computing, the traditional storage system is no longer able to meet the I/O requirements now. Parallel file system in High Performance Computing and Big Data applications are getting more and more popular for its high performance, availability and scalability. A typical parallel file system, PVFS[4][5][6], has been widely used in Big Data computing applications, and achieved good results.

However, With the increase of computing scale, the cluster nodes must be dynamically extended, which means that the file system needs to be expanded dynamically. While PVFS has a nice parallelism in read/write performance, it has deficiencies in the dynamic expansion, which is reflected in two aspects below: First, the scalability of PVFS needs to be strengthened[4][15][16] , which means PVFS has no dynamic extension. PVFS use static configuration. The

system must be restarted after you add a new node in PVFS cluster. PVFS configuration file (unlike Ceph, Hadoop, Lustre and other file systems)can not be edited manually, but is dynamically generated by pvfs2-genconfig program. Therefore, the generation of node IDs in PVFS is dynamic and unique each time. Second, PVFS has no load balancing mechanism[4][15][16]. Some program might cause the advent of hot spot node.

This paper presents a dynamic data node extension method as well as the subsequent data migration algorithm based on PVFS. The algorithm first adds a new data node automatically and transparently. After that, the algorithm find out the most loaded data node in the original file system using a new load evaluation method and transfer the data into the newly added data node to mitigate the imbalance of the system. The experimental results show that our dynamic data node extension method could improve the performance of PVFS and reduce the probability of hot point effectively.

The remainder of the paper is organized as follows: Section 2 introduces the related work about dynamically extension. Our dynamic extension and data migration algorithm is presented in Section 3. Section 4 describes the data migration algorithm framework. The experimental results and performance analysis of our algorithm are shown in Section 5. Finally, Summary and future work are drawn in Section 6.

## 2 Related Work

PVFS, developed by Clemson University is a parallel virtual file system, which is an open source parallel file system designed to run on linux. The design of PVFS fully embodies the characteristics of a parallel file system: provide high-performance of I/O[4]. PVFS currently has two versions: PVFS1 and PVFS2, PVFS in this paper is refers to the PVFS2 below.

Fig.1 describes the PVFS system structure. PVFS file system has three components: the client node (Client Node): provides file system interface to users; the metadata node (Metadata Node): manages namespace and metadata, stores metadata; the data node (I/O Node ): stores and manages data; nodes connected by the network, a physical node may have all functions of the above three components. file store in PVFS is divided into multiple sub-file in different nodes and can be concurrently access. A file is stored as metafiles and datafiles in PVFS. datafiles are split across multiple I/O node, metafile stores the distribution and attributes of the file.

Dynamic extension[16] of distributed and parallel file systems has always been a hot research problem. Different distributed file system adopt different solution for this problem, while the solution also gives the relevant data migration algorithm. The next two paragraphs describes the typical solution of the dynamic extension problem in different distributed file system.

Ceph[1] uses an object-oriented storage architecture, it takes intelligent storage devices (Object Storage Devices: OSDs) to replace a traditional hard drive. OSDs might represents CPU, network, the local cache and the underlying

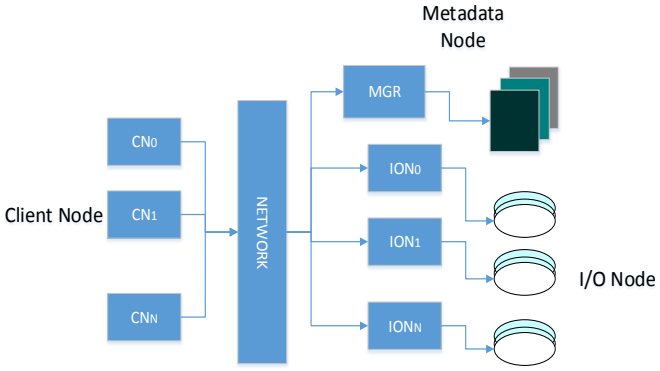



Fig. 1: PVFS system structure

disk or RAID which integrated these resources. A pseudo-random distribution algorithms (Pseudo-random data distribution function: CRUSH) used for system expansion and update in Ceph, CRUSH[1] algorithm uses a pseudo-random distribution of data, distributed to various OSDs[1] cluster maps are available through the cluster map marked.

GFS[2]: GFS converts dynamic extension problem to component failures problem. In the GFS, the component failure is common. GFS system must monitor its status uninterruptedly, detect component failure quickly and recover component failure timely. Furthermore, GFS maintain the consistency among files and its copies. When GFS extension, first add a node information to the configuration file, and reload the configuration. Thus for GFS, the dynamic extension problem is transformed into a node failure recovery problems. The system detect whether a new node on the line by heartbeat. After GFS expanded node, Mapreduce program[3] will be given priority in the below-average hard disk usage chunk server to stores the new copy. As the new node has no data, GFS will be given priority to storage file in the newly expanded node. Meanwhile, GFS will periodically load balancing and gradually fill a new chunk servers, rather than fill the new chunk servers in a short time that help prevent the new one overload. GFS data migrations through this strategy.

## 3 Algorithms

### 3.1 Dynamic Extension

Dynamic expansion of distributed and parallel file system[18] is always a hot research issues. Dynamic extension in parallel system refers to how to increase computing resources to meet the growing performance and functional requirements, or by reducing its resources to reduce costs, claimed that the system is

scalable. There are three aspects in dynamic extension: a) Functional and Performance: scalable parallel system should be able to provide more functionality or better performance after extension. b) The cost of extension: Consideration Extended spent must be reasonable. c) Compatibility: after dynamic extension, the parallel computing system (software and hardware) still work after only small changes.

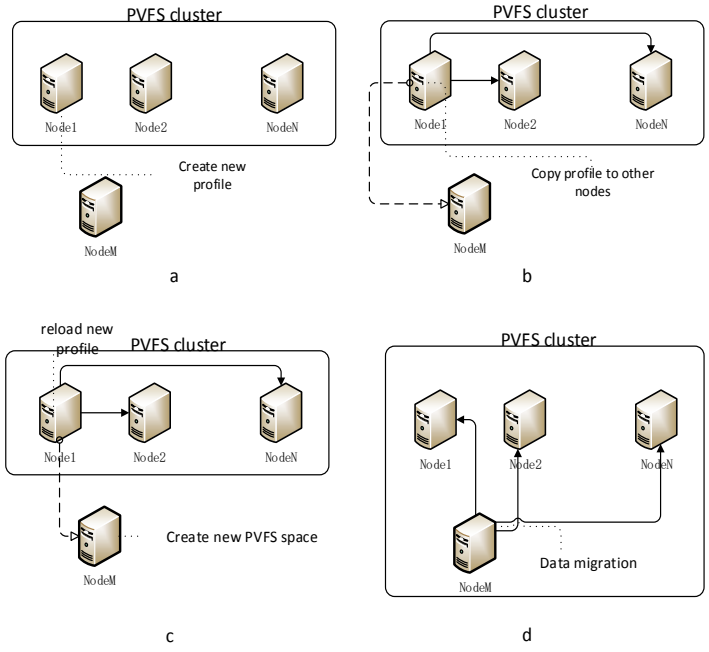



Fig. 2: Dynamic extension of PVFS

We proposes a new method to dynamic extend data nodes in PVFS. First of all, we rewrite the pvfs2-genconfig program and the related source code for adding a new function which allows the pvfs2-genconfig program read a configuration file and rebuild a new one based on the read one dynamically. As shown in Fig.2, the algorithm is divided into four steps to complete the PVFS node expansion:

a) A node is chosen randomly for re-building configuration file without shutting down the cluster by the new pvfs2-genconfig program, which inserts the new node information in configuration file and arrange a new handle to the new

node. the new handle should not conflict with the original handles . The role of the new node in the PVFS is also decided.

b) The node selected in first step sends the new configuration file to the others, including the new node.

c) We rewrite the pvfs source code to support the original system nodes reload the configuration file and read the new node information without shutting down the cluster. The new node creates a new storage space and start PVFS deamon. Whether or not the expansion is successful is tested.

d) If the expansion is successful, the system expansion is completed, the data migration algorithm starts after the expansion.

As we can see from the expansion process, the expansion process just adds a physical node into a PVFS cluster, regardless the role of the new node. Actually, the roles of the new node in PVFS are determined by user as his needed. The new node can be PVFS metadata node or datafile node or both of them. The dynamic extension algorithm only does the hardware expansion. The contents and the distribution of the file system are still not changed. There are no data stored in the extented node. Therefore, we need to consider the appropriate data migration algorithms next.

### 3.2 Stripe size

In a parallel file system, hot spot might occur when imbalanced load happened. In a file system, load balance has two meanings: the distribution of data in the file system is balanced and the visiting load of each node is balanced.

The stripe size of PVFS has a greatly effect on the hot spot problem. Assume that a distributed file system has $n_{node}$ node and each time a user will access file with size x, which is obedient to uniform distribution. A node is a hot spot only if the visiting time of this node is more than other nodes by $C$ times. $n_{visit}$ means the visit time of user . $s_{block}$ means the PVFS stripe size. the max size of file in PVFS is l.

The probability density of x is:

$$f(x) = \frac{1}{s_{file}}(l>x>0)$$

$s_{file}$ represents the size of file, and $C$ represents the visit times of the node more than other nodes.

Assume the size of data access meets the following conditions:

$$(i-1) * s_{block} < x < i * s_{block}$$

Each node store $s_{block}$ size data. Thus each access will visit i nodes in PVFS according to the conditions above.

The probability of one node visited is $f$ when an access happen. The access from different users are independent. so $C = n_{visit} * f$

Since files are stored in PVFS node sequentially, the probability of the node on which stored the visited data is:

$$f_{file} = \int_{(i-1)*s_{block}}^{i*s_{block}} \frac{1}{s_{file}} dx$$

and this access will visit i nodes for the conditions above, the probability of the node visited is

$$f_{visit} = \frac{i}{n_{node}}$$

$f_{visit}$, $f_{file}$ are independent, so $f = f_{visit} * f_{file}$, and present below, when $(i-1) * s_{block} < x < i * s_{block}$

$$f = \int_{(i-1)*s_{block}}^{i*s_{block}} \frac{1}{s_{file}} dx * \frac{i}{n_{node}}$$

we can accumulate f when the number of node is 0 up to $n_{node}$, the result is:

$$f = \sum_{0}^{n_{node}} (\int_{(i-1)*s_{block}}^{i*s_{block}} \frac{1}{s_{file}} dx * \frac{i}{n_{node}})$$

After transformation, we can get the below formula:

$$f = \frac{s_{block*(n_{node}-1)}}{2 * s_{file}}$$

$$C = n_{visit} * \frac{s_{block*(n_{node}-1)}}{2 * s_{file}}$$

According to the above formulas, we can find a positive correlation between $C$ and the stripe size in PVFS, which means we can control the stripe size in PVFS to reduce the probability of hot point.

### 3.3 Design principles of data migration algorithm

Data migration algorithm is the key part of the migration process, we analyze it from two aspect: the detail about migration algorithm itself and the application environment[11][12][13].

First, the system performance will be greatly affected by the distribution state of data migration algorithm, so we need to discuss the advantage and disadvantage between centralized and distributed migration algorithms. Generally distributed data migration algorithm has a good performance in scalability with respect to a centralized one because each node only need to decide itself. There is no single decision node in a distributed data migration algorithm, which means no single point failure. But on the other hand, if every decision node can only get part of the system information, the result of the distributed algorithms might not be the global optimal solution. Our situation is that the system has

been extended one node just now, therefore the scalability of the data migration algorithms is not necessary, hence centralized algorithm is chosen.

Data migration algorithm will migrate a large file data from one node to others. Resources for the algorithm itself is little, which requires the algorithm make decisions in a short time and without much resource supported. It also limits the complexity of the algorithm. In a real environment, the system load of each original node are likely high, the new extended node is an idle node, so data migration algorithm arbitration can be set on the new node.

Second, data migration algorithm needs to be applied to different environments, there is a huge difference in the effect of the same migration design in the different environment. So, to choose a suitable data migration algorithm based on the different environment is very important. Under the circumstances that PVFS just extended one node, and we need to migration data to the new node. So the design algorithms must be appropriate to consider this characteristics:

a). As a highly efficient parallel file system, PVFS faced the user requirement and the goal of PVFS is to achieve a high speed when program read and write data and accelerating the program performance. Therefore, the main problem to be solved in the data migration is: After the migration, how could the distribution state of the data to meet the needs of users, plus allowing users to get a higher speed in accessing the data. The load of each node become a secondary consideration issues. When considering user requirements and hardware load capability, we take the user needs as our main purpose[15].

b). We should choose those argurements obtained by a simple method for the poor resources used. we should discover the existence of load unbalance based on the consideration of the complexity of the method implementation. Also, the data migration algorithm should try to make the distribution of the file in PVFS reasonable[14].

Considering the above reasons, we choose a centralized migration algorithm with the arbitration set on the extended node. The algorithm face to user needs[13], also hot issues will be considered, and ultimately improve the performance of data throughput.

## 4 The framework of data migration algorithm

Based on the data migration algorithm of the section 3, we implement it with three subsystems: node detection module, migration arbitration subsystem, migration itself.

Node detection module: The goal of this module is to find the node congestion and load imbalances in this module. The most important thing is how to quantify a node load. Therefore, we design a performance monitoring module which running on each node for collecting node information as well as user operation and then feedback to the arbitration node.

Migration arbitration: This module is in charge for collecting data from every node in the system, generating a quantitative value for each node and making decision of what to migrate.

Migration: According to the arbitration result, migrate the selected files.

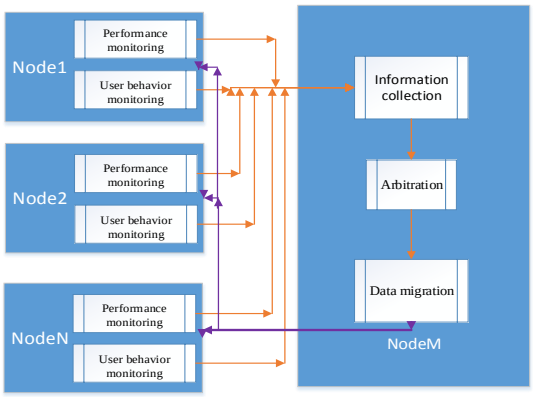



Fig. 3: Data migration algorithm

The algorithm module as shown in Fig.3. The node detection module located on each node, the remaining modules located on the extended node.

We choose the following arguments for evaluation of the load of a node as follow:

The number of access of a file recently: $c_{visit}$, for the following reasons:

a) According to the principle of locality, if a files has been accessed recently, it will be visited in the near future. Therefore migrating this file to a new node will receive higher benefits. And only a little access is the sequential access, which will easily leads to an unbalanced load in PVFS cluster and generates a hot spot node. Files in PVFS can be split into stripes in different sizes dynamically[14]. With this feature, during the file migration, we can reduce the the probability of hot spot issues by reducing the file stripe size.

b) PVFS system has cached data in datafile and metadata node. If a file is accessed recently, the file is cached in system memory. Select this file to migrate, the system can obtain files directly from memory, which is much faster than access files from hard disk.

File size $s_{file}$:

When selecting the files to migrate, we should consider the extra load generated from migrating file itself. Data migration is equal to a read and write operation. In order to minimize the impact on system performance, we should choose small files for migration.

Node load $s_{load}$:

we acquire load information through linux kernal functions. According to the above description, we decide if or not a file data should be migrated using three system feature values, i.e. network load, disk I/O throughput and load

information through linux kernal functions. Because of a hot file or a hot spot is a relative result, during the choosing of hot spots, the computed feature values are sorted first, Then, the files need to be migrated are chosen according to the computed values. Similarly, the hot spot is the node with much greater load compared to other nodes in the System. Then to make the migration process according to file information[10].

According to the above description, we can formulate the following load evaluation formula:

$$\mu = \frac{\alpha}{s_{file}} + \beta * c_{visit} + \gamma * s_{load}$$

The data migration algorithm is composed of two parts: a node information statistical algorithm and arbitration algorithm respectively.

The pseudo codes of a node information statistical algorithm is shown as below:

**Algorithm 1** Node information statistical algorithms

```
1: procedure INFORMATION STATISTICAL(input:null)
2:     loop
3:         for all file ∈ PVFS do
4:             C_VisitOfFile = file.getCvisit()
5:             S_file = file.getFileSize()
6:                                 InfoFile[file] = α/s_file + β * C_VisitOfFile
7:             Send InfoOfFile to Arbitration node
8:         end for
9:         Send LoadOfNode to Arbitration node
10:        sleep(TimeThreshold)
11:    end loop
12: end procedure
```

The node information statistical algorithm is responsible for collecting information from each node: the number of access PVFS file, size of each file and the node load, then sending the information to the arbitration node.

The pseudo codes of arbitration algorithm is shown as below:

The arbitration node accumulates the number of accesses from different nodes visiting the same file, and then chooses file to migrate according to the information collected. TimeThreshold in the algorithm is utilized to control the execution frequency of the algorithm.

According to the arbitration node algorithm, we don't take threshold method for deciding data transfer, but sort all node load evaluation value. Then we choose the highest load node for data migration. This mechanism guarantees data migration each time. It doesn't matter how to optimize $\alpha, \beta, \gamma$ in the load evaluation formula. Because it will only change the absolute value of evaluation

**Algorithm 2** Arbitration algorithm

**procedure** ARBITRATION ALGORITHM(InfoOfFile,LoadOfNode)
  **loop**
    **for all** $file \in eachnode$ **do**
      $$InfoOfFile = \sum_{i=0}^{n} InfoOfFile_{eachnode}$$
    **end for**
    Load.add(loadOfNode)
    sort Load and InfoOfFile
    Choose file migrate by the load and InfoOfFile order.
    sleep(TimeThreshold)
  **end loop**
**end procedure**

function, and the relationship of size between each node load evaluation will not change. So we set value of $\alpha, \beta, \gamma$ equal to 1 in the next experiments.

## 5 Experiments

In this section, a series of experiments are carried on to verify our data migration method's effectiveness and efficiency. We do our experiments on a cluster with four nodes, each node has 2 2.1GHz Intel Xeon E5-2620 processors, 64GB of memory and a hard disk of 2 TB. The nodes of the cluster are connected by InfiniteBand with bandwith 40Gb/s, and the operating system is CentOs6.5.

First, we deploy PVFS in three nodes, then dynamic extend a new node in PVFS. We access data in PVFS using our reading program in the next twenty-five minites. the read speed of each node is shown in Fig.4.

The node 4 is the extented node as we can see. In our experiments, we execute the arbitration algorithm each 60 seconds, i.e. TimeThreshold equals to 60s. We can see from the fig.4 that the speed of the node4 is growing while the others speed are stable.

Next, we discuss the affection of different values of TimeThreshold. Experimental results as shown in fig.5.6.

As shown in the fig.5. When TimeThreshold value is greater than 30 seconds, the speed of the extension node grows smoothly, the original nodes keep stable. When TimeThreshold value is less than 30s in fig.6 below, the system of the reading speed is very poor in a period of time. This is because the data migration algorithms perform too frequently, caused a huge influence in the PVFS system. PVFS can not afford to respond to user requirements. So choosing TimeThreshold greater than 30s, the PVFS can perform smooth migration, and ensure the needs of users at the same time.

Finally, we test our algorithm for hot spot issues. First, for a period of time, we make the program randomly read data in PVFS, then execute our data

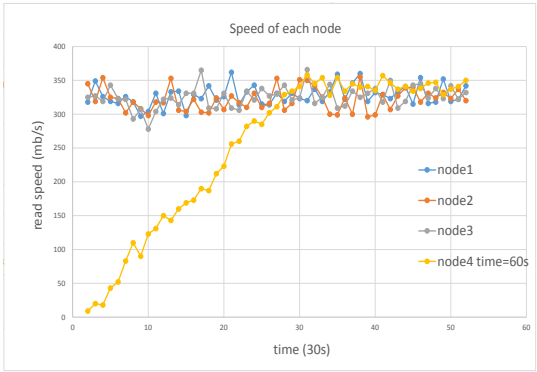



Fig. 4: Speed of read speed for each node when data migrate

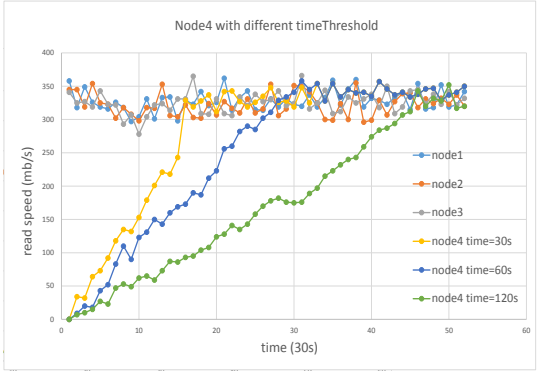



Fig. 5: Speed of each node with different data migration time

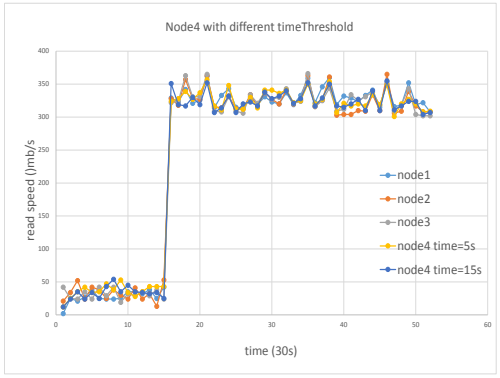
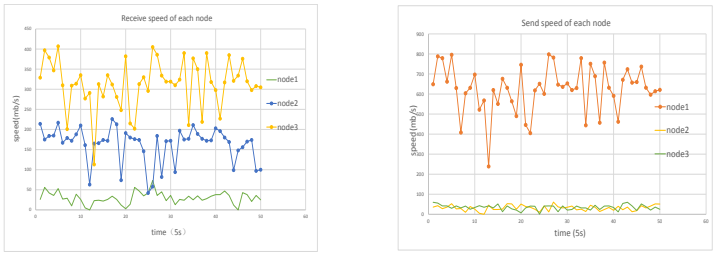



Fig. 6: Speed of each node with different data migration time

migration algorithm, perform the same operation, the results shown in fig.7.8. As can be seen in the fig.7, before data migration, we randomly read file in



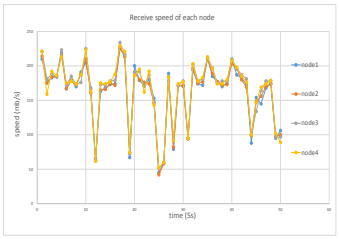

(a) Speed of network to receive (b) Speed of network to send

Fig. 7: Speed of network before data migration

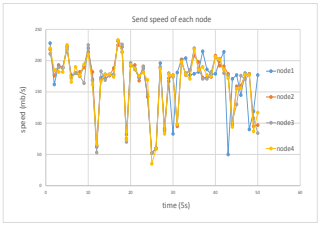
PVFS and all data access stored in node 1. Node 1 become a hot spot node. In the fig.8, in the data migration process, the algorithm detects hot spot, and split file in the hot spot, and then distribute the file to other nodes. In fig.9, the send and receive amount of four nodes are balanced.

## 6 Conclusion

We present a dynamic node extension method based on PVFS, which can add a new node automatically and transparently. After that, we design a data migra-




(a) Speed of network to receive




(b) Speed of network to send

Fig. 8: Speed of network after data migration

tion method to find out the most loaded node in the original file system using a new load evaluation method and transfer the data into the newly added node to mitigate the imbalance of the system. Experiments prove it can effectively meets user demand, while reducing hot spot occurrence probability. However, our load evaluation method is a little simple and we do not consider the situation that more nodes added at a time. After the migration is completed, there still exist a hot spot potentially. So in the future work, we consider to design an algorithm dealing with node load reasonable and add more nodes at a time.

## Acknowledgments

We would like to thank the anonymous reviewers for helping us refine this paper. Their constructive comments and suggestions are very helpful. This paper is partly founded by National Science and Technology Major Project of the Ministry of Science and Technology of China under grant 2011ZX05035-004-004HZ.

## References

1. Weil S A, Brandt S A, Miller E L, et al.: Ceph: A scalable, high-performance distributed file system[C]//Proceedings of the 7th symposium on Operating systems design and implementation. USENIX Association, 2006: 307-320.
2. Ghemawat S, Gobioff H, Leung S T.: The Google file system[C]//ACM SIGOPS operating systems review. ACM, 2003, 37(5): 29-43.
3. Dean J, Ghemawat S.: MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
4. Haddad I F.: Pvfs: A parallel virtual file system for linux clusters[J]. Linux Journal, 2000, 2000(80es): 5.
5. Kuhn M, Kunkel J M, Ludwig T.: Dynamic file system semantics to enable metadata optimizations in PVFS[J]. Concurrency and Computation: Practice and Experience, 2009, 21(14): 1775-1788.

6. Tantisiriroj W, Son S W, Patil S, et al.: On the duality of data-intensive file system design: reconciling HDFS and PVFS[C]//Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011: 67.
7. Wu J, Wyckoff P, Panda D.: PVFS over InfiniBand: Design and performance evaluation[C]//Parallel Processing, 2003. Proceedings. 2003 International Conference on. IEEE, 2003: 125-132.
8. Pfister G F.: An introduction to the infiniband architecture[J]. High Performance Mass Storage and Parallel I/O, 2001, 42: 617-632.
9. Hsiao H C, Chung H Y, Shen H, et al.: Load rebalancing for distributed file systems in clouds[J]. Parallel and Distributed Systems, IEEE Transactions on, 2013, 24(5): 951-962.
10. Wang K, Zhou X, Li T, et al.: Optimizing load balancing and data-locality with data-aware scheduling[C]//Big Data (Big Data), 2014 IEEE International Conference on. IEEE, 2014: 119-128.
11. Kunkel J M.: Towards automatic load balancing of a parallel file system with subfile based migration[J]. 2007.
12. Kobayashi K, Mikami S, Kimura H, et al.: The gfarm file system on compute clouds[C]//Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on. IEEE, 2011: 1034-1041.
13. Dong B, Li X, Xiao L, et al.: Self-acting load balancing with parallel sub file migration for parallel file system[C]//Computational Science and Optimization (CSO), 2010 Third International Joint Conference on. IEEE, 2010, 2: 317-321.
14. Jenkins J, Zou X, Tang H, et al.: Parallel data layout optimization of scientific data through access-driven replication[R]. Technical Report-Not held in TRLN member libraries, 2014.
15. Soares T S, Dantas M A R, de Macedo D D J, et al.: A data management in a private cloud storage environment utilizing high performance distributed file systems[C]//Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on. IEEE, 2013: 158-163.
16. Yanmei Huo, Kexin Yang, Liang Hu, et al.: Summary of parallel file system research[J]. Journal of Chinese Computer Systems.2008, 29(9): 1631-1636.
17. Congping Zhang, Jianwei Yin, et al.: Dynamic load balancing algorithm of distributed file system[J][J] Journal of Chinese Computer Systems. 2011, 32(7): 1424-1426.
18. Yingzhi Zhu, Bingfeng Li, Tingting Sun, et al.: Parallel computing system scalability[J]. Computer Engineering and Applications:2011, 47(21): 47-49.