MMDB로 구현한 내장형 리눅스 파일 시스템 설계

이동근[0] 김기천
건국대학교 컴퓨터공학과
{dklee, kckim}@kkucc.konkuk.ac.kr

Design of a embedded Linux file system using MMDB

Dong-keun Lee[0] Kee-Cheon Kim
Dept. of Computer Science & Engineering, Konkuk University

요 약

내장형 파일 시스템을 구현하기 위한 MMDB(Main Memory DB)는 주 기억 장치 상주 메모리 데이터베이스 시스템으로서 이동전화 단말기나 PDA(Personal digital assistants)같이 하드디스크를 가지지 않는 시스템을 대상으로 한다. 하드디스크를 고려하지 않는 점에서 기존의 MMDB와는 다른 메모리 관리 방식이 필요하며, 이를 위한 방안으로 내장형 파일 시스템을 구축하여 효율적인 메모리 관리가 가능하다. 내장형 파일 시스템은 리눅스의 가상 파일 시스템과 유사한 기능을 제공하여 다양한 형태의 파일 시스템을 쉽게 구축하고 사용할 수 있으며, OS가 필요에 따라 파일 시스템의 설정을 변경하여 사용할 수 있다. 또한 OS의 메모리관리자와 파일 시스템의 중복 기능을 제거함으로써 시스템의 성능을 개선시킬 수 있고, 데스크 탑 PC와의 데이터 호환성을 향상시킬 수 있다.

1. 서론

MMDB(Main Memory DB)란 주 기억 장치에 상주하여 실시간으로 데이터를 처리할 수 있는 주 기억 장치 상주 데이터베이스 시스템을 말한다. MMDB는 메모리에 비해 현저하게 떨어지는 하드디스크의 억세스 속도로 인해 발생하는 DB처리 속도의 저하를 개선하기 위해 하드디스크에 있는 모든 데이터베이스 파일들을 메모리에 올려놓고 메모리 상에서 작업을 처리한다. 따라서 MMDB에서는 하드디스크에 억세스하는 횟수를 최소화하는 것이 중요한 이슈가 된다. 그러나 본 논문에서 개발하려고 하는 파일 시스템의 기반이 되는 MMDB는 기존의 시스템들과는 달리 하드디스크를 가지지 않는 내장형 시스템을 대상으로 한다. 즉, 이동전화 단말기나 PDA (Personal digital assistants)같이 하드디스크를 가지기 어려운 시스템에 장착되는 내장형 리눅스를 대상 OS로 하는 MMDB이다. 하드디스크 없이 메인 메모리만을 가지는 시스템을 대상으로 하므로 기존의 MMDB와는 다른 방식으로 메모리를 관리할 수 있는 수단을 제공하여야 한다. 여기서는 내장형 MMDB의 구조와 내장형 MMDB로 구현하는 내장형 리눅스 파일 시스템의 설계에 대하여 설명한다.

본 논문은 정보통신연구진흥원의 대학S/W연구센터 지원 사업의 연구비 지원에 의한 것임

2장에서 관련연구 동향을 알아보고 3장에서는 내장형 리눅스를 위한 MMDB의 구조에 대해서 그리고 4장에서는 MMDB로 구현하는 내장형 리눅스 파일 시스템 설계에 대해서 설명하고 5장에서 결론을 맺는다.

2. 관련연구

해외 관련 연구 및 제품으로는 Dali라는 프로젝트명으로 개발된 회복기능과 동시 제어기능을 제공하는 main memory storage manager[2]와 MMDB를 이용해 구축한 번역 시스템인 D□j□ Vu[3] 등이 있으며, 국내에서는 서울대의 '확장된 주기억장치 저장시스템(eXtesion MAin memory Storage system : XMAS)'과[4] 충남대의 범용 OS상에서 다양한 실시간 응용서비스를 위한 주 기억 장치 상주 DBMS인 MrRt[5] 등이 연구 중이며, 한국 전자통신연구원(ETRI)에서 개발된 TDX 및 이동망 교환기를 위한 주 기억 장치 상주 DBMS인 DREAM-S[6] 등이 있다.

3. 내장형 MMDB의 구조

기존의 MMDB 제품들은 프로세스의 가상메모리를 대상으로 한다.[2] 하드디스크에 있는 데이터 파일들을 프로세스가 사용하는 가상 메모리 공간에 로드하여 사용하는 것이다. 따라서 OS의 스와핑 정책에 의해 메모리와 하드디스크간에 스왑이 발생하며 자원 공유를 위한 정책이 필요

하다. 이에 반해 내장형 리눅스를 대상 OS로 하는 내장형 MMDB는 하드디스크를 배제하고 오로지 메모리만을 갖는 시스템을 대상으로 하여 설계된다. 하드디스크가 없기 때문에 메모리를 OS의 작업공간뿐만 아니라 파일 시스템을 위한 공간으로도 사용할 수 있어야 하므로 그림1과 같은 메모리 구조를 갖게 된다. 파일 시스템을 위한 공간은 메모리를 마치 하드디스크인 것처럼 사용할 수 있도록 하기위한 공간이다.

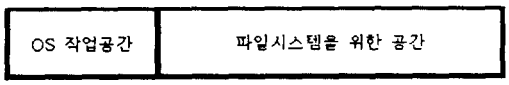

그림 1 내장형 OS 메모리 구조

프로그램 코드 파일과 데이터 파일이 저장된 공간과 OS의 작업 공간이 모두 같은 메모리 공간에 위치하므로 프로세스의 관점에서는 연속적인 선형 메모리 주소 공간을 사용하는 것과 같은 효과를 보게 된다. 메모리의 두 공간의 성격상 각각에 대한 관리 정책은 따로따로 적용되어야 하며, 스왑 기법을 적용하여 파일 시스템 공간의 사용하지 않는 부분을 OS의 작업공간으로 확장하여 사용함으로써 시스템의 성능을 극대화 할 수 있다. 내장형 MMDB는 그림1에서의 파일 시스템 공간에 대한 관리를 맡는다. 메모리 할당과 반납에 대한 처리와 내장형 리눅스의 메모리 관리자와 연계하여 전체 메모리의 효율적 사용을 위한 정책을 마련하게 된다.

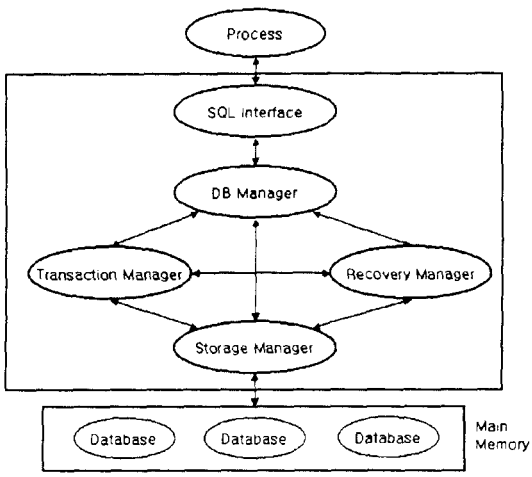

그림 2 내장형 MMDB 의 구조

그림2는 내장형 MMDB 의 대략적인 구조를 나타낸다. MMDB 의 각 기능은 모듈화 되어 컴포넌트로 구성된다. SQL Interface는 커널 및 유저 프로세스에게 MMDB에 접근할 수 있는 수단을 제공한다. SQL문과 기본 연산 함수 등을 제공하며 외부로부터 질의를 받아 이를 해석하여 DB manager에게 서비스를 요청한다. DB manager는 SQL interface로부터 서비스 요청을 받아 transaction manager, recovery manager, storage manager 와 연계하여 작업을 처리하게 된다. Recovery Manager는 내장형 MMDB가 기존의 MMDB들과는 달리 하드디스크를 가지지 않는 환경 에서 동작되므로 기존의 방법과는 차별화 된 회복 기법이 요구된다. 마지막으로 Storage Manager는 MMDB가 사용하게 될 메모리를 관리하는 역할을 한다. Storage manager가 관리하게 될 메모리 영역은 그림1에서 파일 시스템을 위한 공간이 된다. 파일 시스템 공간에 대한 접근은 MMDB의 storage manager를 통해서만 가능하다. 이렇게 함으로써 사용자와 프로세스는 메모리상의 파일 구조에 상관없이 일관된 인터페이스를 통해 메모리에 접근할 수 있으며 전체 메모리를 연속적인 선형 공간으로 인식하게 된다.

4. MMDB를 이용한 내장형 리눅스 파일 시스템 설계

리눅스는 기본 파일 시스템인 Ext2외에도 minix, msdos, vfat, iso9660 등 다양한 파일 시스템을 지원한다. 뿐만 아니라 각 파티션 별로 서로 다른 파일 시스템을 마운트하고 동시에 사용할 수 도 있다. 물론 사용자 입장에서는 현재 파일 시스템에 상관없이 항상 동일한 인터페이스를 대하게 된다. 이것은 리눅스에서 제공하는 가상 파일 시스템으로 인해 가능하다.[1]

가상 파일 시스템은 리눅스에서 다양한 파일 시스템을 지원하기 위한 기술로서 커널과 실제 파일 시스템사이에서 둘 사이를 이어주는 역할을 한다. 파일 시스템에 대한 작업을 하기 위해서는 프로세스가 가상 파일 시스템에 시스템 콜을 발생시키고, 가상 파일 시스템은 해당 파일 시스템에 작업을 요청함으로써 시스템 콜을 처리한다. 지원 가능한 파일 시스템에 대한 정보와 각 파일 시스템의 i-node 테이블과 슈퍼블록 등 필요한 정보들과, 파일을 다루거나 i-node를 다루기 위한 함수포인터를 만들어 놓고, 현재 로딩된 실제 파일시스템의 함수들을 이 함수 포인터와 연결하는 방법으로 가상 파일시스템의 인터페이스를 통해 실제 파일시스템을 사용할 수 있다.

가상 파일 시스템을 통해 리눅스에서는 커널이나 어플리케이션들이 현재 OS가 어떤 파일 시스템을 사용하던지 간에 항상 가상 파일 시스템이 제공하는 동일한 인터페이스를 사용함으로써 다양한 파일시스템을 사용할 수 있다.

MMDB로 파일 시스템을 구축하면 내장형 시스템에서도 가상 파일 시스템과 유사한 기능을 수행할 수 있으며, 다양한 파일 시스템의 지원으로 데스크 탑 PC 와의 데이터 호환성을 증가시킬 수 있다.

시스템에 저장되는 모든 파일들과 i-node 테이블, 오픈 파일 테이블, 슈퍼 블록, 파일 기술자 테이블 등을 MMDB의 데이터베이스 파일로 만들고 커널이나 일반 프로세스로부터의 파일시스템에 대한 접근은 반드시 MMDB의 인터페이스를 거치도록 하면 리눅스의 가상 파일시스템과 유사한 기능을 제공할 수 있다. 즉, 그림 3처럼 시스템에서 지원 가능한 파일 시스템들의 상태 정보들과 모든 파일들을 MMDB의 데이터베이스로 구축하여 놓고 시스템 콜이 발생하면 MMDB 인터페이스가 그에 맞는 데이터베이스와 연결 시켜 줌으로서 MMDB를 이용한 내장형 리눅스 파일 시스템을 구축할 수 있다.

MMDB로 파일 시스템을 구축하면 실제 파일 시스템의 설계와 구현을 리눅스 가상 파일 시스템보다 더 단순화 할

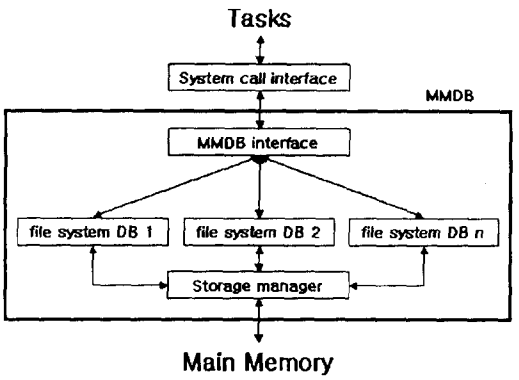

그림 3 MMDB로 구현한 내장형 파일 시스템

수 있으며, 파일 시스템의 교환도 간단히 현재 연결된 데이터베이스 파일을 교체함으로써 이루어 진다. 또, 단순히 데이터베이스파일을 수정함으로써 파일 시스템의 설정을 바꿀 수 있으므로 필요에 따라 OS가 동적으로 파일 시스템을 쉽게 수정할 수 있다. 파일 열기, 파일 읽고 쓰기, 파일 생성과 삭제 등의 파일연산을 DB 생성과 삭제, DB 업데이트 등 MMDB가 제공하는 기본 연산을 이용하여 간단히 구현할 수 있으며, MMDB가 제공하는 SQL문을 이용하여 다양한 파일 조작 함수들을 쉽게 구현 할 수 있다. 그리고, MMDB가 제공하는 메모리 조작 함수를 통해 직접적으로 메모리에 접근함으로써 빠르게 파일을 억세스 할 수 있다.

내장형 파일 시스템을 구현하기 위해 MMDB가 관리해야 할 데이터베이스로는 모든 지원 가능한 파일 시스템의 정보를 가지고 있는 데이터베이스 파일과, 각 파일 시스템별로 슈퍼블록 등 파일 시스템 자체에 대한 정보를 가지고 있는 테이블과, i-node 테이블, 커널의 오픈 파일 테이블, 프로세스의 오픈 파일 테이블, 그리고 파일 시스템 콜 함수 포인터 정보를 가지고 있는 함수 테이블을 가지는 데이터베이스 파일이 있다. 메모리에 저장된 파일의 위치 정보는 i-node 테이블에서 관리한다. 이 들에 대한 관계는 그림 4와 같다.

MMDB파일 시스템을 사용함으로써 얻을 수 있는 이점으로는 서로 다른 파일 시스템을 동일한 인터페이스를 통해 다룰 수 있다는 점 이외에 시스템 성능의 개선을 들 수 있다. 내장형 시스템의 모든 파일들은 메모리 상에 존재하므로 파일 시스템 관리자의 기능이 많은 부분에서 메모리 관리자의 기능과 중복되고 또 디스크와 관련된 기능들이 불 필요해진다. 따라서 저장 공간의 관리를 MMDB의 storage manager에게 일임함으로써 저장 공간의 할당 문제 및 메모리와 파일의 락킹(locking) 등 많은 중복 기능을 제거하여 시스템의 크기를 줄이고 처리 절차의 간소화로 시스템의 성능을 개선할 수 있다. 그리고, 다양한 파일 시스템의 지원이 가능하고 OS가 필요에 따라 파일 시스템 설정을 바꿀 수 있으므로 같은 기종의 시스템이나 데스크 탑 PC등 다른 기종과의 데이터 호환성을 높여 시스템의 다양한 활용을 가능하게 한다.

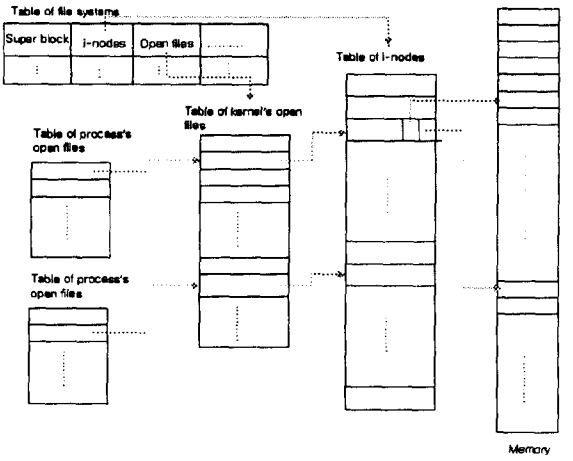

그림 4 파일 시스템 관련 테이블

5. 결론

하드디스크를 가지지않는 시스템을 대상으로 하는 내장형 리눅스의 메모리 구조는 OS의 작업공간과 파일을 저장할 수 있는 파일시스템공간으로 구분되어야 한다. 이런 구분은 OS가 어떤 메모리 관리정책을 사용하는가에 따라 그 경계가 모호해 질 수 있지만 최소한의 시스템 성능을 보장하기 위해서는 OS를 위한 최소 공간이 보장되어야 하며, 최대한의 성능을 나타내기 위해 파일시스템 공간의 미 사용부분을 충분히 활용할 수 있어야 한다. 따라서 메모리 공간의 유동적인 활용이 가능해야 하는데 이는 내장형 MMDB를 사용함으로써 해결할 수 있다.

파일시스템공간에 대한 관리를 MMDB가 맡음으로써 파일 저장을 위한 메모리 요청과 OS의 작업을 위한 부가적인 메모리 요청을 효율적으로 처리할 수 있으며, 스와핑에 들어가는 OS의 오버헤드를 대신함으로써 OS의 부담을 경감시킬 수 있다. 또한 가상 파일 시스템과 유사한 인터페이스를 제공하여 다양한 형식의 파일 시스템을 쉽게 사용할 수 있고 메모리 관리자와 파일 시스템의 중복 기능을 제거하여 시스템의 성능을 개선할 수 있으며, 데스크 탑 PC등 이종의 시스템과의 데이터 호환성을 증가 시킬 수 있다.

6. 참고 문헌

[1] Remy Card, Eric Dumas, Frank Mevel, "The LINUX KERNER book" WILEY, pp. 121 -248 269 -322, December 1998,

[2] Philip Bohannon, Daniel Lieuwen, Rajeev Rastogi, S.Seshadri, Avi Silberschatz, S. Sud arshan, "The Architeture of the Dali Main-Memory Storage Manager", April 1997, http://www.bell-labs.com/project/dali/

[3] DDjD vu, http://www.atril.com/index.html

[4] Xmas, http://kdb.snu.ac.kr/project/Xmas/

[5] MrRt, http://rtislab.chungnam.ac.kr/~rtdb/

[6] DREAM-S, http://dream.etri.re.kr/