< 목 차 >

◉ M-system (ANAND, FMAX)

○ 서론

기존의 Block-Mapping 알고리즘을 기반으로
M-system 의 ANAND 알고리즘과 FMAX 알고리즘을 구현한다.
결론적으로 ANAND 와 FMAX 의 write 횟수와 erase 횟수를 비교하여
성능을 분석한다.

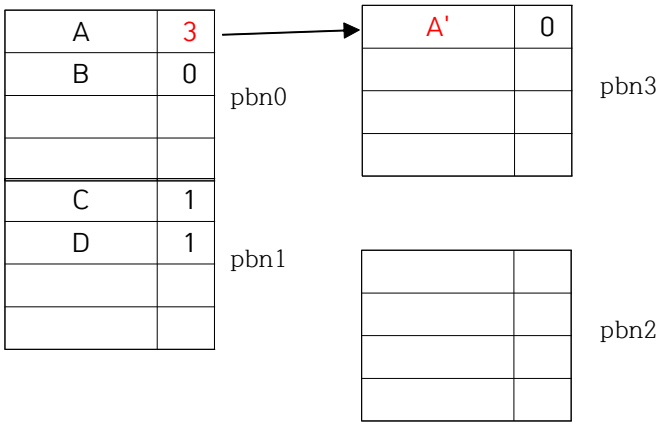
○ 가정

ANAND	FMAX
1.하나의 섹터의 크기는 512 바이트 이고 블록당 섹터의 크기는 32개이다. 2.메모리 생성시 한 개의 여분블록(log block)을 추가로 생성한다 3.합병(merge)를 수행하기위해 하나의 프리블록(free block)를 선정한다 4.프리블록 선정은 메모리의 끝부분부터 데이터가 빈 블록으로 선정한다. 5.합병은 기존의 블록의 유효데이터와 로그블록의 유효데이터를 프리블록으로 이동하고 프리블록이 새로운 데이터 블록이 되는 것을 의미한다.	
업데이트가 된 데이터는 오프셋 값에 따라서 여분블록에 입력한다.	업데이트가 된 데이터는 오프셋 값에 상관없이 여분블록에 삽입한다.
메모리 섹터마다 spare공간에 pbn값을 입력한다.	메모리 섹터마다 spare 공간에 lsn값을 입력한다.
여분블록에는 다른블록의 접근을 금지한다	여분블록에 다른블록의 데이터가 들어와도 된다.

○ ANAND 알고리즘

write 0 A
write 1 B
write 4 C
write 5 D
write 0 A'

lbn	pbn
0	0
1	1
2	2
3	3



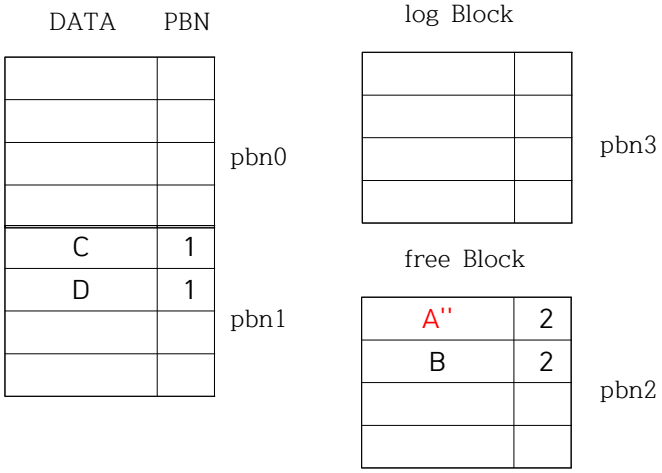
(그림1-1)

ANAND 알고리즘에서는 기존의 블록매핑과 마찬가지로 입력받은 lsn을 블록당 섹터수로 나누어서 몫을 lbn,나머지를 offset 로 구하고 pbn 값을 찾아서 offset에 데이터를 입력한다.
그림(1-1)에서 wirte 5 D 까지 수행하여 메모리에 pbn으로 해당블록을 찾고 offset 값에 데이터를 입력하였다. 이후 write 0 A'를 수행하면 pbn0의 첫 번째 섹터에 데이터가 있기 때문에 로그블록의 같은 오프셋 값인 첫 번째 섹터에 삽입된다.
이후에 기존에 데이터 A가 있던 섹터의 PBN 값을 로그블록으로 바꿔주고
유효데이터인 로그블록에 있는 A'의 PBN을 기존블록의 PBN인 0으로 바꿔준다

이후에 write 0 A''를 수행하면 로그블록의 같은 오프셋 위치에 데이터가 있으므로
합병작업을 실행한다
기존블록의 유효데이터와 로그블록에서 유효데이터를 프리블록으로 옮기고
기존 블록과 로그블록을 지운후에 프리블록에 데이터를 쓰고 매핑테이블을 업데이트한다
합병작업 후의 데이터 이동은 아래 (그림1-2)와 같다

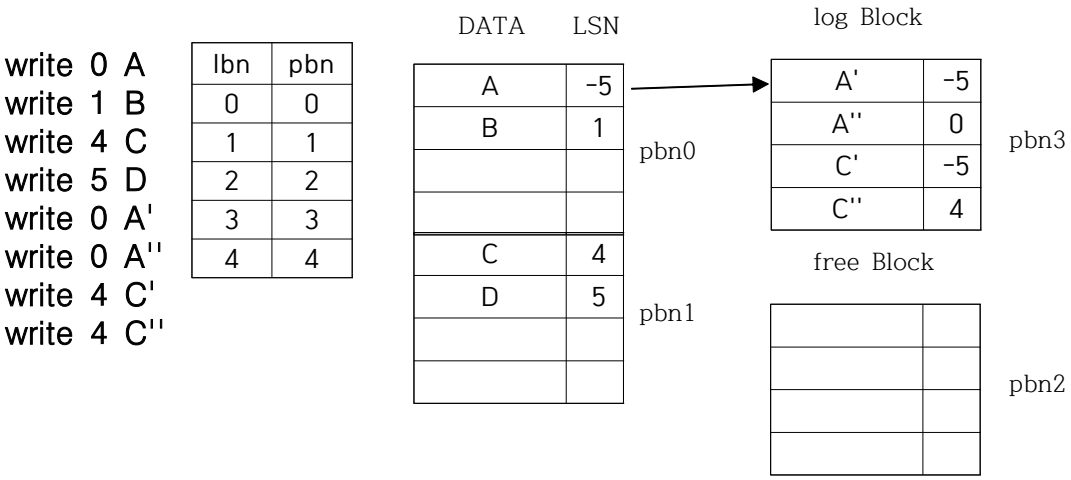
write 0 A
write 1 B
write 4 C
write 5 D
write 0 A'
wrtie 0 A''

lbn	pbn
0	2
1	1
2	0
3	3



(그림1-2)

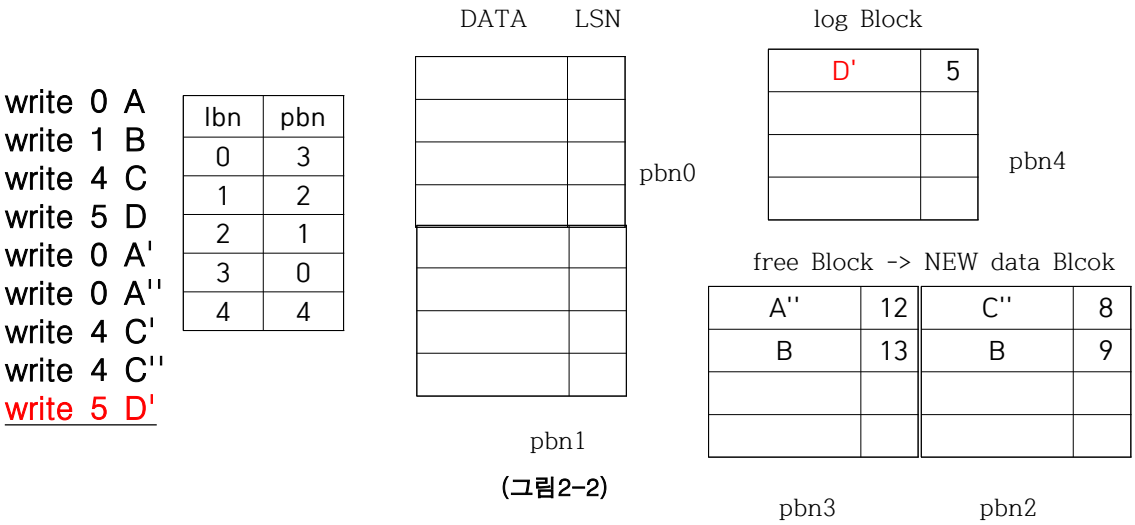
○ FMAX 알고리즘



(그림2-1)

FMAX 알고리즘에서는 기존의 블록매핑과 마찬가지로 입력받은 lsn을 블록당 섹터수로 나누어서 몫을 lbn,나머지를 offset 로 구하고 pbn 값을 찾아서 offset에 데이터를 입력한다.
그림(2-1)에서 write 4 C'' 까지 수행한 결과 업데이트가 일어난 데이터는 오프셋과 상관없이 첫 번째 섹터부터 순차적으로 로그블록에 삽입된다.
업데이트가 일어난 기존 블록의 데이터는 무효데이터('-5')임을 표시한다.

FMAX에서는 합병작업이 로그블록이 가득찼을때만 일어난다.
(그림2-1) 이후에 write 5 D'를 수행하게 되면 로그블록이 가득찬 상태이므로 로그블록에 있는 동일한 블록의 유효데이터와 기존블록의 유효데이터를 프리블록에 이동하고 이동된 데이터는 무효데이터 표시를해주고 프리블록이 새로운 데이터 블록이 되면서 기존 데이터 블록이 지워지고 매핑테이블이 업데이트된다. 로그블록에 무효데이터가 아닌 남아있는 데이터들도 같은블록 데이터를 찾아서 합병을 한다. 합병작업이 모두 끝나면 로그블록이 지워진다. 이후에 데이터 D'는 로그블록에 첫 번째에 쓰여진다.
합병 작업 실시후의 데이터 이동은 그림(2-2)와 같다

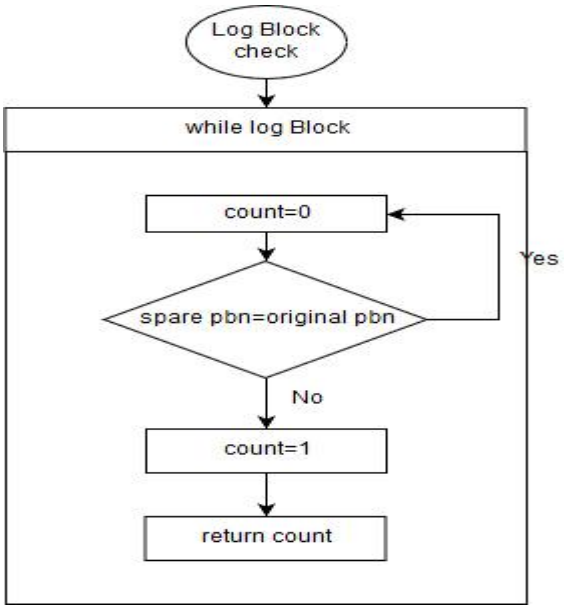


(그림2-2)

ANAND and FMAX

○ FTL_Read

FlowChart



설명

●로그블록 확인

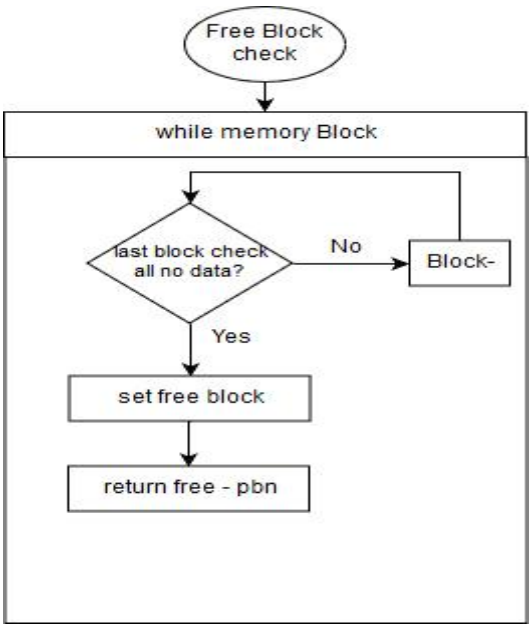
●로그블록 반복

●로그블록을 차례대로 검사하면서 데이터가 비어있는지 가득차있는지 다른블록의 데이터가 들어갈 수있는지 등을 검사한다

●count 값으로 확인

○ FTL_Read

FlowChart



설명

●프리블록 선정

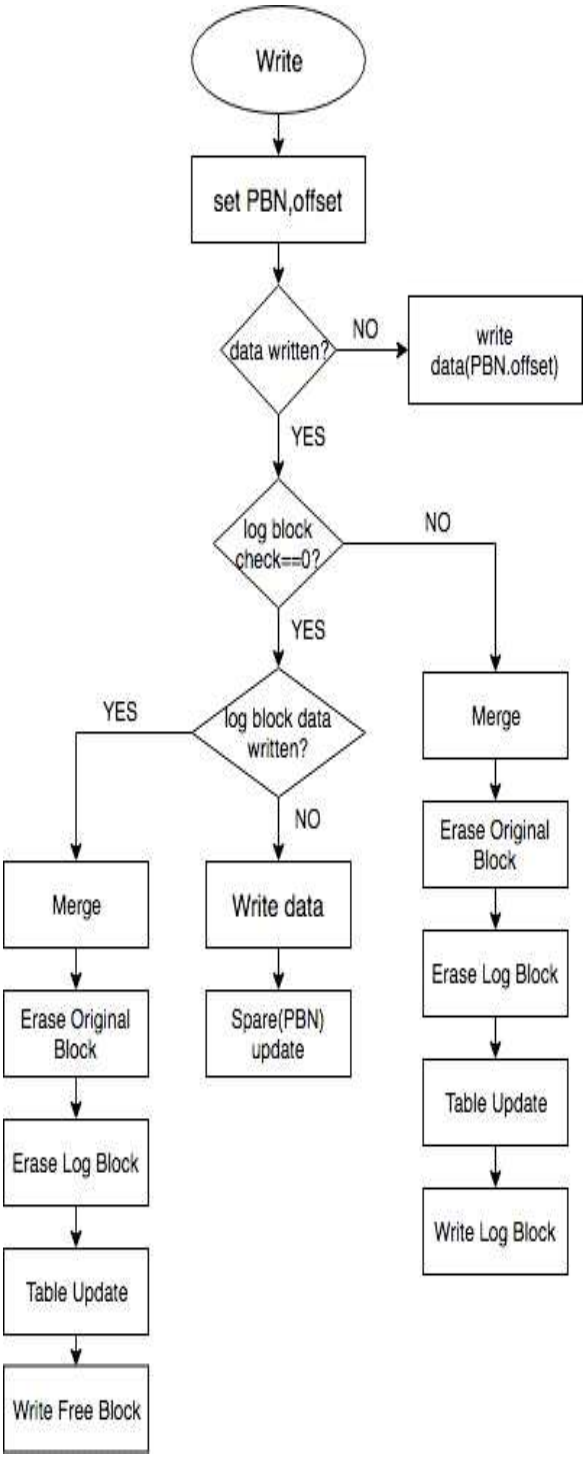
●메모리블록을 뒤에서부터 반복

●마지막 블록부터 데이터가 하나도 들어있지 않은 비어있는 블록을 프리블록으로 선정한다.

ANAND Algorithm

○ FTL Write

FlowChart



설명

●시작

●LSN을 이용해서 몫과 나머지를 이용해 LBN,offset을 구한다

●PBN.offset 자리에 데이터가 없으면 데이터 쓰기

●로그블록을 확인해서 데이터가 비어있는지,다른블록의 데이터가 있는지 확인한다

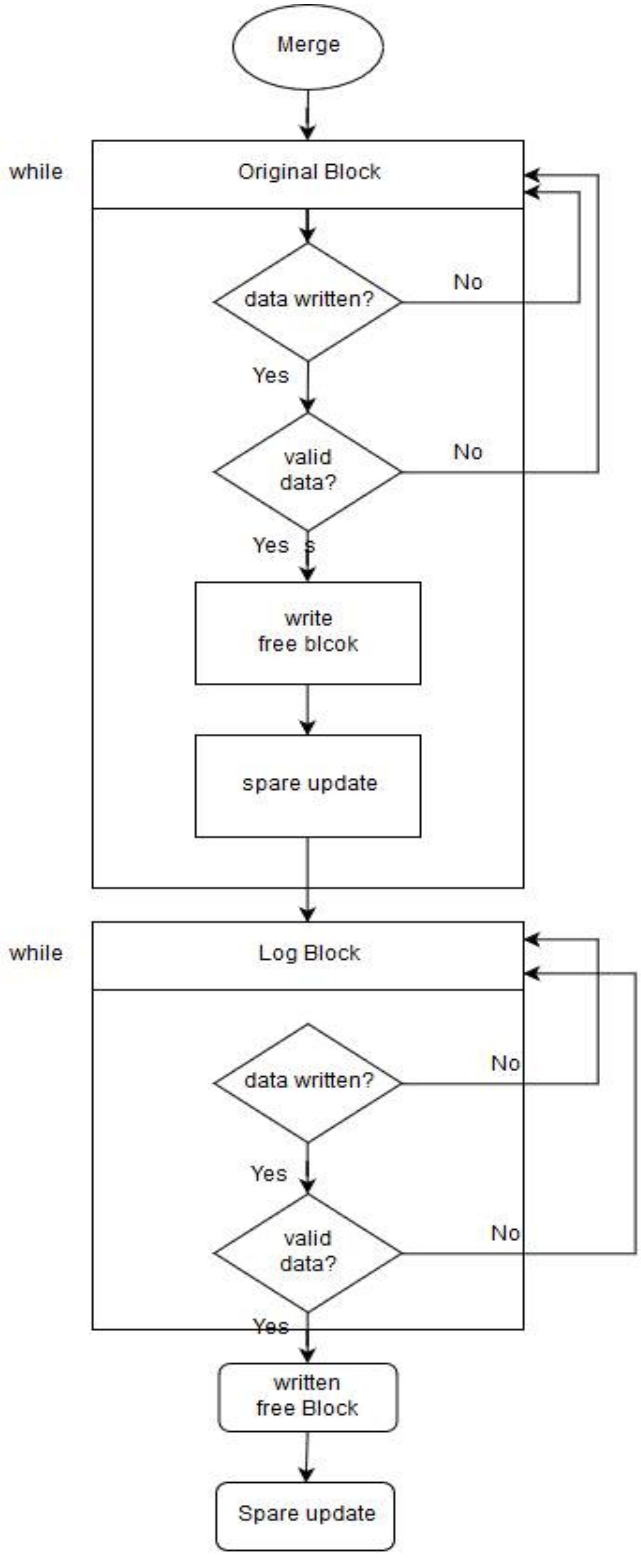
①로그블록 체크가 0이 아니면 다른블록의 데이터에서 업데이트가 일어난 경우이므로 함병을 실시하고 기존블록과 로그블록을 지운다 새로운 데이터는 로그블록에 쓰인다.

②로그블록에 데이터가 없으면 오프셋에 따라서 데이터를 쓴다

③로그블록에 데이터가 있으면 같은 오프셋에서 3번째 업데이트가 일어난 경우이므로 합병을 실시하고 프리블록에 데이터를 쓴다

○ ANAND MERGE

FlowChart



설명

●합병작업

●업데이트가 일어난 기존의 블록을 반복한다

●데이터가 쓰여져있고

●쓰여져 있는 데이터가 유효데이터이면

●프리블록으로 데이터 합병 수행

●여분공간인 PBN값을 업데이트 한다.

●로그블록 반복

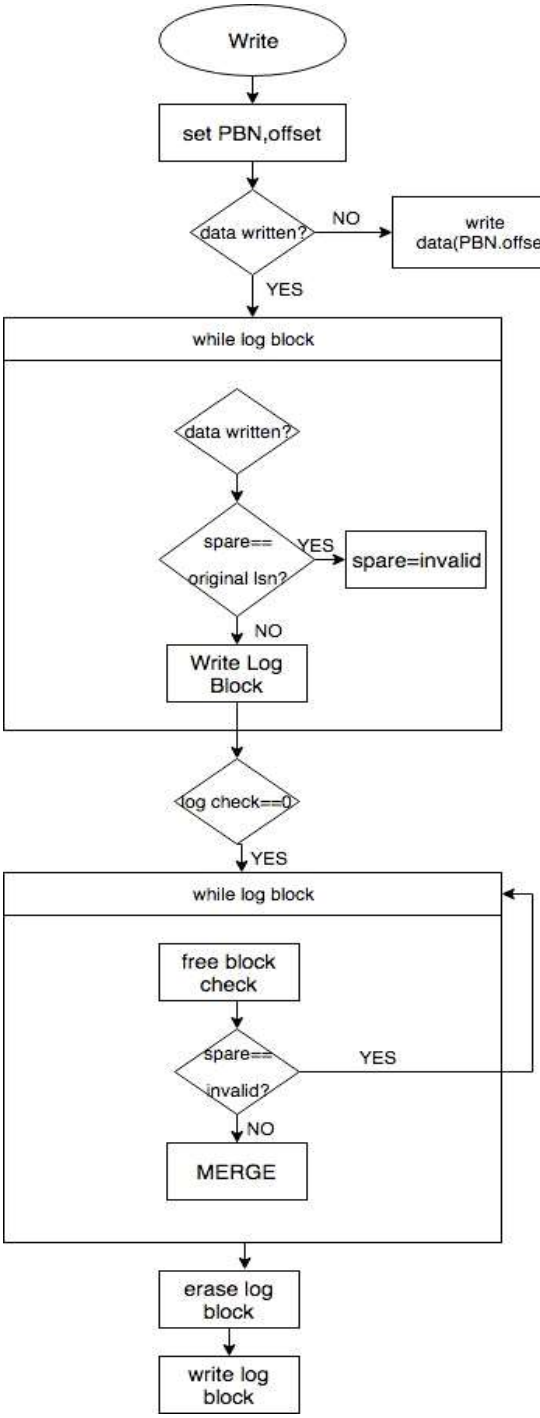
●데이터가 쓰여져있고
쓰여져있는 데이터가 유효한 데이터이면

●프리블록으로 데이터 합병 수행

━ FMAX Algorithm ━

○ FTL Write

FlowChart



설명

●시작

●LSN을 이용해서 몫과 나머지를 이용해 LBN,offset을 구한다

●PBN.offset 자리에 데이터가 없으면 데이터 쓰기

●로그블록을 반복하면서 spare의 lsn값과 기존의 lsn값을 비교한다
같으면 로그블록의 데이터를 무효 데이터 처리하고
아니면 로그블록에 데이터를 쓴다

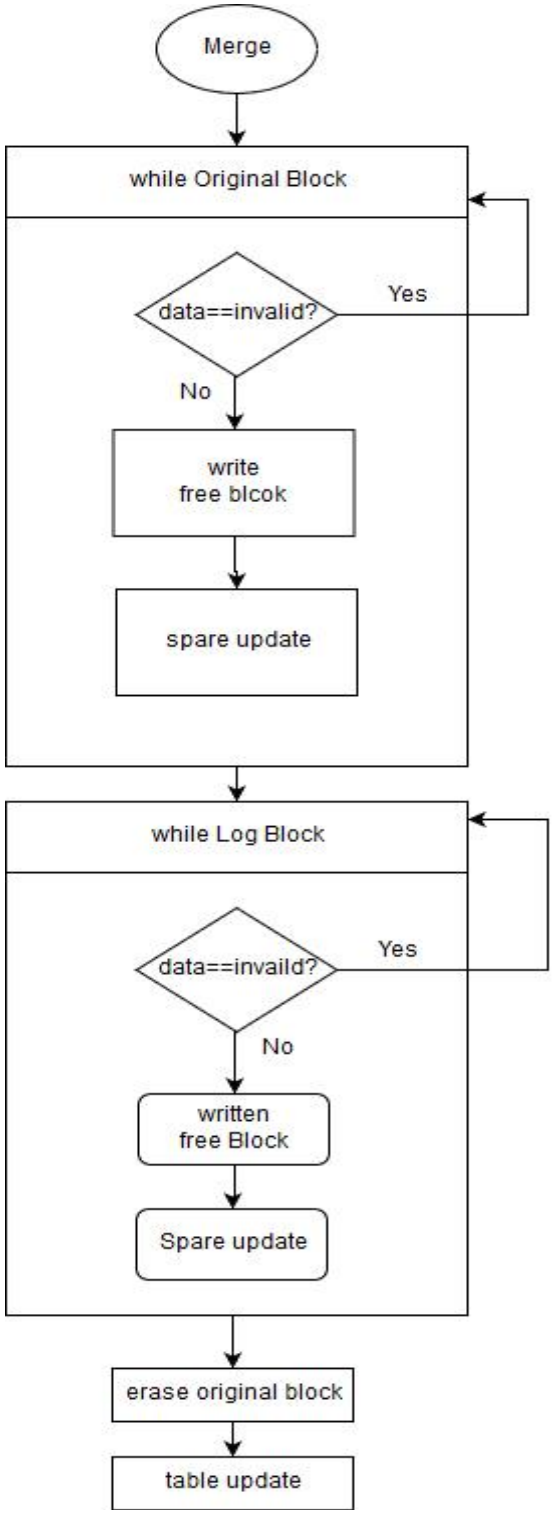
●로그 블록이 다찼으면

●유효한 데이터들만 합병을 수행한다

●합병이 끝났으면 로그블록을 지우고 새로운데이터를 로그블록에 입력한다

○ FMAX MERGE

FlowChart



설명

●합병

●기존의 데이터블록 반복

●유효한 데이터들을 프리블록에 합병하고 spare를 업데이트한다

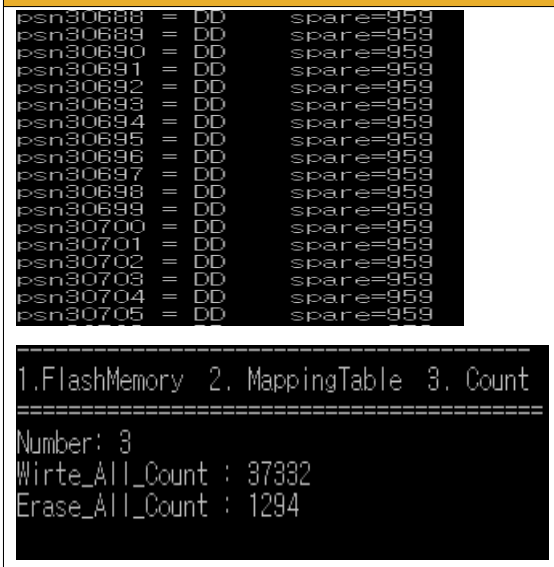
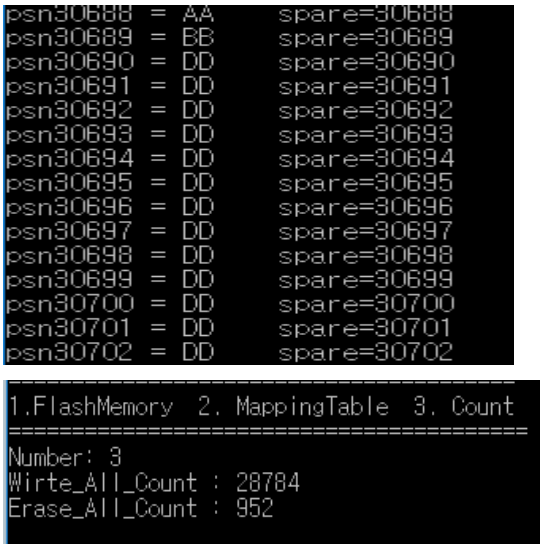
●로그블록을 반복

●유효한 데이터들을 프리블록에 합병하고 spare를 업데이트 한다

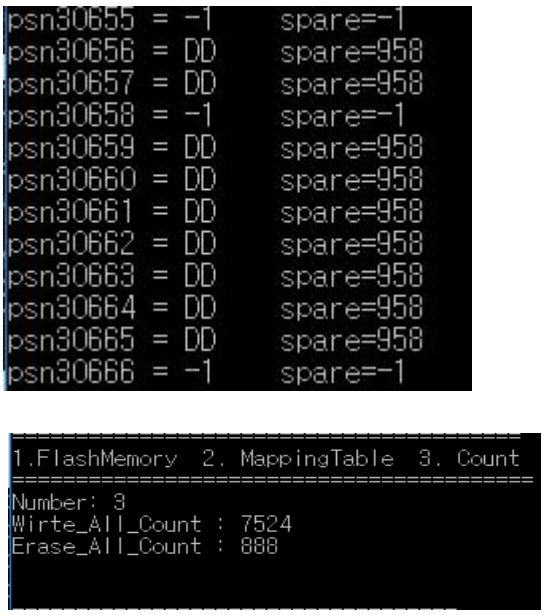
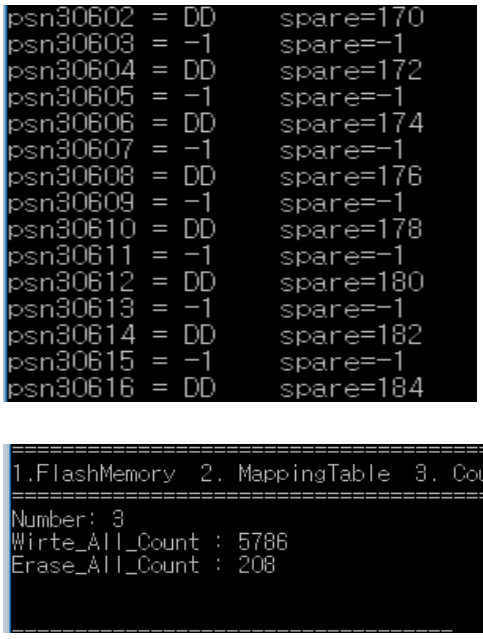
●합병이 완료되면 기존블록을 지우고 매핑테이블을 업데이트한다.

◉연산횟수 출력결과(linux.txt, kodak.txt)

1.linux.txt(15MB)

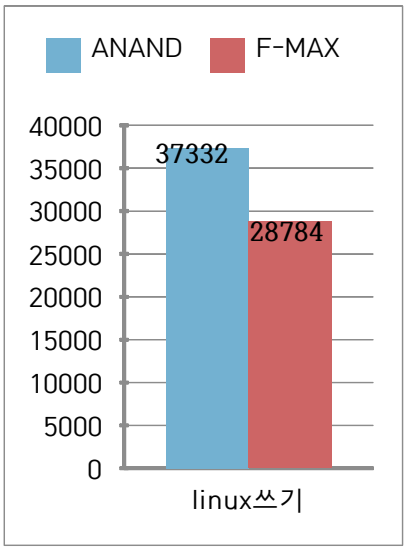
ANAND	FMAX
	

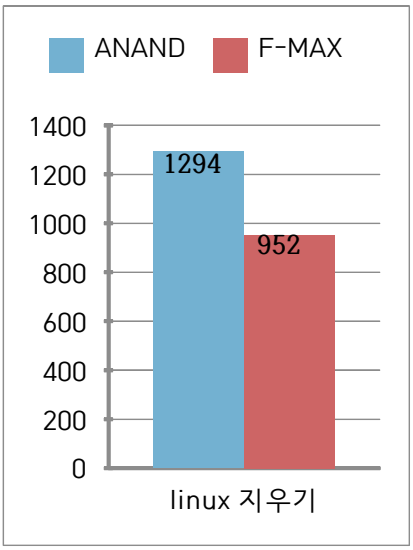
2.kodak.txt(15MB)

ANAND	FMAX
	

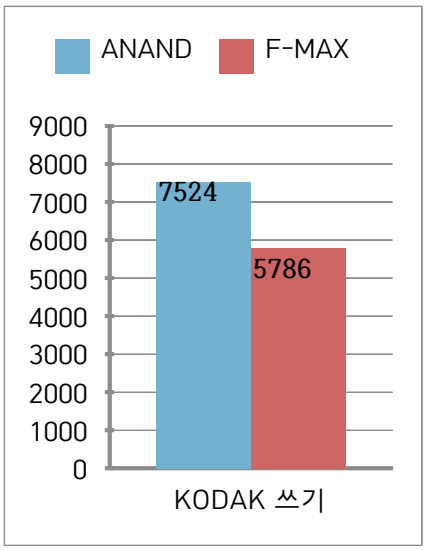
◉연산횟수 비교

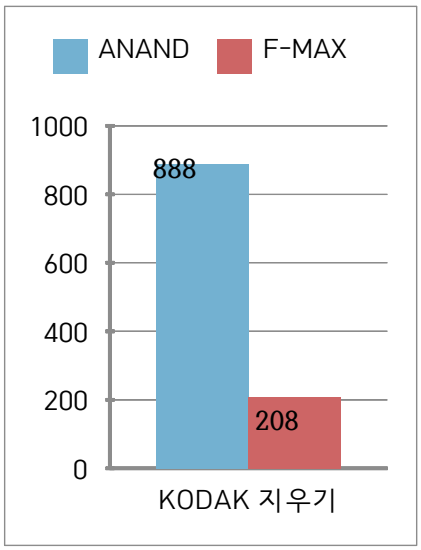
1.linux.txt(15MB)





2.kodak.txt(15MB)





◉결과분석

	ANAND	FMAX	write count	erase count
linux.txt	wirte : 37332 erase : 1294	write : 7524 erase : 888	ANAND>FMAX	ANAND>FMAX
kodak.txt	wirte : 28784 erase : 952	write : 5786 erase : 208	ANAND>FMAX	ANAND>FMAX

ANAND 알고리즘은 업데이트가 일어났을 때 로그블록에 오프셋에 따라서 데이터가 입력되고 FMAX 알고리즘은 오프셋에 상관없이 로그블록에 순차적으로 입력된다.

결론적으로 ANAND는 같은 섹터에 동일하게 세 번 쓰기가 나타나거나 로그블록에 있는 데이터블록이 아닌 다른블록에서 업데이트가 일어났을 때 합병작업을 수행하는 반면에 FMAX는 로그블록이 모두 데이터로 가득 차 있을때만 합병작업이 수행되기 때문에 쓰기 연산이나 지우기 연산에서는 FMAX가 더 적게 수행된다