과제 3: Flash Memory에서의 Block Mapping FTL 구현

1. 개요

Block mapping 기법(강의자료 "Flash Memory Overview"의 20쪽)을 따르는 FTL을 구현한다. 다음과 같은 제약사항들을 지켜야 한다.

- 파일 I/O 연산은 system call 또는 C 라이브러리만을 사용한다.
- 아래의 (1), (2), (3), (4)의 기능을 ftl.c에 구현한다.
- blockmapping.h와 flashdevicedriver.c는 주어진 그대로 사용하며 예외를 제외하고 수정해서는 안된다. (자세한 것은 두 개 파일 내부의 설명서 참조)
- 네 개의 함수를 테스트하기 위해 main() 함수를 본인 스스로 만들기 바라며, file system의 역할을 수행하는 main() 함수에서의 대략적인 시나리오는 (1) flash memory 파일 생성 및 초기화 (2) ftl_open() 호출 (3) ftl_write()나 ftl_read()를 호출하여 테스트하는 것으로 이루어진다. 테스트 프로그램을 실행하기 할 때 이전에 사용했던 flash memory 파일이 존재하면 이것을 재사용하지 않으며 새로운 flash memory 파일을 생성하여 사용한다.

(1) ftl_open() 구현

일반적으로 FTL을 구현하기 위해서는 논리주소를 물리주소로 변환하는 address mapping table이 필요하다. Block mapping FTL에 맞는 address mapping table을 생각해 보고 이에 맞는 data structure를 각자 정의해서 사용한다 (강의자료 참조). 이러한 data structure를 이용하여 address mapping table을 하나 생성한다 (blockmapping.h에 정의되어 있는 상수 변수를 활용한다).

ftl_open()에서는 일반적으로 여러 초기화 작업을 하는데, 위에서 생성한 address mapping table에서 각 lbn에 대응되는 pbn의 값을 모두 -1로 초기화한다. 여기서 lbn은 0, 1, ..., (DATABLKS_PER_DEVICE-1)의 값을 가진다 (blockmapping.h 참조).

그 이외 free block의 pbn 값에 대한 초기화가 필요하며, 그 값으로 0을 지정한다.

**file system에서 ftl_write()나 ftl_read()를 최초로 호출하기 전에 반드시 초기화와 관련된 ftl_open()를 호출해야 한다.

(2) ftl_write(int lsn, char *sectorbuf) 구현

File system이 ftl_write()를 호출할 때 인자값으로 lsn(=lpn)과 sectorbuf에 저장되어 있는 512B 데이터를 전달한다. FTL은 이 데이터를 flash memory에 쓰기를 해야 하는데, 이때 어떤 물리적 페이지(ppn)에 써야 할지 결정해야 한다. 이것은 block mapping 기법의 동작 원리대로 결정되어야 한다. 함수에서 주어진 lsn에 최초로 데이터를 쓰는지

아니면 갱신(update)을 하는지에 대한 판단이 필요하며, 후자의 경우에는 조금 복잡한 연산들이 요구된다. 마지막으로 address mapping table에 대한 갱신이 필요하며, 또한 필요에 따라 free block에 대한 추가적인 갱신 작업이 요구된다.

FTL과 flash memory 파일 간의 데이터 쓰기는 반드시 페이지 단위로 이루어져야 하며, 또한 flash device driver의 dd_write() 함수의 호출을 통해 이루어져야 한다. Flash device driver의 dd_write(int ppn, char *pagebuf)를 호출하기 전에 FTL은 ftl_write()에서 받은 sectorbuf의 데이터를 pagebuf의 sector 영역에 저장하고, pagebuf의 spare 영역에 ftl_write()의 lsn을 저장하고 이후 이 pagebuf를 인자값으로 dd_write()에 전달한다. Spare 영역에 lsn을 저장할 때 lsn의 크기는 4B이며, spare의 맨왼쪽에 저장한다. 참고로, 이 lsn의 저장 여부를 통해 해당 페이지가 비어있는지를 판단할 수 있다.

File system으로부터 ftl_write()가 호출되었을 때, FTL은 주어진 lsn를 통해 pbn을 알아내고 데이터 갱신인지를 파악할 수 있어야 하며, 데이터 갱신인 경우 free block을 활용하여 쓰기 작업을 수행한다. 이 과정에서 복잡한 연산 작업들이 필요할 수 있다 (과제 2의 쓰기 부분 응용). 데이터 갱신의 경우에는 새로운 free block은 앞서 구한 pbn이 되어야 하며, pbn에 해당되는 블록은 flash device driver의 dd_erase()를 통해 초기화가 되어야 한다.

** dd_write()를 호출하기 전에 pagebuf에 sector 데이터와 spare 데이터를 저장할 때 memcpy()를 쓰면 편리하며 물론 다른 방식을 사용해도 됨

(3) ftl_read(int lsn, char *sectorbuf) 구현

File system이 ftl_read()를 호출하면, FTL은 인자로 주어진 lsn(=lpn)에 대응되는 ppn을 구한 후 flash memory 파일에서 ppn에 해당하는 페이지를 읽어서 인자로 주어진 sectorbuf에 복사하여 전달한다 (당연히 페이지의 spare 복사는 필요 없음). FTL이 flash memory 파일로부터 데이터를 읽을 때는 반드시 페이지 단위를 사용하며 이것은 flash device driver의 dd_read() 함수를 호출함으로써 자연스럽게 해결된다.

** dd_read()를 통해 flash memory에서 페이지를 읽어 온 후 ftl_read()의 sectorbuf에 섹터 데이터를 복사할 때 memcpy()를 사용하면 편리함

(4) ftl_print() 구현

일반적으로 FTL이 제공해야 할 함수는 ftl_open(), ftl_write(), ftl_read() 세 개뿐이며, ftl_print() 함수는 단지 FTL의 address mapping table을 확인하기 위한 용도로 사용하기 위한 것이다. 이 함수는 화면에 lbn, pbn, free block의 pbn을 출력한다. flash memory의 전체 블록의 수가 8이고 이 함수를 호출하는 시점에서 free block의 pbn=5라고 가정할 때 화면에 아래와 같이 출력될 수 있다.

```
lbn pbn
0 7
1 -1
2 1
3 2
4 3
5 -1
6 4
free block=5
```

** ftl_print() 함수를 호출하였을 때 반드시 위와 같은 출력 포맷을 사용해야 하며, 그렇지 않는 경우 채점 프로그램이 제대로 인식을 하지 못하여 불이익을 받을 수 있음

(5) spare 영역에 메타데이터 쓰기

- Flash memory 파일의 모든 페이지에서 16B의 spare 영역 중 첫 4B는 lsn을 저장하여 ftl_write()에서 쓰기가 갱신인지를 파악하기 위한 용도로 사용한다. 함수 ftl_write()가 호출되면 FTL은 해당 페이지에 섹터 데이터와 lsn을 반드시 저장해야 한다. 나중에 이것을 통해 갱신 여부를 판별한다. 해당 페이지의 spare 영역에서 lsn을 읽었을 때 0보다 같거나 큰 정수가 나오면 갱신을 의미하며, 그렇지 않고 0xffffffff이라면 해당 페이지에 대한 첫 번째 쓰기라는 것을 의미한다 (flash memory를 맨처음 초기화할 때 0xff로 채우는 것을 기억하기 바람).

2. 개발 환경

- OS: Linux 우분투 버전 22.04 LTS (Ubuntu 홈페이지에서 버전 확인 가능)
- 컴파일러: gcc 13.2

* 과제 채점 환경은 위와 동일하며, 따라서 프로그램 개발 환경도 위의 환경에 맞추길 권장하며 이를 따르지 않아서 발생하는 불이익은 본인이 책임져야 함

3. 제출물

- 프로그래밍한 소스파일 ftl.c를 하위폴더 없이(최상위 위치에) zip파일로 압축하여 스마트캠퍼스 과제 게시판에 제출한다 (모든 제출 파일들의 파일명은 반드시 소문자로 작성). 제공된 blockmapping.h와 flashdevicedriver.c는 제출할 필요가 없음.
- 압축한 파일은 반드시 학번_3.zip (예시 20201084_3.zip)과 같이 작성하며, 여기서 3은 세 번째 과제임을 의미함

* 채점은 채점 프로그램을 통해 자동으로 처리하기 때문에 위의 사항들을 준수하지 않는 경우 채점 점수가 0이 될 수도 있기 때문에 반드시 준수하기 바라며, 이를 따르지 않아서 발생하는 불이익은 본인이 책임져야 함