International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp. 59-78
http://dx.doi.org/10.14257/ijdta.2014.7.4.06

Design and Implementation of Multiple Volume Servers Block Level Storage System based on Load-Balanced Strategy

Jian Wan[1*#], Xun Chen[2*#] and Xindong You[3*#]

[*]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China*
[#]*Key Laboratory of Complex System Modeling and Simulation, Ministry of Education*
[1]*wanjin@hdu.edu.cn,* [2]*cx787@163.com,*[3] *youxindong@hdu.edu.cn*

Abstract

Block level storage system has been widely used as it can provide users raw block device. On the basis of Orthrus system, this paper designs and implements a multiple volume servers block level storage system based on load-balanced strategy-Orthrus Plus. This system uses the dynamic load-balanced strategy based on genetic algorithm strategy to load balance the volume servers clusters and proposes an iSCSI pause and resume method to solve the breakpoint retransmission problem during load-balancing. At last, we design a hierarchical management mechanism based on volume life cycle at storage. Experiments show that the load balancing capability in Orthrus Plus is more accurate and efficient under the situation of larger amount of data, and the hierarchical storage strategy also increases the I/O throughout capacity.

Keywords: *cloud storage, load balancing,genetic algorithm, iSCSI pause and rusume;hierarchical storage strategy*

1. Introduction

With the widely use of cloud computing technology, as well as the surge of global amount of user data , cloud storage fields also began to grow up.

Cloud storage is an extension and development of the concept of cloud computing,it means to gather a variety of heterogeneous network storage device to work together by using clustering technology, distributed file system, grid technology, *etc*. It provides users safe, reliable, efficient data storage and access service.

According to the difference of storage level, cloud storage system can be divided into two parts: file level storage and block level storage. File level storage system mainly refers to distributed file system. It provides file level storage access services through the Network File System protocol. Such as Hadoop Distributed File System(HDFS)[1] ,Google File System(GFS)[2],Amazon Simple Storage Service(S3)[3]and so on. While Block level cloud storage system provides users row block level storage resources via Small Computer System Interface(SCSI) or Fibre Channel(FC). Like Amazon Elastic Block Store(EBS)[4],Virtual Block Store(VBS)[5] from Indiana University and Orthrus from Hangzhou Dianzi University[6].

Because of the block level storage's high versatility, flexibility and other characteristics, it is widely used in file storage, database storage, the virtual machine file system volume[7].But different block level storage products has their own shortage. (1)Amazon EBS has high coupling with its own platform, which is the disadvantage of integration between different

ISSN: 2005-4270 IJDTA

systems and second development; (2)VBS only uses one single node as volume server, that is likely to cause single point of failure, thus affecting the stability of the system. (3)Orthrus adopts multiple volume servers to improve the single point of failure. Meanwhile, it uses the mutated genetic algorithm strategy to ensure the load-balancing between each volume servers. However, when Orthrus uses the load-balancing strategy and the iSCSI [8] connections will be reset, if a user is uploading data at that time, after the new iSCSI connection is rebuilded, user should re-upload the data. This lowers the system efficiency. Besides, as Orthrus uses the mutated genetic algorithm strategy, which is lack of mutation, it is easily to get the locally optimal solution, rather than the global optimal solution. At last, as users of the system gradually increased, with the amount of data, the back-end storage pool is also extended to the many heterogeneous storage devices. Orthrus system lacks full use of existing storage resources strategy, as well as depth and unified management of heterogeneous storage devices and logical volumes.

Therefore, this paper has implemented a multiple volume servers block level storage system based on hierarchical storage strategy——Orthrus Plus. This system focuses on the problems such as local optimum in load balancing, iSCSI breakpoint retransmission and the lack of unified management of heterogeneous storage devices. It does certain improvements in load balancing. Firstly, put the mutation process of genetic algorithm into the load balancing strategy to improve the accuracy of global optimal solution as much as possible. Secondly, this paper has designed an iSCSI pause-and-resume oriented method. After reset the iSCSI connection, users who are uploading data do not need to re-upload data. System will continue to upload from the breakpoint automatically. This improves the system performance. Thirdly, we proposed a hierarchical storage management mechanism based on volume life cycle through the existing heterogeneous storage environments. Lastly, we have verified through experiments that the load balancing strategy in Orthrus Plus is more reliable and efficient under large amount of data, and the performance improvement on hierarchical storage management mechanism on the overall system.

In this paper, Section 2 designs the basic architecture of Orthrus Plus system, and briefly introduces the process of logical volume management and failure detection in Orthrus Plus; Section 3 adopts volume server load model and performance model, and proposes an improved load balancing strategy based on genetic algorithm. Meanwhile, we propose an iSCSI pause-and-resume method to improve the system's load balancing efficiency; Section 4 proposes a hierarchical storage management mechanism based on volume life cycle; In Section 5, we compare the experiments of before and after improve the load balancing strategy, analysis the performance of iSCSI pause-and-resume, and verified the performance improvement on hierarchical storage management mechanism on the system access efficiency; Section 6 concludes this paper and proposes the next steps.

2. Architecture of Orthrus Plus

2.1. Introduction of System Architecture and Module Function

This paper designs a kind of multiple volume servers block level cloud storage system based on hierarchical storage strategy—Orthrus Plus. It can allocate storage resources for virtual machines according to users' requirement. The system adopts multiple volume servers architecture, and volume servers faults' monitor-detection-switching mechanism to avoid single point of failure as much as possible, and strengthen the stability of the entire system.

Orthrus Plus system mainly consists of seven modules. They are Orthrus Plus client module, Orthrus Plus delegate module, Volume server status listening module, Volume

delegate module, Load monitoring module, Status monitoring module and VMM delegate module. Except that the Orthrus Plus client module is deployed on the Orthrus Plus Client, the others are respectively deployed on services node, and details are shown in Table one. All management monitor modules packed their interface functions, and publish them through web service[9],these functions will be invoked by other modules for achieving functional assignments together like creating/deleting volume, attaching/detaching volume. Figure 1 is the Orthrus Plus system architecture.

Table 1. Orthrus Plus System Functional Module and Deployment Position Table

Functional Module	Deployment Position
Orthrus Plus client module	Orthrus Plus Client
Orthrus Plus delegate module	Orthrus Plus Server
Volume server status listening module	Orthrus Plus Server
Volume delegate module	Volume Server
Status monitoring module	Volume Server
Load monitoring module	Volume Server
VMM delegate module	VMM Server

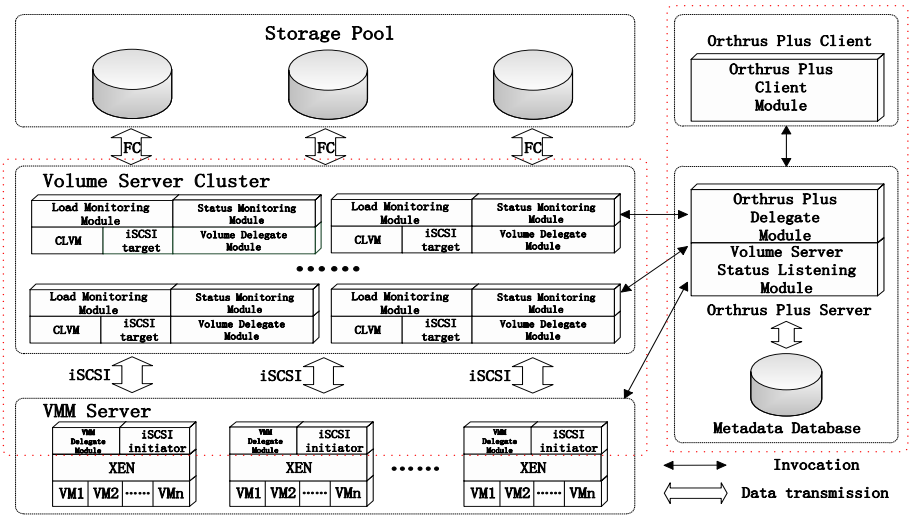

Figure 1. Architecture of Orthrus Plus

Orthrus Plus client is deployed on users' client through which user can creating, deleting, attaching and detaching commands, and receive feedback information. Orthrus Plus system management module is mainly used for receiving commands from client module, and then send command to volume management server, VM management server according to command type, and receive feedback information, store logical volume, snapshot, and logical attaching information into database.

Volume server status listening module send volume server running status request and volume server load status request to every node of volume server cluster timely to get the runtime status and load of every volume server. When find errors on one volume server, it will start error handling module to switch volume server immediately. And after getting the volume server's load data, when find that the server's load is not balance, it will invoke Load monitoring module, run the load balance strategy to make the load balance.

Volume server cluster use CLVM [10] to co-manage back-end heterogeneous storage devices and synchronize information of volumes and snapshots. Volume delegate module takes advantage of CLVM interface to achieve functions of creating/deleting volume/snapshot. Meanwhile, it uses the iSCSI Enterprise Target(IET) [11]on volume server to map a volume to an iSCSI target for connecting from iSCSI initiator to VMM server.

Status monitoring module regularly invokes CLVM functional interface to estimate if the volume server is on the work state or not, and returns the result to Volume server status listening module.

Load monitoring module can record each iSCSI connection's load in this volume server in real time through the TCP port and the total load in the volume server, and return the result to Volume server status listening module.

VMM delegate module mainly provides functions of attaching and detaching volumes for users' VMs. At first, it connects to the published iSCSI target on volume server through iSCSI initiator to get the volumes that users need. And then, VMM delegate module use Virtual Block Device (VBD) [12] technology, attach the block device from Domain 0[13] to user's VM in Domain U [14] so that users can use this remote block device.

2.2. Failure Listen-Detect-Switch Process

System failure listen-detect-switch process can be divided into listen-detect and listen-switch two parts.

(1)Listen-Detect:Orthrus Plus system management module will send volume server runtime status request to each volume server regularly and the status monitor module in volume management server will invoke CLVM interface to judge whether the volume server is normal or not. If some error occur, it will return error information to volume server status monitor module, and then the system will enter into Listen-Switch period.

(2)Listen-Switch:When it finds thar some errors occur in one volume server, the system will cut off all the iSCSI connection with the volume server, and then choose another one volume server, republish iSCSI connection to VMM server through it. At the same time, if there is new attach request, the system will automatically avoid this volume server.

3. Dynamic Load Balancing Strategy and ISCSI Pause and Resume Technology

Orthrus Plus system adopts the architecture of multiple volume servers, and use the error listen-detect module to deal with the problems in volume server which can effectively avoid the single point of failure problem. However, in the structure of multiple volumes, for the reasons of volume detaching and so on, volume servers' load is usually not balance. So, how to make load balance and keep the system steady is a key problem that need to solve immediately. Meanwhile, when volume server cluster re-allocate the iSCSI connections according to some kind of load balancing strategies, if some users are uploading the data through the iSCSI connections, then after the connections are re-connected, users need to re-

upload these data. It undoubtedly will lowers the system efficiency and adds extra weight to the system.

Therefore, in the first half of this section, we will focus on a dynamic load balancing strategy based on the performance of volume server, and introduce a method to achieve iSCSI pause-and-resume in the second half.

3.1. Performance Model of Volume Server

There are many factors [15] affect the performance of volume server, such as CPU frequency, memory size, network bandwidth, operation system. So it's hard to quantize the performance of volume server directly using White Box method. But under the same load, the IO performance of the logical volumes on the VMs will be affected by the different performance of volume server. Therefore, we can use the volume performance under the same load to characterize the performance of volume server.

This paper studies the relationship between the different characteristics of load (OIOs, %Randomness, %Read, IO Size) and the IO performance of volume (Average Response Time, Throughput, IOPS) under the different network bandwidth through experiments. The experiment figure shows that there is a good liner relation between OIOs and Average Response Time(ART) (Figure 2).So the slops of ART-OIOs line can characterize the relative performance of volume servers.

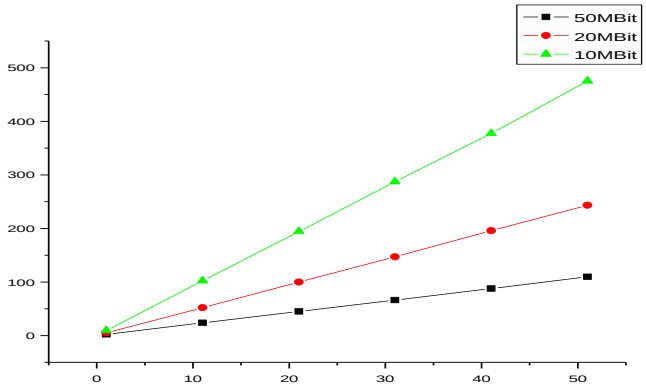

Figure 2. Relation between OIOs and ART

Definition 1 (Performance Model of Volume Server):If there are M volume servers in Orthrus Plus system. Their relative performance values are $\{c_1, c_2, \ldots c_m\}$, and the corresponding slops of OIOs-ART line is $\{k_1, k_2, \ldots k_m\}$. Let $c_1=1$, then for $\forall c_i$,

$$c_i = \frac{k_i}{k_1} * c_1 = \frac{k_i}{k_1}, \quad i=1,2,\ldots,m \tag{1}$$

3.2. Load Model of Volume Server

The main function of volume server is publishing the logical volume as an iSCSI target, and ready for connecting with VMM server and attaching to users' VM. So the main load of volume server relays on a serious of IO operations to the volumes through the iSCSI

connections with users' VMs. Therefore, the total load of one volume server can be indicated by the sum of all the IO load on it.

Definition 2 (Load Model of Volume Server):For any volume server V_i in volume server cluster, let assume that there are n iSCSI connections , the load of each connection is W_k,, then the total load L_i of volume server V_i is:

$$L_i = \sum_{k=1}^{n} w_k \tag{2}$$

Where the number of connections n is recorded in IET, the load of each connection can be got from Orthrus Plus load monitor module on volume server through listening the 3260 port.

In order to match the load of volume server with its performance, this paper defines the performance weighted load model as follows.

Definition 3 (Volume Server Performance Weighted Load Model):If the performance weighted load value of M volume servers in Orthrus Plus system are $\{Lc_1, Lc_2, \ldots, Lc_m\}$, then

$$Lc_i = L_i / c_i, \quad i = 1, \quad 2, \quad \ldots, \quad m \tag{3}$$

Where L_i is the total load of volume server V_i, which can be calculated from Definition 2,and c_i is the performance value of volume server V_i, which can be get from Definition 1.

3.3. The Volume Server Dynamic Load Balancing Strategy

3.3.1. The General Framework of Dynamic Load Balancing Strategy: If there are M volume servers $\{V_1, V_2, \ldots, V_m\}$ in Orthrus Plus system, their performance values and load values are $\{c_1, c_2, \ldots, c_m\}$ and $\{L_1, L_2, \ldots, L_m\}$, as the load size in each iSCSI connection is discrete, it's difficult to reach the ideal status, when each volume server reaches the load balancing status

$$Lc_1 = Lc_2 = \ldots = Lc_m \tag{4}$$

also is

$$\frac{L_1}{c_1} = \frac{L_2}{c_2} = \cdots = \frac{L_m}{c_m} \tag{5}$$

In Orthrus Plus system, if any two volume servers Vi, Vj can reach the status in (6),we judge the volume servers in system are in the load-balancing status.

$$| Lc_i - Lc_j | \leq \omega, \; i, j = 1, 2, \ldots, m; \; i \neq j \tag{6}$$

Where ω is a defined threshold according to the actual requirement in the system.

Volume server status listening module in Orthrus Plus System Manage Server invokes the load monitoring module in Volume Server in every time τ, to obtain the load condition in each volume server. Then it calculates the performance weighted load value according to the Definition 3.

Definition 4 (The Volume Server Unbalanced Degree Value):In Orthrus Plus system,the volume server unbalanced degree value

$$T(\{ Lc \}) = Lc_{max} - Lc_{min} \tag{7}$$

Where Lcmax is performance weighted load values maximum in the systems, Lcmin is performance weighted load values minimum in the systems.

When the unbalanced degree value T({ Lc }) is larger than certain threshold ω, Orthrus Plus system will start the dynamic load balancing strategy, choosing two volume servers whose performance weighted load values are the maximum and the minimum. Then reset their iSCSI connections.

3.3.2. Dynamic Load Balancing Strategy Based on GA: Orthrus system uses the dynamic load balancing strategy based on mutated genetic algorithm, it produces the next generation via mutation and roulette wheel, and not join the crossover operation in genetic algorithm [15]. In genetic algorithm, as the global searching ability is reflected by crossover operation, if simply rely on the mutation operation; it is likely to obtain a local optimal solution, rather than the global optimal solution. Therefore, this paper has improved this existing issue in Orthrus system load balancing strategy, the mproved dynamic load balancing strategy processes are as follows:

(1) Coding:In this paper, we use the common binary coding. Suppose the two volume servers A and B with the maximum and minimum performance weighted load value according to this system, their iSCSI connection numbers are n1 and n2. Order these connections in chronological order, then create a 0/1encoded string with the length of n1+n2. For example, string "0110001110", means that there are 10 iSCSI connections in this two volume servers. The 1st, 4th, 5th, 6th, 10th connections(0 bit) are connected with volume server A, while others(1 bit) are connected with volume server B.

(2) Selection:This paper defines the fitness function is

$$F(\mathrm{i}) = 1 \Big/ \Delta L_c = 1 \Big/ \left| Lc_a - Lc_b \right|, \quad \mathrm{i} = 1, 2, \cdots, \mathrm{k} \tag{8}$$

Where (I) Lca and Lcb are the performance weighted load values of corresponding iSCSI connections' allocation plan in volume server A and volume server B; (II) k indicates the total number of chromosomes in the population. Then, using roulette wheel selection, the probability of being selected of each individual is

$$P(\mathrm{i}) = \frac{F(\mathrm{i})}{\sum_{j=1}^{k} \mathrm{F}(\mathrm{i})}, \quad \mathrm{i} = 1, 2, \cdots, \mathrm{k} \tag{9}$$

Accumulation probability is

$$q_i = \sum_{j=1}^{i} P(\mathrm{i}), \quad \mathrm{i} = 1, 2, \cdots, \mathrm{k} \tag{10}$$

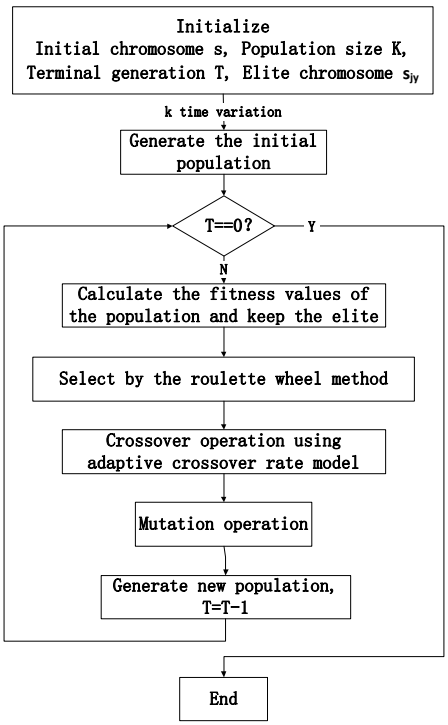

Figure 3. GA-based Load Balancing Strategy Flow Chart

(3) Crossover:In basic genetic algorithm, crossover rate is a fixed value. But a fixed crossover rate is hard to meet the needs of the constantly evolving population. At the beginning of the search, a bigger crossover rate could improve the searching efficiency of algorithm; but in the latter, as the individual fitness values have generally increased, small crossover rate will make the good individual genes are not excessively damaged. Therefore, this paper designs an adaptive crossover rate model based on the cosine function, adding the single point crossover method, to achieve the purpose of cross a pair of chromosome.

Definition 5 (Adaptive Crossover Rate Model) As to the crossover rate Pc in an iterative process,

$$Pc = \begin{cases} \cos(\frac{\pi}{2} * \frac{t}{\tau}) & (Pc > Pc_{\min}) \\ Pc_{\min} & (Pc \le Pc_{\min}) \end{cases} \quad (11)$$

Where (I)t denotes the current generations; (II) T means the total generations in the algorithm; (III) Pcmin takes the empirical value of 0.6[16].

(4) Mutation:After a global search by crossover operation, mutation operation can obtain a batter effect in local search. The mutation operation defined in this paper, will randomly change a 0 bit to 1 and change a 1 bit to 0 at the probability of 50%. This ensures the probabilities of mutate to two directions are equal.

The entire dynamic load balancing strategy based on GA can be expressed as follows: First, encoding the iSCSI connections in two load balancing volume servers selected by the system to a 0/1 string, and calling this string s as the initial chromosome. Then, mutate k

times on initial chromosome to generate k offspring s1, s2, …, sk as initial population. Second, calculate the fitness value and cumulative probability of all individuals in the population. In order to prevent the good individual genes is forced to destroy, this paper adopts the elitism strategy. That is as each time, after produce offspring, comparing them with the elite and keeping the individual with the highest fitness value as the elite. Then, using the roulette wheel method to select k offspring, and taking crossover and mutation operation. Repeat the above process until reach the defined generation. The algorithm flow chart is shown in Figure 3. At last, re-allocate the iSCSI connections in two volume servers according to the coding calculated by above procedures.

3.4. An iSCSI Pause and Resume Method

The volume server dynamic load balancing strategy based on genetic algorithm effectively ensures the volume server cluster's ability in load balance, improving the performance of system. However, when the volume server cluster resets the iSCSI connections according to this strategy, if there are some users uploading the data, then after the iSCSI connections have been re-established, users should have to re-upload the data, this no doubt increases the burden of the system, and waste user time. Therefore, this paper designs an iSCSI pause-and-resume method to solve this problem.

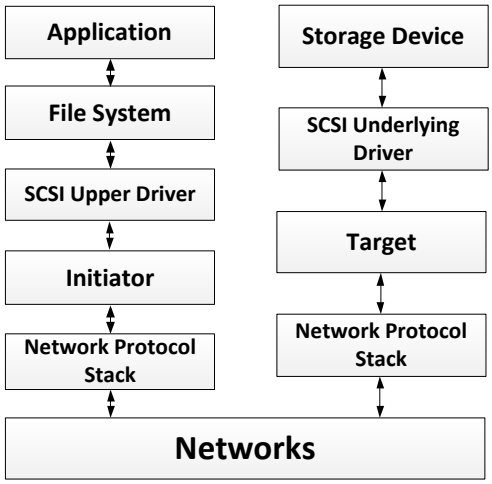

Figure 4. Process of Dealing with the SCSI Commands and Data by iSCSI Protocol

iSCSI mainly transmit SCSI command and data through TCP/IP network, for users to connect and manage the remote storage. When user accesses the iSCSI target device via iSCSI initiator, it will access the file system through the application, the file system analysis the IO command to get the corresponding device and address, and transmit the request to SCSI upper driver. Then the driver changes the IO command to SCSI command and delivers to iSCSI initiator. iSCSI initiator capsulate iSCSI protocol header into commands and data, sending to the iSCSI target through the IP network. After iSCSI target receive the command package, it sends the package to SCSI underlying driver. The driver completes the IO operations and transfers to users through the opposite direction [17]. The whole process of dealing with the SCSI commands and data by iSCSI protocol is shown in Figure 4.

When users access the remote storage by iSCSI, it generally invokes the iSCSI interface through file system, so it is more reasonable and convenient to achieve the iSCSI pause-and-

resume from the file system level. In this paper, we use log file to record these information including the source address and destination address of the file, realize the pause-and-resume technology. When user uploads a file to the iSCSI volume, system will firstly search the log file according to the source address and destination address to see if there is a record. If there is no corresponding record, which means this file is the first time to upload, system will record the source and destination address in log file, the record will be deleted after the transfer is complete. If find the record in log file, which means it is a resume. First, system will use a specific function to calculate the size of uploaded file accord to the record in log file. Then set the read pointer to the breakpoint, and read the remaining uploading file into the buffer, at the same time, set the write pointer to the end of destination file, and write the data from buffer. After the upload is finished, system will delete the record in log file.

4. Hierarchical Storage Management Mechanism Based on the Life Cycle of Volume

With the increase of Orthrus Plus system user, data also showed explosive growth, the data access speed requirements of users are also increasing, these make traditional data storage methods have been unable to meet people's needs. The contradiction between cost of storage and data access performance is a great challenge for us. This paper designs a hierarchical storage management mechanism based on the life cycle of volume, use the existing heterogeneous storage resources, save storage costs, improve data access efficiency.

4.1. Hierarchical Storage Profile

Hierarchical storage based on Information Lifecycle Management [18], according to the importance of data, frequency of access, storage costs and other indicators, the data is stored in storage devices of different performance levels (SSD, ordinary hard drive, CD-ROM, tape, *etc.*). Meanwhile, through the hierarchical storage management mechanism [19], artificially realize the data migration between storage devices, eventually high access frequency data is stored in high-performance and high-cost storage devices, less frequently accessed data is stored in low performance and low-cost storage devices, make the cost of the storage system tends to low price devices, and overall performance tends to be high performance devices.

Hierarchical storage can generally be divided into online storage, near-line storage, and offline storage. Online storage stores data on high speed disk devices, its performance and access rate is good, but its price is expensive. Offline storage stores data on the low speed disk devices, its access rate is low, but its price is low. Near-line storage is between online storage and offline storage. The performance and price of its disk devices is between two other method.

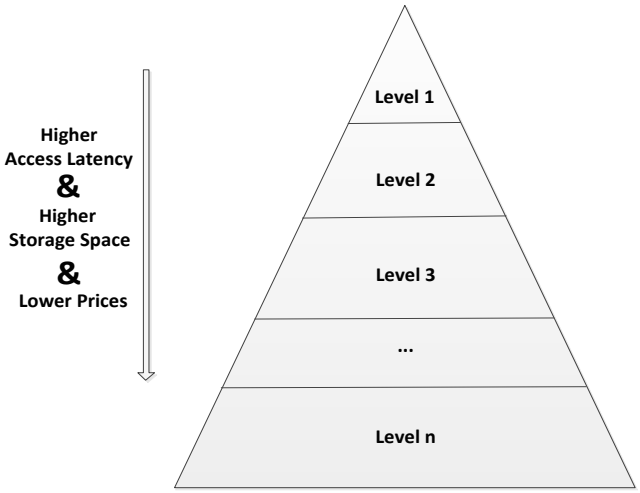

Figure 5. Hierarchical Storage Pyramid Structure

Hierarchical storage structure is shown in Figure 5, it make full use of existing heterogeneous storage resources, utilize the advantages and features of each levels. It can effectively improve the storage system performance, reduce overall storage costs, and strengthen the unified management and protection of data.

4.2. Dynamic Volume Migration System Structure and Volume Migration Strategy

According to the characteristics of hierarchical storage management, this paper add the hierarchical storage management mechanism based on the life cycle of volume into Orthrus Plus system, the design and implementation are mainly from two aspects, the volume migration architecture and the volume replacement and migration strategy.

4.2.1. Design and Realization of Dynamic Volume Migration System Structure; Hierarchical storage management system based on the life cycle of volume is mainly composed of the following modules:

(1)Volume Migration Trigger: Considering the newly created volumes may be used by the user as soon as possible, therefore, the volume created by default must be created on the OnStore storage devices. But when idle resources of online storage devices are insufficient or the access frequency of one volume is higher than the average access frequency of all the volume in primary storage devices, volume migration trigger VMT is invoked, which started the whole migration process.

(2)Volume Value Judgment Manager: Volume Value Judgment Manager mainly obtain real-time volume information from the volume metadata server regularly, including the number of volumes, size, frequency, etc. Then according to the collected data and certain strategy (see 4.2.2), Volume Value Judgment Manager create volume migration queue in advance. When VMT is triggered, VVJM will send the information about the head volume of the migration queue to the Volume State Detector.

(3)Volume State Detector: After Volume State Detector receive the information from VVJM ,VSD will detect the volume usage. If the volume is detected being used, it will send feedback error to the VVJM, and VVJM will send the new volume information to VSD; If the

volume is detected not in use, the VSD will continue to send the volume information to the Volume Migration Executor.

(4)Volume Migration Executor: Volume Migration Executor receive the information about volume detected not used by VSD, and the volume migration is done according to the steps: create new volume - duplicate content - remove old volume - update the metadata server. After all steps are complete, VME return the complete information to the system.

(5)Volume Usage Collector: Volume Usage Collector deploy in the form of daemons on each volume server. By monitoring the iSCSI port 3260, it collects each volume usage, update the volume information in the metadata server, for VVJM to analyse and calculate, so as to draw the best migration queue.

When a certain condition of the volume migration is satisfied, VMT can detect it and then be triggered, it will send a request of migration volume information to VVJM .VVJM obtain real-time volume information from the volume metadata server regularly, and using certain strategy constantly to update and improve the migration queue it maintains. After VVJM receives the request of get migration volume information, take out the first volume in the migration queue and send it to VSD to detect the volume usage now. If the volume is detected being used, it will send feedback to the VVJM and request to send migration volume information until one reasonable and not being used volume.VSD will send the information of volume which is detected to VME, and VME will perform the final migration work. The volume migration is done according to the steps: create new volume - duplicate content - remove old volume - update the metadata server. After all steps are complete, VME return the complete information to the system. During the operation process of Orthrus Plus system, the VUC will monitor the information about access to the volumes real-time, and the volume access information will be updated to the metadata server. Figure 6 intuitively describes the logical structure of the modules above and the migration process.

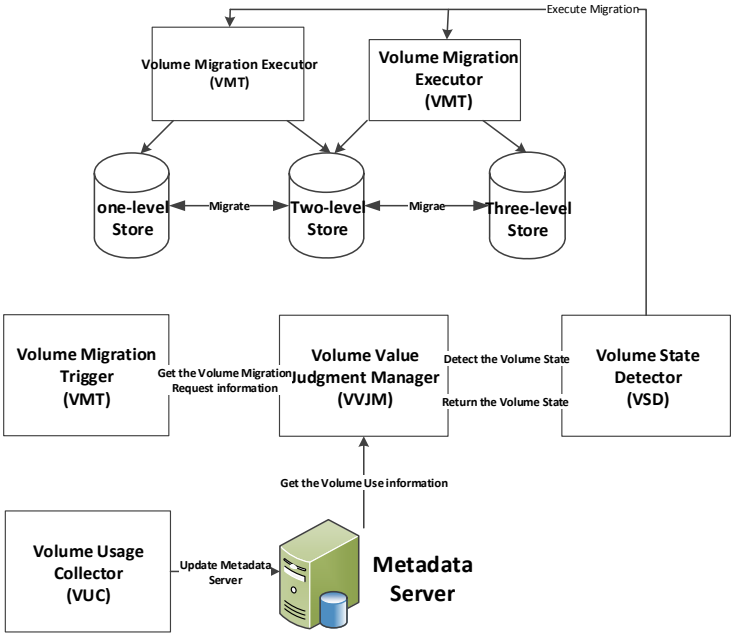

Figure 6. Logic Structure Diagram of Volume Migration Model

4.2.2. Design and Realization of Replacement and Migration Strategy based on Volume Life Circle: In hierarchical storage, the online storage device has high read-write speed and high performance, but capacity is limited, the price is expensive, and this is similar to the cache. Therefore, when designing volume replacement and migration strategy, common web cache replacement strategy can be used for reference. In general, common web cache replacement algorithm include LRU [20] algorithm, SIZE [21] algorithm, *etc*. LRU algorithm only consider the recent visit time of data, so the efficiency is not high; Data size is considered as a characteristic in SIZE algorithm, the max data will be the first to be replaced, so that more small data can be stored. But the algorithm may pollute cache, that is, although some data have already been expired but not been replaced. In order to avoid the above situation, this paper USES a multiple parameter GDSF algorithm (Greedy Dual - Size Frequency) [22].

GDSF algorithm replace the key factor ,minimum volume, each key factor Ki of volume can by calculated by the formula

$$Ki = Fi * Ci / Si + L \quad (12)$$

Among the formula, Fi indicates the use frequency of volume i; Ci indicates the cost to replace the volume i , we take the commonly used classical number 2 + size/536 [22] here; Si indicates the size of the volume i; L is inflation factor and the initial value is 0, when volume j is replaced, the inflation factor L of the rest volumes which have not been replaced are updated to Kj. The algorithm fully considers the use frequency, the volume replacement cost and the size of the volume. At the same time, the inflation factor completely overcome the pollution problem of the SIZE algorithm, makes the volume not to be used can be replaced out.

5. Orthrus Plus System Experimental Evaluation

5.1. Experimental Environment Description

This paper deploys the Orthrus Plus system in a real environment, and achieves the load balancing strategy based on GA. The experimental environment is as follows: a 10TB Inforcore Nextor SS5048 storage array, an Orthrus Plus Server node, two volume server nodes, two VMM nodes, an Orthrus Plus Client node and an internal 1GB Ethernet experimental environment. Table 2 shows the hardware and software configuration of each node. We carve out a 1TB space from the storage array as the storage pool of the system. Two volume servers shared manage the storage space by using CLVM, and provide VMs block device. There create a virtual machine in each VMM server. The configuration of VM is shown in Table 3.

Table 2. The Hardware and Software Configuration Table of Each Node

	Quantity	CPU	Memory	OS	Others
Orthrus Plus Server Node	1	Intel Core 2.98GHz CPU*4	2GB	CentOS 5.4 2.6.18-238	HSQLDB
Volume Server	2	Intel Xeon	2GB	CentOS 5.5	CLVM

Node		2.13GHz CPU*4		2.6.18-164	
VMM Node	2	Intel Core 2.98GHz CPU*4	2GB	CentOS 5.4 2.6.18-238	Xen
Orthrus Plus Client Node	1	Intel Core 2.98GHz CPU*4	2GB	CentOS 5.4 2.6.18-238	

Table 3. VM Configuration Table

CPU	Memory	Storage	OS
Intel Core 2.98GHz * 1	512M	4GB	CentOS 5.4

5.2. Experimental Comparison Before and After Improving the Dynamic Load Balancing Algorithm

This paper designs the following two sets of experiments to verify the dynamic load balancing strategy in Orthrus Plus has some improvement than in Orthrus in the accuracy and efficiency to search for the optimal solution.

At first, artificially limit the bandwidth of two volume servers which result in the significant performance difference. Second, modeling by performance weighted load model in Section 3.2, and getting the performance weighted load values of two volume servers. At the same time, the population size of the two algorithms is set to experience value 50. When the population doesn't produce better elite for 30 generations, it tacitly approves that the function has converged, and the evolution will terminate.

(1) Accuracy Experiments:This experiment takes the chromosome length (the total number of connections in two volume servers) as the variable which range from 10 to 100 by steps of 10. Each group of experiments will generate the initial chromosome and initial population under the defined length randomly. Each length will take 10000 experiments. We take the ultimate elite's fitness value as criterion to compare the searching accuracy of original and improved algorithm. The experiment result is shown in Figure 7.

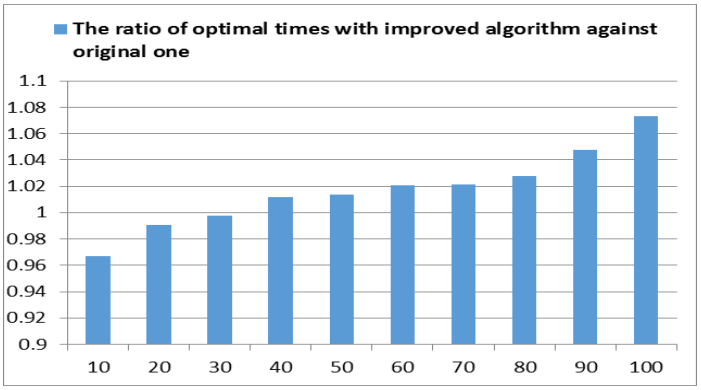

Figure 7. The Experiment Figure of Searching Accuracy of Original and Improved Algorithm

(2) Efficiency Experiments: This experiment also considers the chromosome length as a variable, and the range and steps are also the same. But this time we take the average iteration steps when get the same optimal solution in two algorithms as criterion to compare the efficiency of getting the same solution. The experiment result is shown in Figure 8.

The above two experiment result figures show that as the chromosome length is small at the beginning, the complexity of entire search is not high. The load balancing strategy based on mutated genetic algorithm which reduces the global leap search caused by crossover is more appropriate in the case of small amount of data. However, when the chromosome length becomes larger, the entire search process is more and more complex, the global search capability of load balancing strategy in Orthrus Plus system demonstrates its superiority in accuracy and efficiency.

5.3. Experimental Analysis of ISCSI Pause and Resume

Compared with a complete transmission process, the iSCSI pause-and-resume in Orthrus Plus system will add some additional overhead, such as the process of break iSCSI connections, the process of re-select the volume server by Orthrus Plus Server, the process of establish the iSCSI connections, the process of calculate the breakpoint, *etc*. We adopt the timing operation of various additional aspects to accurately quantify their time overhead, and compare with a complete transmission to reflect the effect to users.

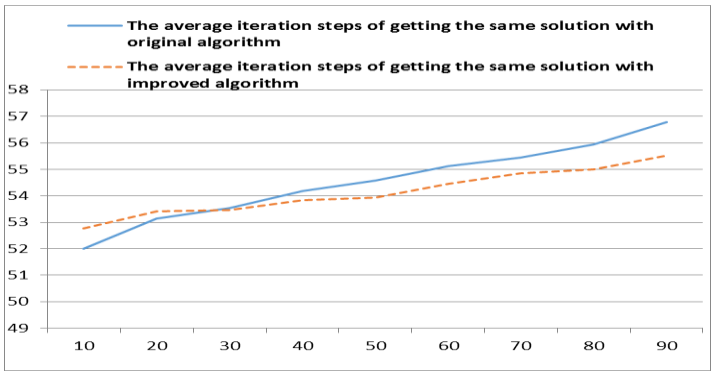

Figure 8. The Experiment Result Figure of Searching Efficiency of Original and Improved Algorithm

We design the following experimental scene: A virtual machine VM1 in VMM1 connects to a remote volume whose size is 5G through volume server A. The user of VM1 is uploading a 2G file to this volume. At this time, we artificially set the status of volume server A failure. When the volume server status listening module has detected the failure of volume server A, it will disconnect the connection between volume server A and VMM1 immediately, and search for another normal volume server B. Then it establishes the connection between volume server B and VMM1. At last, system calculates the breakpoint and continues to upload until the uploading is finished. At the same time, we set a group experiments without pause-and-resume process as a contrast.

Table 4. The Experimental Result of ISCSI Pause-and-Resume

	Break iSCSI connections	Re-select the volume server	Establish the iSCSI connections	Calculate the breakpoint	File transmission	Total time
Time Cost（ms）	158.7	1.3	1228.4	0.1	9986.8	11517.9

System calculates the total time of above process and the respective time of these additional overhead. We make several experiments using different files with the same size to reduce the effect by cache, and calculate the average time. The result is shown in Table 4.

From Table 4 and Figure 9, we can see the process of establishing the iSCSI connection costs a lot of time because this process contains the connection between VMM and volume server and attaching the volume to VM from VMM. From another side, although the proportion of time cost of breaking and establishing the iSCSI connection is relatively large, it only costs 1.5 seconds if we quantify the time. It should be in the acceptable range of users. Therefore, there has certain advantages in iSCSI pause-and-resume compared with the time cost of data retransmission.

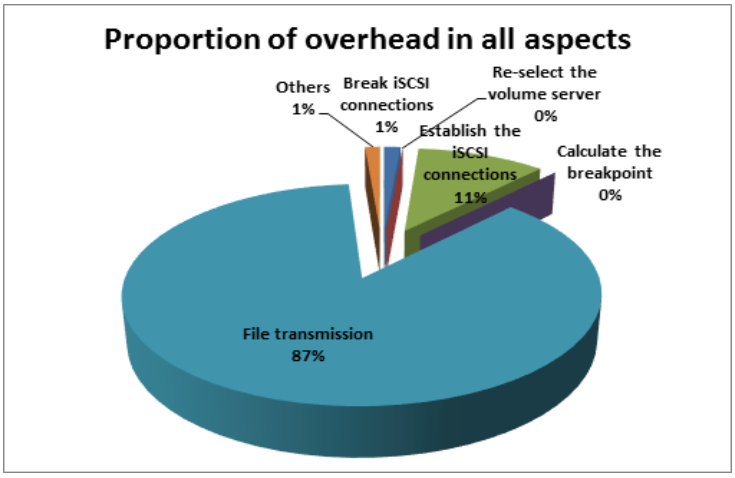

Figure 9. The Proportion Figure of Overhead in All Aspects in ISCSI Pause-and-Resume

5.4. Experimental Analysis of Tiered Storage Performance Based on Volume Life Cycle

This experiment takes two disk in the storage array as the back-end storage array, one of which used for online storage, another of which is artificially limited the network bandwidth as 1/10 of the original, to simulate the speed gap between SSD and HDD disk. At the same time, in advance, we respectively create 10 10G-size logical volumes in the two types of storage devices. Experiment will be performed as two groups, one of which will not use any strategies and another uses the tiered storage strategy, a random volume access sequence of equal length 100 will be used in each group.

Table 5. The Improvement of Logical Volume Throughput under Different Request Block Size

request block size	logical volume throughput without tiered storage（MB/s）	logical volume throughput with tiered storage（MB/s）	Percentage raising
512B	0.01522	0.0207	36.00%
1KB	0.030322	0.041222	35.95%
2KB	0.060855	0.0828	36.06%
4KB	0.120847	0.164456	36.09%
8KB	0.238151	0.324022	36.06%
16KB	0.466598	0.637762	36.68%
32KB	0.903897	1.231253	36.22%
64KB	1.679757	2.27906	35.68%
128KB	2.986225	4.076104	36.50%
256KB	4.384132	5.950705	35.73%
512KB	5.775538	7.892975	36.66%
1MB	7.214614	9.875994	36.89%

This paper uses the iometer [23] as the load produced for each logical volume in this experiment, and according to the random volume access sequence to test the IO throughput of logical volume under the load. The results are shown in Table 5 and Figure 10.

From the experiment data ,we can see that after using the tiered storage strategy, the logical volume IO throughput of the whole system has large ascension, between 35% and 37%. One important reason is about 10 times the speed between the two hard disks, and the volume replacement and migration strategy based on volume life circle enables the volume frequently accessed to reside for a long time in the faster hard disk, so that the overall IO throughput rate is improved.

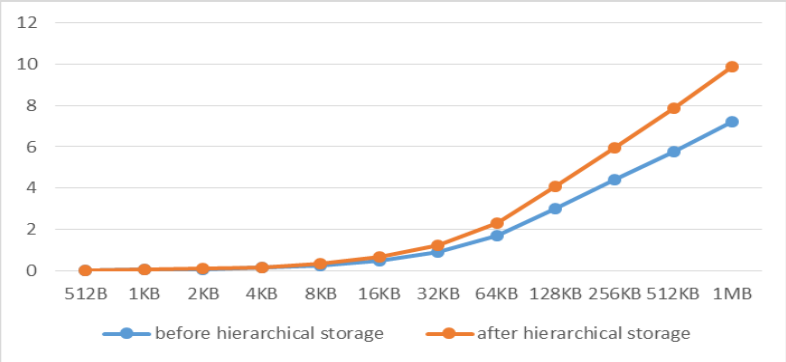

Figure 10. Logical Volume Throughput with Tiered Storage

6. Conclusions and Future Work

This paper proposes and achieves a multiple volume servers block cloud storage system based on load balancing—Orthrus Plus. Dynamic load balancing strategy based on genetic algorithm is used in the volume server to make the whole cluster load more stable. At the same time, we put forward the iSCSI pause-and-resume technology, to solve the problem about user data retransmission in the process of load balancing. he experiments show that Orthrus Plus system with dynamic load balancing strategy achieve higher efficiency and reliability in the case of large amount of data, and tiered storage strategy to a certain extent, also improves the IO throughput of the system.

In the future, we will consider the safety of the whole system: 1) the metadata server records all the system volumes, snapshot information, which is the key to the Orthrus Plus system. Distributed deployment of the metadata server can guarantee its reliability to a certain extent; 2) due to the tiered storage strategy, the users' data risk in the volume migration process. Therefore, snapshot backup for each logical volume established can be way to ensure the security of user data.

Acknowledgement

This paper is granted by National Natural Science Foundation of China under Grant(No. 61202094), Zhejiang Provincial Natural Science Foundation(No.LY13F020045, LY13F020047), China Postdoctoral Science Foundation(No. 2013M541780) and The National Key Technology R&D Program under Grant(No.2012BAH24B04).

References

[1] Hadoop, http://lucene.apache.org/hadoop/.
[2] S. Ghemawat, H. Gobioff, S. Leung, "The Google file system", Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing NY, USA, October, **(2003)**, pp.19-22
[3] Amazon S3, http://aws.amazon.com/s3/.
[4] Amazon EBS service, http://aws.amazon.com/ebs/.
[5] X. Gao, M. Lowe, Y. Ma, M. Pierce, "Supporting Cloud Computing with the Virtual Block Store System", Proceedings of e-Science, Oxford, UK, **(2009)**, pp.208-215.
[6] J. Wan, Y. Wang, J. Zhang, L. Zhou, "Orthrus: A Block-Level Virtualized Storage System with Multiple-Volume Servers", Advanced Science Letters, vol. 7,iss. 5, **(2012)**, pp. 68-72.
[7] W. Zeng, Y. Zhao, K. Ou, "Research on Cloud Storage Architecture and key technologies", IEEE International Conference on Intelligent Computing and Intelligent System, **(2009)**, pp. 24-26
[8] Internet Small Computer Systems Interface (iSCSI), http://tools.ietf.org/html/rfc3720.
[9] D. Roman, "Web Service Modeling Ontology", Applied Ontology, vol. 1, no. 1, **(2005),** pp. 77-106.

[10] CLVM Project Page, http://sources.redhat.com/cluster/clvm/.
[11] iSCSI Enterprise Target, http://iscsitarget.sourceforge.net/.
[12] D. Teigland, H. Mauelshagen, "Volume Managers in Linux", Proceedings ofthe FREENIX Track USENIX Annual Technical Conference **(2001)**,pp. 185-197.
[13] M. Rosenblum, "Tal Garfinkel, Virtual Machine Monitors: Current Technology and Future Trends", Computer, v.38 iss.5, May **(2005)**, pp. 39-47.
[14] T. Berczes, G. Guta, G. Kusper, W. Schreiner *et al.*, "Analyzing Web Server Performance Models with the Probabilistic Model Checker PRISM", Technical report no. 08-17 in RISC Report Series, University of Linz, Austria. November **(2008)**.
[15] J. Hu, Y. Sun, Q. Xu, "The theory and application of Genetic Algorithm", Proceedings of Computer and Communication Technologies in Agriculture Engineering, vol, 1, **(2010),** pp.155 - 157.
[16] K. A. de Jong, "An analysis of the behavior of a class of genetic adaptive systems", University of Michigan, USA, **(1975)**.
[17] B. Li, J. Shu, W. Zheng, "SCSI Target Simulator Based on FC and IP Protocols in TH-MSNS", Proceedings of Tsinghua Science and Technology, vol. 10, **(2006)**, pp.89 – 93.
[18] G. Shah, K. Voruganti, P. Shivam, "Ace: Classification for information lifecycle management", Proceedings of NASA Mass Storage Systems and Technologies, **(2006)**.
[19] M. He, L. Xing, G. Li, "A Data Migration Strategy for HSM Based on Data Value", Proceedings of Journal of Information & Computational Science, **(2011)**.
[20] D. L. Willick, D. L. Eager, R. B. Bunt, "Disk Cache Replacement Policies for Network Fileservers", Proceedings of the 13th International Conference on Distributed Computing Systems, Pittsburgh, 1993.
[21] S. Williams, M. Abrams, C. Standridge, G. Abdulla, E. Fox, "Removal Policies in Network Caches for World Wide Web Documents", Proceedings of ACM SIGCOMM, Stanford, CA, **(1996)**.
[22] L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy", Technical Report HPL-98-69R1, Hewlett-Packard Laboratories, **(1998).**
[23] Iometer, http://www.iometer.org/.

Authors

Jian Wan, he received the PhD degree in Computer Application Technology from Zhejiang University, Zhejiang, China, in 1989. He is currently a professor in software engineering in Hangzhou Dianzi University, China. His research interests include Grid Computing, Service Computing and Cloud Computing.

Xun Chen, he received the Bachelor Degree of Communication Engineering in Central China Normal University, Hubei, China, in 2011. He is now studing the Master of computer software and theory in Hangzhou Dianzi University, China. His research interest is cloud computing and cloud storage system.

Xindong You, she is a lecturer of School of Computer Science and Technology, Hangzhou Dianzi University, China. She is with the Grid and Service Computing Lab in Hangzhou Dianzi University. Before joining Hangzhou Dianzi University, she was a PhD candidate in Northeastern University from 2002 to 2007. She received her PhD degree in 2007. Her current research areas include virtualization, distributed computing, etc.