2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.128

Network Sharing Mechanism of Disk Images: A Multicast-based Research and Implementation

Zhou Xiangyang[1,+], Jiang Xiangtao[2], Hou Yi[3]

[1,2,3]School of Computer Science and Engineering, Beihang University , Beijing, China

Abstract. The existing disk image sharing mechanism adopts the unicast communication, which results in wasting network bandwidth and increasing the server load. Meanwhile, the unicast-based sharing mechanism is vulnerable to the bottleneck of server performance and the limitation of network bandwidth. In order to address the problem above, we present a network sharing mechanism in this paper, which is based on multicast, to guarantee its key role in the lifetime of virtual cluster. By means of the multicast-based mechanism, we implement a FUSE File System, called mUFS. Compared to previous works, our system achieves the following goals: (1). Saving network bandwidth effectively; (2). Reducing the load of the server; (3). Increasing the hit rate of the data required by virtual machines.

Keywords: Virtual machine, Disk images, Share, Multicast

1. Introduction

Virtual machine technology transforms physical resour-ces into logically-manageable resources, which breaks the barriers of physical structure and brings about the new computing mode that can shield the distributives, hetero-geneity, dynamicity as well as other features of physical resources. It reduces the coupling of hardware and software so that the software runs more efficiently, stably and reliably. With features such as isolation and encapsulation, virtual machine technology gives birth to many new applications, one of which is the virtual cluster. Based on the physical resource pool as a runtime support environment, virtual cluster refers to providing users with the virtual computing environment by using virtual machine technology according to the users' demand.

The entire life cycle of the virtual cluster depends on the virtual machine disk image files. Virtual machine disk image is image files that encapsulate the virtual machine operating system and necessary executive software. Usually they are very large and the local access will occupy a lot of storage resources. The common method is to manage and share the image files through the network storage.

2. Related Work

Storage virtualization is an important requirement for virtual cluster. Based on the demands for migration and flexible configuration of virtual machine, the storage management of disk images mostly adopts network-based storage solutions. In response to this problem, one of the simplest solutions is image cloning: transferring an image to the head nodes only once at the start and configure the virtual cluster, to transfer an image to a working node once and after the virtual machines as working nodes meet the demand of the deployment, instantiate the virtual machines dynamically through image cloning to provide virtual cluster computing environment based on distributed grid resources. This could be seen in VMPlants system [3]. Another way is to use NFS [4] (Network File System). NFS allows a system on the network to share directories and files with others. By using NFS, users and programs can transparently access files on remote system. In this mode, the image files are stored in a single unified server and virtual machine

[+] Zhou Xiangyang
zhouxymail@gmail.com

host will use less storage space. Also, in a way of exploring directories, NFS maps the image file directories to the remote client machines using RPC to communicate between the client and the server and follow the on-demand requests to transmit image data.

3. The problem of sharing disk Image in virtual cluster

3.1. Preliminary

The similarity of virtual node will be significant when virtual nodes use the same virtual cluster. Therefore, the high probability of concurrently requesting the same data provides the basis for the multicast during the deployment, start and run of virtual nodes in the cluster

Image cloning and NFS are both used a point to point unicast communication. In this process, the server has to maintain multiple connections, and thus the burden of data transfer becomes more seriously. In addition, a series of duplicate connection transferring virtual machine image files wastes the bandwidth of transmission network. Consequently, the efficiency and performance of disk image sharing should be influenced sharply.

3.2. Transparency

The resources sharing requires shielding the underlying hardware to user that allows users to use these resources like local storage ones. For this reason, transparency plays a crucial role in measuring the quality of the disk image sharing system.

Transparency requires: 1) Storage place should be shielded to the application which means the application need not concern the data is stored in remote or local. 2) Applications should take the same operation no matter remote or local data. 3) When multiple applications operate on the same file, it would not cause data inconsistency. 4) When the nodes of virtual machine operate files remotely, the failure of computer or network would not lead to file inconsistencies. Based on the above points, in the implement of virtual machine disk images sharing system, we should take an important consideration on transparency to ensure the efficient.

3.3. Execution time

If there is a virtual cluster with n nodes, T denotes the overall run time consuming; Ts denotes the start time of deployment of virtual cluster; T1, T2 ... Tn denote respectively the finished time of starting virtual cluster node 1, node 2 ... n of the virtual machine. The deployment and start time of a virtual cluster T should be the maximum of the difference between T1, T2 ... Tn and Ts. Thus, a few virtual machines that have a shorter execution time in a virtual cluster cannot indicate the performance of entire virtual cluster. Therefore, in the process of deployment and start of the virtual cluster, we should minimize the execution time of the slowest virtual machine.

4. System flow

As mentioned above, figure 1 illustrates the interaction of data in the process of sharing.

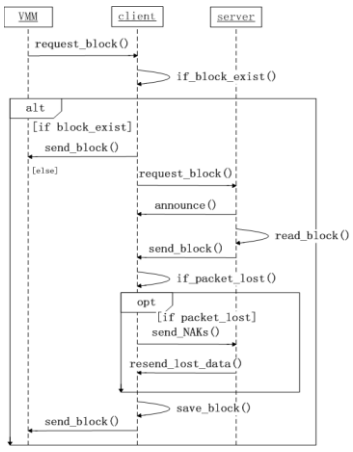

Figure 1. The exchange of data

(1)After the application of client sends out a data request, the client checks whether there has the data locally. (2)If the local data miss, the client sends out data request to the remote server (Step 3), or else goes to the Step 6. (3)When the server listens to the client's request, the server establish the multicast queue through the “announce”. (4)According to the multicast queue, the server uses the multicast controller to transfer data to the corresponding node by multicast. (5)After the client receives data, the client uses local cache to save the data as backup. (6)The client application directly reads the required data from the cache.

5. mUFS

5.1. Structure

The mUFS is a kind of user space file system based on multicast that sustains the disk image sharing of virtual machine, data request and transmission, and it adopts the same C/S structure with NFS. To make the system more flexibility, the system is designed into modularity and hierarchy. The system structure shows in Figure 2.

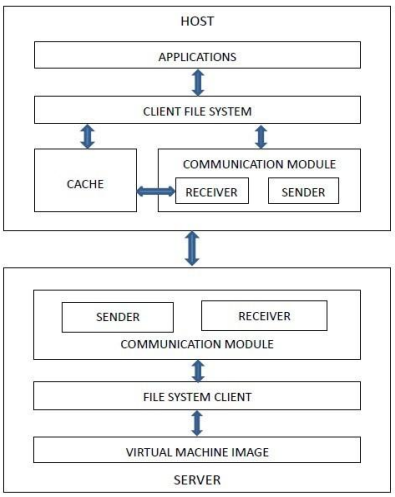

Figure 2. Structure of mUFS

mUFS mainly contains three modules: the file system, communication and cache. The client captures the request of application for file operations, and then informs the server through the communication module. The server transfer the data obtained by reading the disk images to the client via the multicast communication. The main functions of each module are as follows.

File System: The client is responsible for processing the various requests of applications for file operation. When applications need to read the data, the client query cache and send out requests to the remote server. The server is responsible for analyzing the data request from host computer, and then according to the requests the server gets data from image library and generates the data response.

Communication: The client part is responsible for sending out varied data requests to the server and receives the server`s response, and then pass the data directly to the file system or cache. The server is responsible for receiving data requests to send to file system for resolution, and sends the response generated by the file system to all host computers through multicast.

Cache: This module is responsible for the storage and management of data in image used to send to the client by multicast, and provide data to the file system when required.

5.2. File system

Based on the description above, we adopted a relatively simple FUSE framework. FUSE provides a set of API to interact with VFS (Virtual File System). In this system, each operation function consists of two parts that one lies in the client and the other lies in the server. The two parts along with communication module cooperate to complete file operations, and the working mechanism shows in Figure 3.

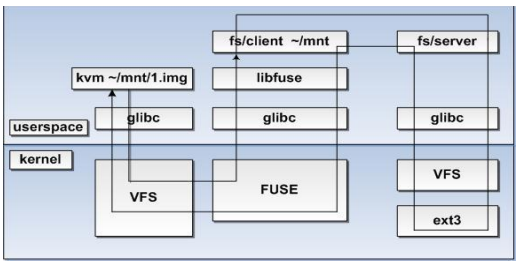

Figure 3. The mechanism of file system

When file operations are requested by the applications, the related interface and function of VFS is called to FUSE. FUSE will map file operations to a function of the file system of client, and then the client informs the server of file system through network. The server processes files and data and responses to the client; the client will pass the data to VFS through FUSE, and VFS will transfer them to the application.

5.3. Communication

Communication is the most important part of the network file system, which is responsible for the communication between the client and server and the client actually get remote file data through this module.

There are two kinds of communication package between the client and server:

The request packet from client to server: This kind packet includes relatively less information such as the file operation type and the information of data blocks.

The response packet from server to client: This respond packet to client can include file features that contains less information, or include file data that is relatively large.

There are two alternative communication ways between the client and server:

TCP and UDP based on multicast. The benefit of using TCP is reliable jet only used to point to point transmission that may results in reduplication, while multicast transmission can reduce reduplication but with poor reliability that need additional cost to ensure the reliability. The compromise choice is to take TCP packets to transfer small packet and multicast transmission to transfer large packet.

The maximum data of FUSE in a single request is 128K, while the UDP is 16K. Obviously, a single UDP packet cannot complete the transfer of large files data, so that the data should be numbered by segment. To ensure the reliability of multicast, besides the segment number of packets, the server informs the number of segments to client before transmission and send confirmation message to notify the client that the data has been transferred completely. According to mentioned above, the format of packet is designed as follow.

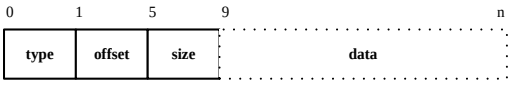

Figure 4. the TCP packet format

TCP packet is a kind of small one, and most of the requests and response packets are TCP packet. The meaning of each segment is shown in Table 1.

TABLE.I The meaning of TCP package segment

Segment	bits	data type	Meaning
Type	1	Char	package type
offset	4	Int	location of data block
Size	4	Int	size of data block
Data	0-128K	char	optional segment, data transferred

Multicast packet is a kind of large one, which is only used when the server send out the file data to the client. The basic information of this kind packet involves the status of multicast, the number of packet or the total of data packets. Moreover, the packet will also include the contents of file data in transfer phase should also. The design of packet is shown in Figure 5, and Table 2 shows the meaning of each segment.

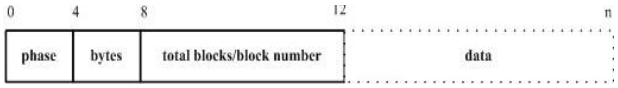

Figure 5. The Multicast packet format

TABLE.II The meaning of UDP package segment

segment	bit	data type	value	Meaning
phase	4	int	1	Announcement
			2	Transmission
			3	transmission completion
bytes	4	int	≦ 1.4K	total bytes in announcement; byte number of each time in transmission
tb/bn	4	Int	Small integer	total segments in announcement; sequence number of each time in transmission
data	0-1400	char		data transferred

Because of the asymmetry of communication between server and client, the request from the client to the server and the response from server to the client contain different information. As the system's modular method, we implement a specific module to process packets. The main task of this module is producing packet, converting the packets data stored in the structure of memory to sequence data of network and then to the structure of host for storing, and providing a common transmission receiver function.

The server in communication module uses TCP socket to receive the packet of request from client and send response partly, and the server uses multicast socket to send data to a specific multicast group.

The client in communication module uses TCP socket to send request and receive the response packet partly. After the establishment of multicast socket, *setsockopt ()* should be called to configure the multicast socket options for the purpose of joining a particular multicast group to receive multicast data. Since receiving the multicast data is passive, it is necessary to establish a separate process to monitor multicast group and receive multicast data.

5.4. Cache

Using multicast can avoid server sending the same data repeatedly. The client will receive multicast data of image file from server, even though these data may not be needed by application currently. However, because of the highly similar function of virtual cluster node, these data requested by other nodes may be needed in this node. Based on the above considerations, the system must have the following functions:

1) The client caches the multicast receiving data;

2) When the application needs data, it queries multicast data in cache firstly. If the data exists, the system directly sends these data to the application instead of requesting to the server;

3) When the cache buffer exceeds the default value, the system replace them based on the LRU(least recently used) strategy.

In addition, because the needs of application to the file data is not completely continuous, it is probably that the requested data may be located continuously in the image file in a short period and but the requested data in different periods is not always continuous. Therefore, a mechanism to save the non-contiguous data blocks is needed, which can also merge adjacent and contiguous blocks of data in order to reduce the search time. Since the size of each data block is not fixed and the total of data blocks is unknown, the system uses the data structure, link-list, to save all the data blocks.

For each node of the list, the system saves the location of data block in the image file, the data block size, the contents of data block, the time that the data block gets into the cache, and the point to the next node.

The nodes are stored in accordance with the location of data block stored in the image, which is available to search sequentially. When the system receives data, the data is inserted as follow.

1) Query the current cache list to determine whether the data blocks already exist. If the data does not exist, go to the second step, or cancel the insert operation.

2) Scan cache list sequentially and determine the location of data blocks to be inserted.

3) Insert a new node.

4) Determine whether the new node is contiguous with the adjacent nodes. If so, merge these nodes to the new one.

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper.

6. Experimental Results and Discussion

In order to verify the performance of mUFS system in a virtual cluster, we design two sets of experiments used to measure the hit rate of data in the start of multiple virtual machines and the bandwidth occupancy. In the experiments, we use four desktop computers as physical machines, and each physical machine has installed KVM (Kernel-based Virtual Machine), FUSE and other relative software. The configuration is shown in Table 3.

TABLE.III The configuration of clients

client	OS	CPU	memory
Client01	Debian Lenny 5.0	Intel(R) Pentium(R) 4 CPU@3.00GHz	1GB
Client02	Debian Lenny 5.0	AMD Sempron(tm) 3000+@ 2.00GHz	1GB
Server	Debian Lenny 5.0	Intel(R) Core(TM)2 Duo CPU E6550@ 2.33GHz	2GB
Client04	Ubuntu 9.04	Intel(R) Core(TM)2 Duo CPU 6300 @ 1.86GHz	1GB

In the experiment, we take Server as a test server of disk image, and Client01, Client02, Client04 as a client. The Client04 and Server support hardware virtualization technology. The network bandwidth is 100M.

6.1. Cache hit ratio

This experiment is to test the hit rate that the client directly accesses data memory. We start four virtual machines at the same time, and deploy a virtual node on client01, client02 respectively, and two on client04.

The four virtual machines all use the same virtual machine disk image with the size of 1GB. The destination of multicast address is 239.5.5.0. The server sends data to the multicast address and the client access data from the multicast address. The result shows that the four virtual machines can start normally, and the start-up time and hit rate of virtual machine on the client are showed in Table 4.

The sharing method of unicast network requires the request to the remote server, so that the application needs to wait until the response reached, which may lead to decrease of the hit rate of local data. As shown in Table 4, there is a great probability that the client need not to request data to the server because of the use of multicast to receive data and save in cache. By doing this, the system can save network bandwidth, while the virtual machine will not significantly slow the speed of startup. In addition, the hit rate of client04 in the table is low. This is due to that the client04 supports hardware virtualization and virtual machine starts faster than other clients. Most of the first data request is executed by the client04, which is also the main reason that client01 and client02 have a higher memory hit rate.

TABLE.IV The start-up time and hit rate of virtual machine on the client

client	startup time	read times	Memory hit times	hit rate
Client01	130.41s	558	464	83.2%
Client02	171.74s	564	472	83.7%
Client04	19.78s	581	34	5.9%

6.2. System performance

This experiment compares the consumption of network bandwidth and server load between NFS and mUFS during the start of four virtual machines simultaneously.

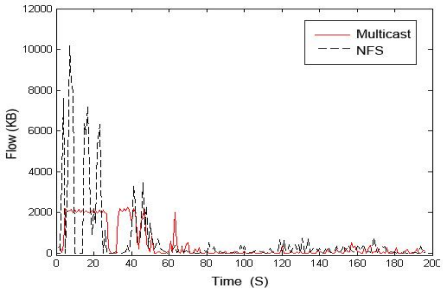

Figure 6. Comparison on the consumption of network bandwidth and server load between NFS and mUFS

TABLE.V Comparison on read and write times between NFS and mUFS

VMs startup mode	Read and write times
NFS	877
mUFS	83

As the bandwidth consumption described in Figure 6, in the early stage of start, the client needs to read the boot program, the kernel modules and other relative data from the server, which occupy more bandwidth. When system gets to initialization phase, it greatly reduces bandwidth. From the comparison, mUFS occupies much lower bandwidth than the NFS.

As described in Table 5, when the system starts multiple virtual machines using the same disk image, NFS has far more read and write disk image times than mUFS. The multicast technology greatly reduce the

condition that the client repetitively requests for the same data, and in most cases the client can directly hit the data from the local memory, which reduces the times of getting data from remote server and server load.

7. Conclusion

This paper focuses on the demerits of existing network sharing of virtual machine, and on the basis of analysis of relevant existing technologies and systems we investigate relative problems and design a disk-based multicast image sharing system. This system can store network disk image as request and save the data storage space. Meanwhile, the caching data improve the hit rate that the virtual machine of virtual cluster obtains the required data in memory. This also reduces the requests to the remote server, network bandwidth and server load. However, the current mUFS system still has possibilities to improve, and we will optimize the system performance and investigate the adaptive management of multicast queue, automatic deployment of system, and the multicast in Wide Area Network environment.

8. References

[1] W Emeneker, D Stanzione. Efficient Virtual Machine Caching in Dynamic Virtual Clusters. FHPC Initiative - SRMPDS Workshop, ICAPDS 2007 Conference, 2007.

[2] Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J.Figueiredo. Vmplants: Providing and managing virtual machine execution environments for grid computing. In Proceedings of the 2004 ACM/IEEE conference on Supercomputing,pages 7–18, Pittsburgh, PA, November 2004.

[3] Kozuch M., Satyanarayanan M. Internet Suspend/Resume[A]. Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications[C]. IEEE Computer Society, 2002: 40.

[4] X. Jiang and D. Xu, Violin: Virtual internetworking on overlay infrastructure. In Proc. International Symposium on Parallel and Distributed Processing and Applications, pp. 937--946, 2004.

[5] Sundararaj A.I., Dinda P.A. Towards virtual networks for virtual machine grid computing[A]. Proceedings of the 3rd conference on Virtual Machine Research And Technology Symposium - Volume 3[C]. San Jose, California: USENIX Association,2004: 14-14.

[6] Nakada, H., Yokoi, T., Ebara, T., Tanimura, Y., Ogawa, H., Sekiguchi, S. The design and implementation of a virtual cluster management system. In: Proc. of 1st IEEE/IFIP International Workshop on End-to-end Virtualization and Grid Management, pp. 61–71 (2007).

[7] Huai JP, Li Q, Hu CM. Research and design on hypervisor based virtual computing environment. Journal of Software, 2007, V18(8):2016-2026.

[8] W Emeneker, D Stanzione. Efficient Virtual Machine Caching in Dynamic Virtual Clusters. FHPC Initiative - SRMPDS Workshop, ICAPDS 2007 Conference, 2007.

[9] Hideo Nishimura , Naoya Maruyama , Satoshi Matsuoka, Virtual Clusters on the Fly - Fast, Scalable, and Flexible Installation. Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, p.549-556, May 14-17, 2007.