Object-based Multimedia Contents Storage for Mobile Devices*

Young Jin Nam, Min-Seok Choi, In-Gil Nam
School of Computer and Information Technology
Daegu University, South Korea
yjnam@daegu.ac.kr

Abstract - Mobile devices, such as PDAs, portable multimedia players, are more likely to encompass large storage devices with prevalance of high-quality multimedia contents. This paper proposes an object-based multimedia contents storage architecture that employs the object-based storage device model and the iSCSI protocol. It also provides a multimedia content player that operates directly with the proposed storage architecture. We implement both the proposed storage architecture and the multimedia content player upon the Linux environment. Performance evaluation by playing MP3 multimedia contents reveals that the proposed storage architecture reduces the total power consumption by 9%, compared with an existing networked storage. This enhancement is mainly contributed to the fact that a large portion of the file system is moved into the object-based multimedia contents storage from the mobile device.

Keywords: Object-based IP storage, multimedia contents, mobile devices, power consumption

1 Introduction

Technology advance in processors, codec, wireless network, I/O peripherals, enables high-quality multimedia contents to play on mobile devices, such as PDAs, portable multimedia players, etc. The mobile devices are mostly equipped with a hard disk or flash memory to contain the multimedia contents. Of these, the hard disk has currently become more commonplace, compared with the flash memory, because of its cost-effectiveness. However, the hard disk consumes more power, as it spins a spindle motor at a high speed and moves read/write heads mechanically.

The increase in Internet access speed from the mobile devices enables other alternatives to store multimedia contents: the network-attached storage(NAS) and the IP-based storage [1]. They can obviate any limits in storage capacity. However, the mobile devices require to have additional software components, such as NFS, CIFS, and a protocol stack, called iSCSI [1,2] to convey the SCSI protocol over IP network. Note that they operate upon the TCP/IP protocol stack. The NAS systems are similar to dedicated NFS or CIFS servers. They provide storage applications with file-level storage services. However, since the NAS server is involved in each file service between the storage applications and the underlying storage device, it suffers from scalability. This, is, as the storage capacity grows, the overhead of the NAS server increases proportionally. By contrast, the IP storage provides storage applications with block-level storage services over the IP-based storage area network(SAN). Typically, it is known that the IP storage guarantees good scalability in terms of storage performance and capacity. However, it does not provide enough interfaces to upper-level storage applications. In result, either storage applications should take care of a part of file-level services, for example, file locking, or they should require a special file system called a SAN file system. Note that the IP storage operates over IP-based SAN, as shown in Figure 1, by using the SCSI protocol, the iSCSI protocol, and the TCP/IP protocol.

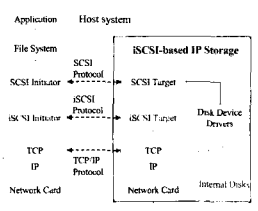

Figure1. iSCSI-based IP storage

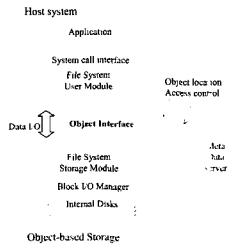

Figure2. Object-based storage

Recently, the object-based storage device(OSD) that exploits both advantages of the NAS server and the IP storage has been proposed as an emerging storage technology [3,4]. The OSD treats user data or files as objects with attributes. For example, a multimedia content like a MP3 file, a database table, or a data block can be mapped onto a single object. Much research efforts on OSD have been underway in many universities and research organizations to design more intelligent OSD

* This research was in part supported by POSTECH CMEST IT Research Center and in part by NURI PoP-iT Research Fund.

architectures [3], OSD-supporting file systems [5,6], QoS-guaranteed OSD [7], etc. Meanwhile, T10/SCSI ratified OSD command as a new SCSI command set in 2004 [8].

This paper designs object-based multimedia contents storage based on the object-based storage device(OSD) architecture. It also devises a multimedia content player operating directly with the proposed object-based multimedia content storage. Note that both the multimedia contents player and the OSD communicate with each other using the OSD SCSI commands and the iSCSI protocol. The remainder of this paper is organized as follows. Section 2 describes the proposed storage architecture. Section 3 provides the results of performance evaluation by running MP3 multimedia contents. Finally, this paper concludes with Section 4.

2 The Proposed Storage Architecture

Figure 3 shows the I/O environment of the proposed storage architecture. The main software components of the object-based multimedia contents storage (also called object-based IP storage) include an extended target-mode iSCSI driver that contains an embedded file system, and disk device drivers to handle the internal disks. The extended target-mode iSCSI driver (briefly, the target-mode iSCSI driver) basically works upon the TCP/IP stack and processes OSD SCSI commands. The embedded file system exists to serve a set of OSD SCSI commands that require file-level services, such as object creation and object open.

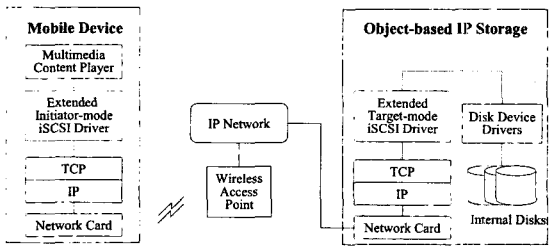

Figure 3. I/O einvironment of the proposed storage architecture

The major software components of the mobile device encompass an extended initiator-mode iSCSI driver and the multimedia content player. The extended initiator-mode iSCSI driver (briefly, the initiator-mode iSCSI driver) sends SCSI commands to its corresponding target-mode iSCSI driver over the IP network. The multimedia content player is in charge of reading a multimedia content file from the object-based IP storage and then playing the file at the mobile device. In what follows, we will describe specific architectures of the key components that are the initiator-mode iSCSI driver, the target-mode iSCSI driver, and the multimedia content player.

2.1 Initiator-mode iSCSI Driver

The initiator-mode iSCSI driver exists as a form of a kernel device driver within the mobile device. This driver can process a set of OSD SCSI commands, as well as the existing SCSI command set. Figure 4 shows a structure of a typical OSD command descriptor block (CDB).

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE(7Fh)							
1	CONTROL							
2-5	RESERVED							
6	SECURITY							
7	ADDITIONAL CDB LENGTH(N-7)							
8-9	SERVICE ACTION							
10	OPTION BYTE 1							
11	OPTION BYTE 2							
12-15	GROUP ID							
16-23	USER ID							
24-27	SESSION ID							
28-35	LENGTH							
36-43	OFFSET							
44-47	GET_ATTRIBUTES_PAGE							
48-51	GET_LIST_LENGTH							
52-55	GET_ALLOCATION_LENGTH							
72-75	SET_LIST_LENGTH							

Figure 4. Structure of a typical OSD command descriptor block

The OPERATION CODE(7Fh) distinguishes the OSD CDBs from the existing SCSI CDBs, and the SERVICE ACTION field represents a specific OSD SCSI command. The GROUP ID, the USER ID, and the SESSION ID respectively identify specific group, user, and session to conduct the given SERVICE ACTION. The LENGTH and the OFFSET represent the length and the offset of data wihtin an object to read and write. The remaining fileds are related to the return values for the associate command. The initiator-mode iSCSI driver supports the following SERVICE ACTION for its upper-level application, the multimedia content player: CREATE GROUP, CREATE, WRITE, READ, REMOVE, and REMOVE GROUP. The specific structures of those commands can be found in the OSD standard [8].

2.2 Target-mode iSCSI Driver

The target-mode iSCSI driver operates in the object-based IP storage. Similar to the initiator-mode iSCSI driver, it can handle a set of OSD SCSI commands, as well as the existing SCSI command set. This driver first receives the set of OSD commands sent from its associated initiator-mode iSCSI driver. Next, it decodes the received OSD SCSI commands and processes it according to its definition in the standard [8]. Table 1 summarizes how to process each of the received OSD SCSI commands that include CREATE GROUP, CREATE, WRITE, READ, REMOVE, REMOVE

GROUP. Managing objects requires an embedded file system within the object-based IP storage. Thus, our implementation employs a simple file system that works at the user level, where each object is treated as a file.

Table 1. Summary of processing the received OSD commands

Command	Behavior Description
CREATE GROUP	Generate a unique GROUP ID, create a group object of the GROUP ID, and return the GROUP ID
CREATE	Generate a unique USER ID within a given GROUP ID, create a user object of the USER ID, and return the USER ID
WRITE	Find the user object associated to the given GROUP ID and the USER ID, and write data received from the initiator-mode iSCSI driver into the position pointed by the given offset
READ	Find the user object associated to the given GROUP ID and the USER ID, read into a buffer data at the position pointed by the given offset, and transfer the buffered data to the initiator-mode iSCSI driver
REMOVE	Find the user object associated to the given GROUP ID and the USER ID, and remove it
REMOTE GROUP	Find the group object associated to the given GROUP ID, and remove it

2.3 Multimedia Content Player

The multimedia content player is mainly responsible for reading a multimedia content file from the object-based IP storage and playing it at the mobile device. The current version of the multimedia content player supports only MP3 audio files. In addition, we provide a couple of special commands to write/remove a MP3 file into/from the object-based IP storage.

We will start by explaining the special commands to write/remove a MP3 file. The write command not only stores a MP3 file into the object-based IP storage, but also creates a 24-byte file called a list file, as shown in Figure 5. The list file consists of a 4-byte magic number, 4-byte GROUP ID, 8-byte USER ID, and 8-byte file size information. The magic number informs the player that the MP3 file is stored at the object-based IP storage, not at the local file system. As previously described, a pair of the GROUP ID and the USER ID identifies the object associated to the MP3 file. Writing the MP3 file employs the OSD commands of OSD_CREAT_GROUP and OSD_WRITE. Note that, while the obtained GROUP ID by issuing the OSD_CREAT_GROUP should be stored in the metadata server in the OSD architecture, our implementation co-locates the metadata server with the storage client(mobile devices). In addition, all the GROUP IDs are stored in the "gid.dat" file. The remove command deletes the object within the object-based IP storage associated to the MP3 file by using the OSD commands of OSD_REMOVE command. Figure 5 and 6 depict how to write and remove a multimedia content file that is mapped into an object within the storage. Note that the root object is created at the initialization time for the object-based IP storage. Subsequently, all the other user objects become descendants of this root object.

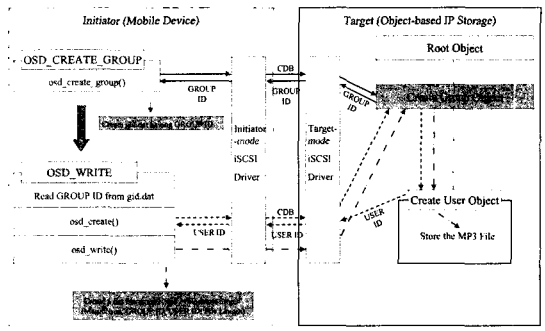

Figure 5. Writing a multimedia content (MP3) file into the object-based IP storage

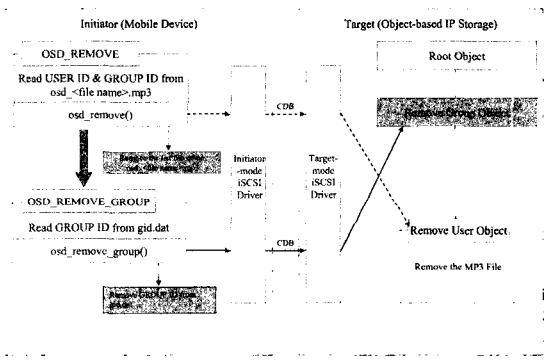

Figure 6. Removing a multimedia content (MP3) file from the object-based IP storage

The implementation of the multimedia content player revised a well-known command-line MP3 player in Linux, mpg321 [9], as shown in Figure 7.

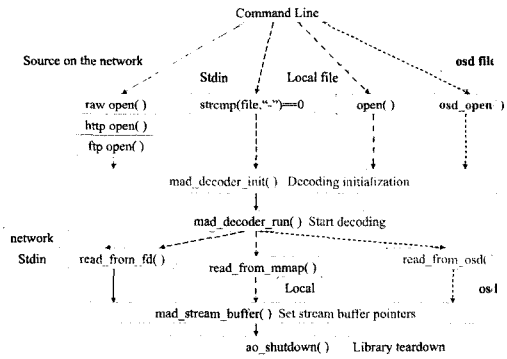

Figure 7. Architecture of the multimedia content (MP3) player

The multimedia content player can communicate directly with the underlying object-based IP storage via OSD SCSI commands. It plays MP3 files either at the local file system or at the object-based IP storage. We added the

osd_open function to open the MP3 file stored at the object-based IP storage when the magic number is detected from the MP3 file. In addition, we added the read_from_osd() function to read a stream of the MP3 file from the object-based IP storage via OSD SCSI commands.

3 Performance Evaluation

We compare the performance of the proposed storage (the object-based IP storage) architecture with the NFS network file system in terms of power consumption by running a MP3 multimedia content file. The consumed power is measured in an indirect manner by observing the CPU utilization. Our current implementation is fully operating upon the Linux environment on wired IP network, as shown in Figure 8. Currently, we have been porting our implementation onto a mobile/wireless environment that consists of PDAs and wireless network.

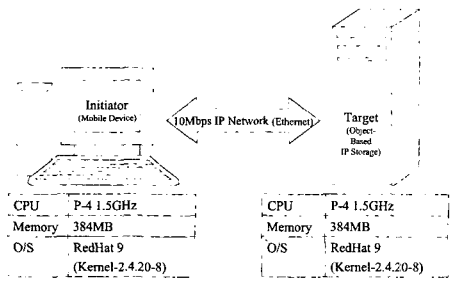

Figure 8. H/W and S/W environment of our current implementation

For performance evaluation, the same MP3 file is initially stored into the NFS file server and the object-based IP storage. Next, we run the MP3 file at each system, while measuring the CPU utilization, the memory utilization, and the kernel time. We repeated this experiment for each system more than ten times and averaged out the obtained results. Note that the kernel time represents the time spent in running in the kernel mode to play the MP3 file. As shown in Table 2, the proposed storage architecture improves an amount of power consumption by 9%, compared with the NFS file server. This gain is mainly attributed to the fact that a large portion of the file system source code is moved into the object-based IP storage itself. Observe that the kernel time of the proposed storage architecture is only half of the NFS file server.

Table 2. Summary of the obtained performance results

	NFS File Server	The proposed storage architecture
CPU utilization	4.7%	4.3%
Memory utilization	1.98%	2.1%
Kernel time	0.2sec.	0.1sec

To summarize, by applying the proposed storage architecture to a mobile environment, we can expect a significant amount of power saving by reducing the CPU utilization within the mobile device.

4 Concluding Remarks

We, in this paper, designed object-based multimedia contents storage (also described as object-based IP storage) and its associated multimedia content player for mobile devices. We also verified its effectiveness by implementing prototypes under desktop environments. The results of the performance evaluation revealed that the proposed storage architecture is very relevant to the mobile devices equipped with limited battery. Currently, we have been changing the desktop environment to the mobile/wireless environment, and also upgrading the Linux kernel version to 2.6. In addition, we plan to expand the functionality of the current media content player to run MPEG4-encoded video files.

References

[1] T. Clark, IP SANs: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area networks. Addition-Wesley, 2002.

[2] K. Meth and J. Satran, "Design of the iSCSI protocol," Proceedings of the Mass Storage Systems and Technologies/20th IEEE/11th NASA Goddard Conference, April 2003.

[3] M. Factor, et al., "Object storage: The future building block for storage systems," Proceedings of the 2nd International IEEE Symposium on Mass Storage Systems and Technologies, June 2005.

[4] Erik Riedel(Seagate Research), Object-based storage device(OSD) basics: http://www.snia.org/education/tutorials/spr2005/storage, 2005.

[5] http://www.lustre.org.

[6] F. Wang, et al., "OBFS: A file system for object-based storage devices," Proceedings of the 21st IEEE – 12th NASA Goddard (MSST2004) Conference on Mass Storage Systems and Technologies, April 2004.

[7] Y. Lu, D. Du, and T. Ruwart, "QoS provisioning framework for OSD-based storage system," Proceedings of the 22nd IEEE – 13th NASA Goddard (MSST2005) Conference on Mass Storage Systems and Technologies, April 2005.

[8] OSD Standard ver.1.0(rev.10): http://www.t10.org/ftp/t10/drafts/osd.

[9] mpg321, http://sourceforge.net/projects/ mpg321.