모바일 장치용 객체기반 대용량 멀티미디어 컨텐츠 스토리지 개발

남영진*, 최민석*, 남인길*
*대구대학교 컴퓨터·IT공학부
e-mail:yjnam@daegu.ac.kr

Development of Object-based Multimedia Contents Storage for Mobile Devices

Young Jin Nam*, Min-Seok Choi*, In-Gil Nam*
*School of Computer & Information Tech., Daegu University

요 약

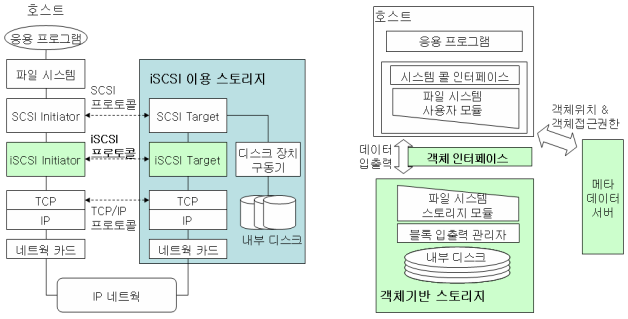
모바일 장치 상에서 대용량 멀티미디어 컨텐츠에 대한 활용이 점차 증가 추세에 있으며, 또한 모바일 장치 내 저장용량의 제약을 극복할 수 있는 저전력 멀티미디어 컨텐츠 저장기법에 대한 개발이 절실히 요구되고 있다. 본 논문에서는 이러한 문제의 한 해결법으로 최근 차세대 스토리지로 각광받고 있는 객체기반 스토리지를 이용한 모바일 장치용 대용량 멀티미디어 컨텐츠 스토리지를 설계 및 구현한다. 제안된 객체기반 스토리지 입출력 구조는 MP3 멀티미디어 컨텐츠를 이용한 성능 평가에서 기존 NFS에 비해서 전력소모 측면에서 약 9%의 향상을 보였다.

1. 서론

모바일 장치내의 프로세서, 네트웍, 입출력 장치의 성능이 높아짐에 따라서 고화질 동영상과 같은 대용량의 데이터로 구성된 동영상 및 MP3 등과 같은 멀티미디어 컨텐츠에 대한 서비스가 일반화되고 있다. 하지만, 이러한 대용량 멀티미디어 컨텐츠 모두를 플래시 메모리에 직접 저장하는 것은 아직 현실적으로 불가능하다. 대신, PMP(Portable Multimedia Player)에서와 같이 하드 디스크를 이용하거나, 모바일 장치에 장착된 무선 네트웍을 통하여 해당 멀티미디어 컨텐츠 데이터를 받아 상위 멀티미디어 플레이어 응용 프로그램을 통하여 사용자에게 컨텐츠를 디스플레이 해주는 형태로 서비스가 가능하다. 특히, 전자의 경우에 하드 디스크의 크기에 제약을 받는 반면에, 후자의 경우에는 저장 공간의 크기에 제약을 거의 받지 않는다. 네트웍을 통해서 멀티미디어 컨텐츠 전송을 위해 TCP 및 IP 프로토콜 뿐만 아니라, NFS 혹은 CIFS 등과 같은 네트웍 파일 시스템, HTTP 프로토콜을 이용하는 웹디스크 등을 이용해야 한다. 이러한 일련의 처리과정은 데이터 전송을 위해서 사용해야 하는 하위 프로토콜 스택 및 파일 시스템 등으로 인해서 모바일 장치 내 CPU에 많은 오버헤드를 야기할 수 있다.

iSCSI 프로토콜[1]을 지원하는 IP 스토리지가 국내외 스토리지 업체들에 의해서 개발 및 상용화 되고 있다. (그림 1.가 참조) 기존 디스크 입출력 표준인 SCSI 프로토콜을 SCSI 버스 및 파이버채널 뿐만 아니라, TCP/IP 프로토콜을 이용하는 IP 네트웍 상에서 동작할 수 있도록 하는 iSCSI 프로토콜에 대한 표준화 작업이 완료되었다. iSCSI 프로토콜은 전세계적으로 널리 구축되어 있는 유무선 IP 네트웍 망을 이용하여 언제 어디서든지 자신이 원하는 데이터를 접근 가능하게 하는 유비쿼터스 환경에 대한 기반 기술로 여기지고 있다. 최근 차세대 스토리지 구조로 객체기반 스토리지[2,3]가 각광받기 시작하고 있다. 객체기반 스토리지(OSD: Object-based Storage Device)에서 사용자 데이터는 다양한 크기와 속성으로 표현되는 객체(Object)의 형태로 존재한다. 즉, 한 멀티미디어 컨텐츠 파일, 데이터베이스 테이블, 혹은 블록 하나가 객체로 표현될 수 있다. 객체기반 스토리지는 기존 블록 인터페이스 기반 스토리지에 비해서 데이터 공유성, 성능 확장성, 보안성, 스토리지 지능화 측면에서 장점이 존재한다. (그림 1.나 참조) 객체기반 스토리지에 대한 연구는 미국 내 학계, 기업, 그리고 표준화 단체에서 활발히 진행 중이며, 특히 객체기반 스토리지는 iSCSI 프로토콜을 기반 환경으로 이용하고 있는 추세이다. OSD는 기존 SCSI 표준의 새로운 SCSI 명령어의 형태로 T10 표준[4,5](버전 1, rev 10)으로 2004년 9월에 채택된 상태이다. 현재는 멀티-객체, 스냅샷 등의 확장 기능 제공을 위한 버전 2에 대한 작업이 활발히 진행 중이다.

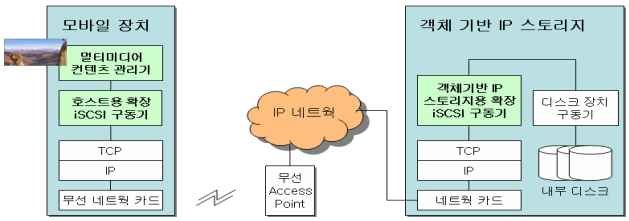
본 연구에서는 지속적인 전원을 공급받는 iSCSI 프로토콜을 이용하는 객체기반 스토리지(이하 객체기반 스토리지)를 이용하여 멀티미디어 컨텐츠를 효율적으로 입출력할 수 있는 기법을 고안하고자 한다. 다양한 멀티미디어 컨텐츠 중에서도 특히 MP3 파일에 대한 서비스에 초점을 맞추었으며, 실제 구현을 통해서 기존 NFS 기법을 이용한 파일 서비스에 비해서 전력 소모측면에서 이득이 있음을 보인다.



(그림 1) iSCSI 프로토콜을 이용한 iSCSI 및 객체기반 스토리지 구조

2. 제안된 객체기반 스토리지 입출력 환경

제안된 모바일 장치용 객체기반 멀티미디어 컨텐츠 IP 스토리지는 (그림 2)와 같은 입출력 환경을 갖는다. 기본적으로 모바일 장치는 객체기반 IP 스토리지를 TCP/IP 프로토콜 스택을 기반으로 한 IP 네트웍을 통하여 접속한다. TCP/IP 스택 상에는 기존 SCSI 프로토콜을 네트웍 상에서 동작시키기 위한 iSCSI 프로토콜이 동작한다.



(그림 2) 멀티미디어 컨텐츠 서비스를 위한 모바일 장치용 객체기반 IP 스토리지 입출력 환경

개발된 객체 기반 IP 스토리지 입출력 환경은 호스트용 확장 iSCSI 구동기, 객체기반 IP 스토리지용 확장 iSCSI 구동기, 그리고 멀티미디어 컨텐츠 관리기로 구성된다. 첫째, **호스트용 확장 iSCSI 구동기**는 모바일 장치 상에서 동작하며 기본적인 iSCSI 이니시에이터(initiator) 기능을 제공한다. 또한, 객체기반 스토리지용 OBS(Object-based Storage) 이니시에이터 측 SCSI 명령어를 추가적으로 지원한다. 둘째, **객체기반 IP 스토리지용 확장 iSCSI 구동기**는 스토리지 상에서 동작하며, 기본적인 iSCSI 타겟(target) 기능을 제공한다. 또한, 객체기반 스토리지용 OBS 타겟 측 SCSI 명령어를 추가적으로 지원한다. 끝으로, **멀티미디어 컨텐츠 관리기**는 확장 iSCSI 이니시에이터 구동기를 통한 멀티미디어 컨텐츠 인출(서비스) 기능, 멀티미디어 컨텐츠 사용자 인덱싱 기능, 그리고 멀티미디어 컨텐츠에 대한 객체기반 스토리지 관련 메타 데이터 관리 기능을 제공한다. 아래에서는 이들 각각에 대한 구체적인 설계 내용에 대해서 기술한다.

2.1 호스트용 확장 iSCSI 구동기

호스트용 iSCSI 확장 구동기는 모바일 장치 상의 커널 내에서 동작한다. 또한, 이 확장 구동기는 기존 iSCSI에 OSD(Object-based Storage Device)용 SCSI 프로토콜을 지원하기 위한 SCSI 명령어(Command)들을 추가적으로 포함하도록 설계 및 구현되었다. OSD SCSI 각 명령어들은 SCSI 명령어 매체인 CDB(Command Descriptor Block) 형태로 타겟에 전달된다. <표 1>은 OSD CDB의 일반적인 구조를 나타낸 것이다.

<표 1> OSD CDB 구조[4]

Bit / Byte	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							

OPERATION CODE(7Fh)는 CDB가 OSD_CDB라는 것을 구분하기 위한 코드이다. SERVICE ACTION 필드가 특정 OSD 명령어를 나타낸다. SCSI는 OPERATION CODE를 보고 CDB를 타겟으로 보낸다. SERVICE ACTION은 OSD 명령어를 저장하는 곳으로서 타겟은 이것에 따라서 실행할 명령어를 결정한다. OPTION BYTE는 SERVICE ACTION을 수정하거나 제어할 때 사용되는 필드이다. GROUP ID는 SERVICE ACTION을 적용할 그룹객체를 결정한다. USER ID는 SERVICE ACTION을 적용할 객체를 결정한다. SESSION ID는 SERVICE ACTION을 수행하기 위한 값이다. LENGTH는 데이터 전송 크기이며 OFFSET은 READ/WRITE을 위한 오프셋의 크기를 나타낸다. GET_ATTRIBUTES_PAGE는 반환되는 값을 결정하기 위한 것이고 GET_LIST_LENGTH는 반환되는 값을 결정하기 위한 것이다. GET_ALLOCATION_LENGTH는 반환되는 데이터의 크기이며 SET_LIST_LENGTH는 반환되는 값을 결정하기 위한 것이다. 멀티미디어 컨텐츠 관리기에서 주로 이용하는 OSD SCSI 명령어들은 <표 2>에 나타낸 CREATE_GROUP, CREATE, WRITE, READ, REMOVE, REMOVE_GROUP이며, 정해진 프로토콜[2]에 따라서 각 명령어를 구현하였다.

2.2 객체기반 IP 스토리지용 확장 iSCSI 구동기

객체기반 IP 스토리지용 확장 iSCSI 구동기에서는 호스트(모바일 장치)로부터 전송받은 CDB를 해석하고 SERVICE ACTION에 있는 명령어 코드에 따라 해당 서비스를 제공한다. 아래에서는 호스트로부터 받은 CREATE_GROUP, CREATE, WRITE, READ, REMOVE, REMOVE_GROUP 각 명령어에 대한 구체적인 서비스 내용을 기술한다. 객체기반 IP 스토리지에서는 Object를 관리하기 위한 일종의 파일 시스템이 필요로 한데, 본 기술개발에서는 이를 커널 수준의 파일 시스템이 아닌 사용자 수준의 파일 시스템 형태로 제공한다.

<표 2> OSD SCSI 명령어[4]

명령어	기능
CREATE GROUP	Group ID를 생성하고 생성된 Group ID를 이름으로 하여 Group Object가 생성하고, 그룹 ID를 반환함
CREATE	특정 Group ID내에 사용자 ID 및 User Object를 생성하고, User ID 값이 반환함
WRITE	Group/User ID를 보고 해당 Object를 찾은 후, OFFSET에 있는 값을 보고 Write를 시작할 위치로 이동, LENGTH에 들어있는 값만큼 Write함
READ	Group/User ID를 보고 해당 Object를 찾은 후, LENGTH 값 만큼 임시 버퍼 할당, OFFSET 값을 보고 Read를 시작할 위치로 이동, LENGTH 값만큼 데이터를 Read하여 임시 버퍼에 저장 후 이니시에이터로 전송함
REMOVE	Group/User ID를 보고 해당 User Object를 찾은 후 삭제함
REMOTE GROUP	Group ID를 보고 해당 Group Object를 찾은 후 삭제함

2.3 멀티미디어 컨텐츠 관리기

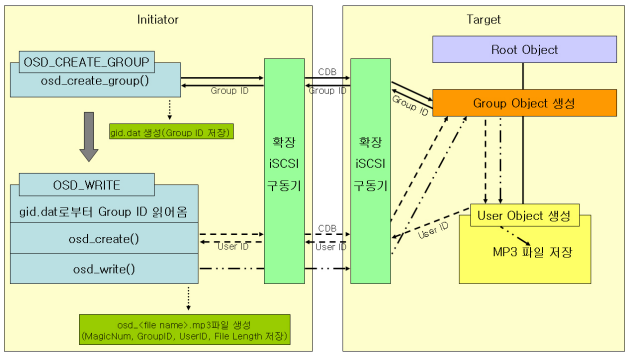
멀티미디어 컨텐츠 관리기의 기능은 크게 2가지로 구분된다. 하나는 멀티미디어 컨텐츠를 타겟에 저장하고 삭제하는 기능을 하는 멀티미디어 컨텐츠 저장/삭제 관리이며, 다른 하나는 저장된 멀티미디어 컨텐츠를 읽어 와서 실행해주는 멀티미디어 컨텐츠 실행기(MP3 Player) 부분이다.

(1) 멀티미디어 컨텐츠 저장 및 삭제 관리

멀티미디어 컨텐츠를 객체기반 IP 스토리지에 저장하면 24-바이트 크기의 파일(이하, 리스트 파일이라 함)을 호스트(모바일 장치)내의 로컬 파일 시스템에 생성한다. 사용자는 생성된 리스트 파일명을 참조함으로써, 모바일 장치에 저장된 멀티미디어 컨텐츠 목록을 확인할 수 있다. 차후에 멀티미디어 컨텐츠 실행기는 이 파일의 정보를 보고 컨텐츠를 실행한다. 리스트 파일은 4-바이트 매직값(Magic number), 4-바이트 그룹 ID, 8-바이트 유저 ID, 그리고 8-바이트 파일 크기 정보로 구성되어 있다. 이 중에서 매직값은 해당 멀티미디어 컨텐츠가 객체기반 IP 스토리지에 저장된 것인지 로컬 파일 시스템에 저장된 것인지를 판단하는데 사용된다.

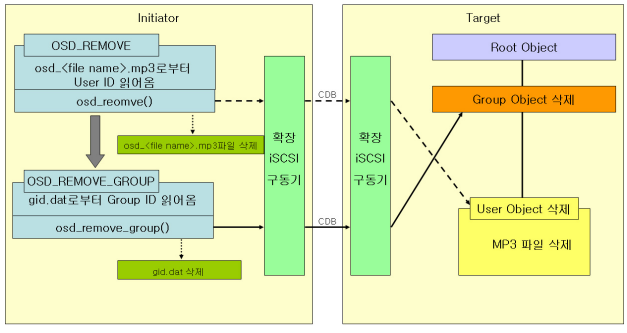
멀티미디어 컨텐츠 저장/삭제 관리는 커널 내의 SCSI 프로토콜 인터페이스를 직접적으로 구동하며, 파일 저장 및 삭제 작업을 위해서 아

래와 같은 OSD SCSI 명령어를 이용한다. 파일을 저장하기 위해서는 OSD_CREATE_GROUP과 OSD_WRITE 명령을 이용한다. 이 명령을 이용하면, 위의 그림 3에서 Group ID와 User ID를 생성할 수 있다. Group ID는 OSD_CREATE_GROUP 명령을 통하여 초기에 한번만 객체기반 IP 스토리지로부터 할당받는다. 할당 받은 Group ID 값은 gid.dat라는 형상 파일에 저장해 두었다가, 차후 멀티미디어 컨텐츠 생성 시에 이용한다. 각 멀티미디어 컨텐츠를 저장할 때마다, 모바일 장치는 객체기반 IP 스토리지(타겟)으로부터 새로운 User ID를 할당 받는다. 최종적으로 Group ID와 User ID는 리스트 파일에 저장된다. 다음으로, 파일을 삭제하기 위해서는 OSD_REMOVE와 OSD_REMOVE_GROUP을 이용한다. <그림 3>은 위에서 설명한 바와 같이 멀티미디어 컨텐츠가 타겟에 저장되는 과정을 보여주고 있다.



(그림 3) 멀티미디어 컨텐츠 저장 과정

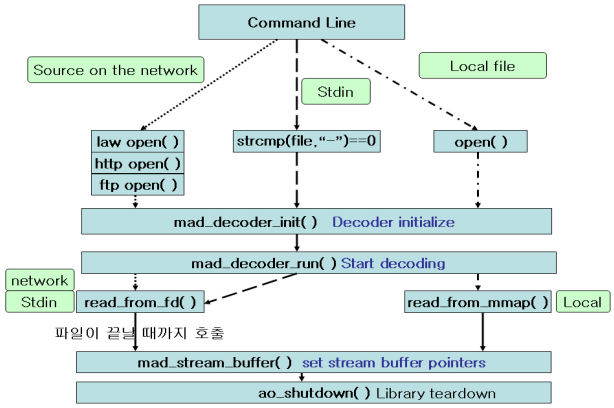
한 멀티미디어 컨텐츠 파일은 객체기반 IP 스토리지 내에서 하나의 객체(object)로 맵핑 되는데, 이중에서 가장 root에 해당하는 Root Object는 객체기반 IP 스토리지용 확장 iSCSI 구동기가 처음으로 초기화 될 때 생성된다. 함수 osd_create_group()가 호출될 때마다, 타겟에는 새로운 Group Object가 생성된다. 생성된 Group ID는 이니시에이터의 gid.dat 파일에 저장되며 이 파일은 Group ID만을 가지며 멀티미디어 컨텐츠 저장 및 Group Object 삭제에 사용된다. 다음으로 OSD_WRITE 명령을 실행하면 우선적으로 gid.dat 파일에서 Group ID를 읽어오고 osd_create()가 호출되어 해당 Group Object에 User Object를 생성한다. User ID는 매직값, Group ID, File Length와 함께 리스트 파일에 저장되고 User Object에 멀티미디어 컨텐츠가 저장된다. 생성되는 리스트 파일명은 osd_<file name>.mp3와 같이 저장되는 원본 파일의 파일명으로 생성된다. 이 파일은 저장된 멀티미디어 컨텐츠의 메타데이터를 가지고 있다. 또한 사용자는 이 리스트 파일 목록을 보고 타겟에 저장된 멀티미디어 컨텐츠 목록을 확인할 수 있다. (그림 4)는 멀티미디어 컨텐츠가 객체기반 IP 스토리지(타겟)에서 삭제되는 과정을 보여주고 있다.



(그림 4) 멀티미디어 컨텐츠 파일 삭제 과정

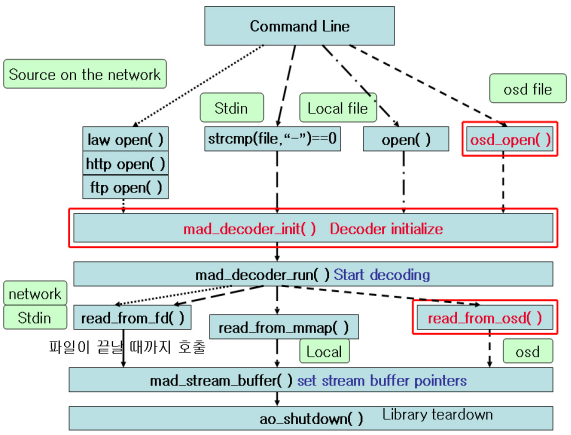
(2) MP3 실행기

본 연구에서 개발한 멀티미디어 컨텐츠 실행기는 리눅스용 기존 mpg123의 변형인 mpg321 MP3 실행기 코드[6]를 기반으로 하였다. (그림 5)는 mpg321의 실행기의 개략적인 구조이다. mpg321은 mmap()을 통한 로컬 파일 시스템에 존재하는 MP3 파일 실행 뿐만 아니라, ftp 및 http를 통하여 네트워 상에 존재하는 MP3 파일에 대한 실행도 가능하도록 해준다. 오디오 재생을 위해서 mad 라이브러리를 이용하고 있다.



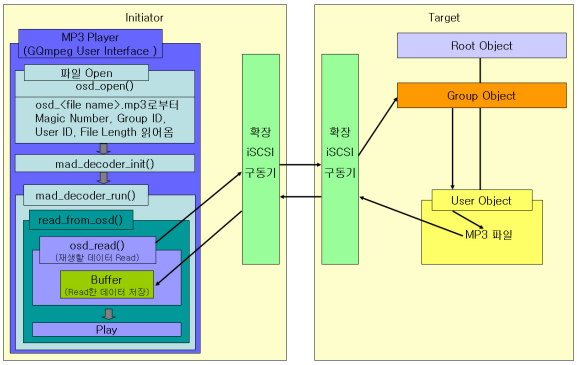
(그림 5) mpg321 실행기 개략 구조

mpg321 실행기를 이전에 설계한 호스트용 확장 iSCSI 구동기와 연동을 시켜주기 위해서 아래의 (그림 6)과 같이 수정을 하였다. 첫째, 객체기반 스토리지 내에 저장된 멀티미디어 컨텐츠 파일을 열기(open)위해 osd_open() 함수를 추가했다. 이 함수는 컨텐츠 저장시 생성된 osd_<file name>.mp3 파일을 여는 작업을 담당한다. 둘째, 타겟에서 데이터를 가져와서 Stream Buffer에 넣어 주는 함수인 read_from_osd() 함수를 추가하고 이 함수를 이용하도록 mad_decoder_init()에 인자로 넣어줬다. 끝으로, mpg321 실행기는 커맨드라인 수준의 실행기로써 사용자 인터페이스가 불편하며, 이를 보완하기 위해 mpg321는 Back-end로 사용하고 GQmpeg[7]을 Front-end로 사용하여 그래픽기반의 사용자 인터페이스를 제공하였다.



(그림 6) 수정된 MPG321의 실행기 구조

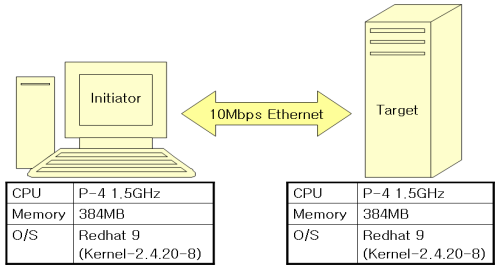
(그림 7)은 멀티미디어 컨텐츠 IP 스토리지에 저장된 MP3 파일을 실행하는 과정을 나타낸 것이다. Play 명령이 실행되면 osd_open()에 의해 osd_<file naem>.mp3에서 매직값을 읽어 와서 멀티미디어 컨텐츠 I객체기반 스토리지용 파일인지 확인 후에 Group ID, User ID, File Length 정보를 읽어온다. mad_decoder_init()에 의해 초기화가 되고 mad_decoder_run()이 호출된다. mad_decoder_run()에는 새로 추가된 read_from_osd()가 있는데 이 함수는 실행이 끝날 때까지 반복적으로 호출되면서 타겟에서 데이터를 읽어오고 Stream Buffer에 넘겨주는 역할을 한다. 또한, 호출될 때 마다 File Length와 읽어온 데이터의 누적크기를 비교하여 실행 종료 플래그를 설정할 지를 결정한다. 종료 플래그가 설정되면 버퍼에 남아있는 데이터를 실행하고 최종 종료한다. osd_read()에서는 읽어온 Group ID, User ID를 가지고 타겟에 저장된 MP3 파일 데이터를 가져와서 버퍼에 넣는다. 버퍼 저장된 데이터는 mad 라이브러리 함수에 의해 재생된다.



(그림 7) GQmpeg/mpg321을 통한 MP3 파일 재생과정

3. 성능 평가

본 과제에서 개발한 모바일 장치용 객체기반 대용량 멀티미디어 컨텐츠 스토리지와의 성능 평가 비교대상은 대표적인 네트웍 파일 시스템인 NFS이다. CPU에서 소모되는 전력을 간접적으로 측정하기 위해서서 CPU 오버헤드(CPU 점유율)를 관측하였다. 호스트 시스템으로는 일반적인 데스크탑 환경 및 유선 네트웍을 이용하였으며, 향후 PDA와 같은 모바일 장치 및 WLAN 혹은 블루투스와 같은 무선 네트웍 환경으로 대체할 예정이다. 성능평가를 수행한 구체적인 하드웨어 및 소프트웨어 실험 환경은 아래의 (그림 8)과 같다. 이니시에이터(호스트) 측에 멀티미디어 컨텐츠 실행기가 동작하며, 타겟에 구현된 객체기반 스토리지가 위치한다.



(그림 8) 성능평가 환경

성능 평가를 위하여 동일한 MP3 파일을 NFS와 객체기반 스토리지에 저장하고 CPU 점유율, 메모리 점유량, Real/User/Sys Time을 각각 10회씩 측정하여 평균 값을 비교했다. 또한, 성능 비교시 NFS의 캐싱 기능은 사용하지 않도록 하였다. Real Time은 프로그램이 시작되어 마칠 때까지 걸린 총 실행 시간이고 User Time은 프로그램 실행에 사용된 CPU 시간이다. 즉, User 모드에서의 CPU 사용시간이다. Sys Time은 프로그램 실행에서 운영체제 호출에 사용된 CPU 시간이다. 즉, 커널 모드에서의 CPU 사용시간이다.

<표 3>에 성능평가 결과를 요약하여 정리하였다. CPU 점유율 측면에서 객체기반 스토리지가 NFS에 비해서 9%정도의 개선을 보이고 있다. 반면에, 메모리 점유량에서는 객체기반 IP 스토리지와 NFS 모두 비슷한 성능을 보이는 것으로 볼 수 있다.

<표 3> NFS와 객체기반 IP 스토리지의 성능평가 결과요약

	NFS	객체기반 스토리지
CPU 점유율	4.7%	4.3%
메모리 점유량	1.98%	2.1%
Real Time	4m7.877s	4m7.148s
User Time	0m12.298s	0m12.249s
Sys Time	0m0.210s	0m0.135s

User Time과 Sys Time 비교에서는 두 항목 모두에서 객체기반 IP 스토리지가 NFS보다 향상된 성능을 보이는 것을 확인할 수 있었다. 특히, 커널에서 수행되는 시간은 36%나 감소한 것을 볼 수 있다. 이는 제안된 스토리지 구조가 파일 시스템을 거치지지 않음으로써 CPU가 수행해야 하는 코드의 양이 줄어들어 나타나는 효과로 볼 수 있다. 결론적으로, 본 연구에서 제안하는 객체기반 스토리지를 이용할 경우에, CPU 오버헤드가 줄어들게 되어 결과적으로 CPU의 소모 전력량을 줄일 수 있을 것으로 기대할 수 있다.

4. 결론

본 연구에서는 모바일 장치의 저장용량의 제약을 극복할 수 있는 객체기반 스토리지 입출력 환경 구조를 제안하고 실제 구현을 통하여 그 성능을 NFS 입출력 환경과 비교 평가하였다. 제안된 스토리지 구조는 최첨단 스토리지 기술을 조기에 수용하여 실제적으로 상용화 가능성이 있는 제품으로 구현한 것에 의의가 존재할 뿐 아니라, 기존 기법에 비해서 모바일 장치에서 절실히 필요로 한 저전력 기능을 제공할 수 있다는데 더 큰 의의가 존재한다.

향후 연구로 현재 Linux Kernel-2.4 기반으로 된 것을 Linux Kennel-2.6 기반으로 이식하는 작업, PDA나 PMP와 같이 독자적인 Embedded Linux를 사용하는 환경으로의 플랫폼 전환, 그리고 WLAN, 블루투스 등과 같은 무선통신 모듈을 이용하는 환경으로 확대 적용하는 작업을 계획하고 있다. 또한, PDA에서 GUI 환경을 위해 많이 사용하는 Qtopia에서 제공하는 MP3 및 동영상 실행기에 본 기술을 적용하는 것을 계획하고 있다.

참고문헌

[1] 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.
[2] 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.
[3] Erik Riedel(Seagate Research), Object-based storage device(OSD) basics: www.snia.org/education/tutorials/spr2005/storage/, 2005.
[4] OSD Standard version 1.0 (rev.10): www.t10.org/ftp/t10/drafts/osd.
[5] SNIA - Storage Networking Industry Association. *OSD: Object Based Storage Devices Technical Work Group.* http://www.snia.org/tech activities/workgroups/osd/.
[6] Project mpg321, http://sourceforge.net/projects/mpg321.
[7] Project GQmpeg, http://sourceforge.net/projects/gqmpeg.