NAND 플래시 메모리 기반 파일시스템을 위한 더블 캐시 정책 설계

박명규° 김성조
중앙대학교 컴퓨터공학부
blackaqua@konan.cse.cau.ac.kr° sjkim@cau.ac.kr

A Design of Double Cache Policy for File System Based on NAND Flash Memory

Myungkyu Park Sungjo Kim
School of Computer Science & Engineering Chung-ang University

요 약

NAND 플래시 메모리는 특성상 쓰기 횟수가 제한적이라는 단점을 가지고 있어 쓰기 연산이 빈번히 발생하게 되면 NAND 플래시 메모리의 수명이 줄어든다. 이러한 문제점을 해결하기 위해 NAND 플래시 메모리의 특성을 고려한 지연 쓰기 기법이 연구되고 있다. 하지만 지연 쓰기를 하기 때문에 쓰기 횟수는 줄어들지만 캐시 적중률이 낮아진다. 이러한 문제해결을 위해 본 논문에서는 NAND 플래시 메모리 기반 파일 시스템을 위한 더블 캐시 정책을 제안한다. 더블 캐시는 실질적인 캐시인 Real Cache와 요구 페이지의 패턴을 관찰하기 위한 Ghost Cache로 구성된다. 이 정책은 Real Cache에서의 지연 쓰기를 하지 않고, Ghost Cache 공간에서 dirty페이지와 clean페이지를 활용하여 효율적인 지연 쓰기가 가능하도록 설계함으로써 쓰기 횟수를 줄이고, 적중률을 높인다.

1. 서 론[1][2]

오늘날 하드 디스크는 디스크에 대한 접근이 기계적인 동작으로 이루어져 원하는 데이터가 저장된 트랙으로 헤드가 이동하는 탐색시간이 필요하다. 그렇기 때문에 데이터 처리속도가 프로세서의 빠른 처리 속도를 따라가지 못하고 있다. 또한 저장공간의 크기는 점차 대용량화 되어 가고 있어서 반도체 기술의 발전으로 디스크와 프로세서에서의 데이터 처리속도의 차이는 더욱 심해지고 있는 상황이다. 이에 따른 유력한 대체 저장장치로 NAND 플래시 메모리가 급부상하게 되었다. NAND 플래시 메모리는 비휘발성 기억장치로 하드 디스크와는 다르게 탐색시간이 빠르기 때문에 데이터 처리속도가 우수하며 외부의 충격에 강하고, 크기가 하드 디스크에 비해 매우 작고 무게도 가벼운 것이 특징이다. 최근 모바일 기기나 이동식 저장매체, MP3 플레이어 등의 디지털 미디어 기기에서 광범위하게 사용되고 있다.

NAND 플래시 메모리의 읽기, 쓰기 속도는 하드 디스크에 비해 우수하다[1]. 하지만 이미 데이터가 기록된 장소에 덮어쓰기가 불가능하기 때문에 다시 쓰려면 블록 단위로 지우기 연산을 수행한 후 기록해야 하는 단점을 가지고 있다. 또한 쓰기 연산을 반복할 경우 해당 블록을 사용하지 못하게 되는 단점을 가진다.

보다 빠른 처리속도로 데이터를 처리하기 위해서는 NAND 플래시 메모리에 접근하는 횟수를 최소화해야 한다. 이를 위해 메인 메모리의 일부를 메모리 버퍼로 사용해서 접근 횟수를 줄일 수 있다. 메모리 버퍼는 프로세서가 처리하고자 하는 요구 페이지를 저장장치로부터 가져와 버퍼에 저장하고 다시 프로세서가 해당 요구 페이지에 대한 처리를 하려고 하면 디스크에 접근하지 않고 이미 메모리 버퍼 상에 있는 페이지를 바로 처리하게 한다. 메모리 버퍼는 캐시라고도 부르며 캐시의 공간은 메인 메모리의 일부만 사용하기 때문에 캐시에 존재하는 페이지들을 효율적으로 관리해서 처리속도를 높여야 한다. 캐시에는 캐시 공간이 모두 사용되고 있을 때 새로운 페이지가 저장되기 위해서 기존의 페이지들 중 가장 가치가 없는 페이지를 새로운 요구 페이지와 교체하는 기ㄷ법이 필요하다.

기존의 캐시 정책 중 대표적으로 LRU(Least Recently Used)[2]와 LFU(Least Frequently Used)[2] 정책이 있다. LRU는 버퍼 안에서 가장 오래된 페이지를 교체하는 정책으로 가장 보편적으로 사용하는 캐시 정책이며, LFU는 페이지에 대한 참조성을 고려하여 버퍼 안에서 참조가 가장 적은 페이지를 교체하는

[1] 본 연구는 서울시 산학연 협력사업(CR070019) 지원으로 수행되었습니다.
[2] 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터(홈네트워크연구센터) 육성.지원사업의 연구결과로 수행되었음.

정책이다. 플래시 메모리는 이전에 언급했듯이 쓰기 횟수가 제한적이기 때문에 그 비용이 가장 크다. 하지만 LRU, LFU와 같은 캐시 정책은 NAND 플래시 메모리의 특성을 고려하지 않기 때문에 dirty페이지가 쉽게 교체되어 쓰기 연산이 지속적으로 반복 될 우려가 있다. 이에 따라 NAND 플래시 메모리의 수명을 단축시키는 결과를 가지게 되어 캐시의 효율성이 떨어진다. 캐시의 효율성을 높이기 위해서는 dirty가 된 페이지를 쓰기에 대해서 최대한 지연하는 방법과 지연에 따른 적중률 감소를 고려해야 한다. 이러한 이유로 CF-LRU(Clean First LRU)[3], LRU-DPL-CD(LRU-Dirty Page Later Cold write page Detection)[4], FLRU(Flash memory LRU)[5]와 같은 NAND 플래시 메모리 기반 캐시 정책이 제안되었다.

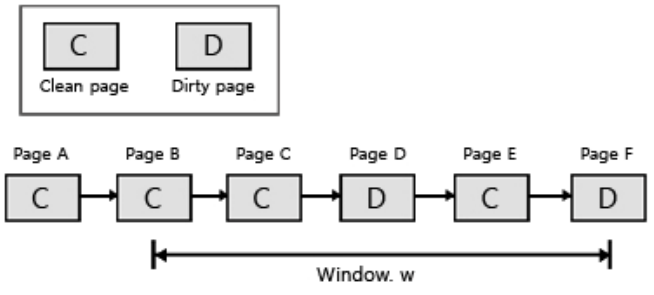
본 논문의 2장은 NAND 플래시 메모리의 특성을 고려한 기존 캐시 정책들에 대해서 기술하다. 3장은 본 논문이 제안하고자 하는 캐시 관리 정책인 DLD-ARC(Double List for Dirty page - ARC)을 설계하고, 다른 기존 캐시 정책들과의 차이점을 설명한다. 4장에서 결론으로 마무리한다.

2. 관련 연구

본 장에서는 기존에 제안된 NAND 플래시 메모리의 특성을 고려한 캐시 정책인 CF-LRU와 LRU-DPL-CD, FLRU의 특성과 장단점에 대하여 기술한다.

2.1 CF-LRU

CF-LRU는 캐시 정책 중 가장 대표적으로 널리 사용되는 LRU 정책을 플래시 메모리 저장장치에 적합하도록 개선한 알고리즘이다. Dirty페이지에 대한 지연 쓰기를 다음과 같은 방법으로 이루어진다.



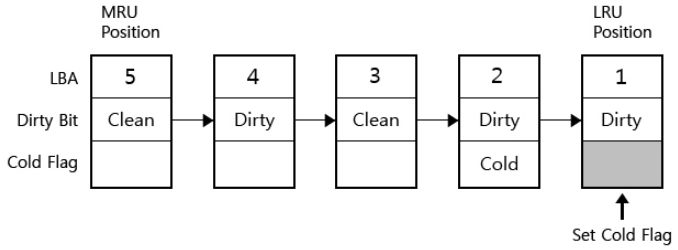
(그림 1) CF-LRU 예

(그림 1)은 CF-LRU의 수행과정을 간단하게 예를 들어 표현하였다. 여기서 페이지의 요청으로 인해 F, E, D, C, B, A 순서로 리스트에 삽입되었고, 이 중 D와 F를 dirty페이지라고 가정한다. 또한 dirty페이지의 지연 쓰기를 위해서 사용자는 윈도우의 w를 정하게 되는데 (그림 1)에서는 w의 크기를 5페이지 크기로 가정하였다. 이러한 조건에서 페이지 폴트가 일어나 교체해야 할 상황이 되면 가장 오래된 페이지들 중 윈도우의 크기인 5개의 B, C, D, E, F 페이지의 dirty 여부를 확인한다. 현재 LRU 위치에 있는 페이지는 F이고 dirty페이지로 교체가 발생하여 F를 교체하게 되면 쓰기 비용이 발생하게 되므로 clean페이지인 E를 교체하게 된다. 만약 B, C, D, E, F 페이지가 모두 dirty페이지일 경우에는 LRU 정책에 의해 가장 오래된 페이지를 제거하게 된다.

지연 쓰기로 인해 플래시 메모리의 쓰기와 지우기 작업의 수행 횟수를 감소시킬 수 있어 전체적인 성능은 향상된다. 하지만 자주 사용되지 않은 dirty페이지까지 장시간 보관할 가능성이 있기 때문에 적중률이 저하되고 w의 크기를 사용자가 정의하는 부분과 크기에 따라 성능이 달라지는 점은 고려할 필요가 있다.

2.2 LRU-DPL-CD

LRU-DPL-CD는 CF-LRU와 마찬가지로 LRU 정책에서 dirty페이지에 대한 지연 쓰기를 수행하는 정책이며 다른 점이 있다면 dirty, clean페이지를 구별하는 비트 이외에 cold flag를 사용한다.



(그림 2) LRU-DPL-CD 구조

(그림 2)에서 캐시의 전체적인 구조를 나타내고 있다. 이 정책에서의 가장 핵심적인 기능으로는 페이지가 dirty인지 clean인지를 구분하는 dirty bit와 지연이 수행이 되었는지 확인하는 cold flag를 사용하여 지연 쓰기를 한다. 페이지 교체가 이루어지게 되면 LRU 위치에서 교체 과정이 이루어진다. 먼저 dirty bit를 확인하여 clean일 경우 교체 대상이 되며, dirty일 경우 cold flag를 확인한다. 만약 cold flag가 설정되어 있지 않다면 cold flag를 설정하고 리스트의 처음 위치인 MRU 위치로 이동시킨다. 만약 cold flag가 설정되어 있다면 교체 대상이 된다. 이러한 방법을 통해 dirty페이지를 캐시 크기의 한 주기 동안 지연이 가능하다.

LRU-DPL-CD는 dirty페이지를 지연하기 위한 메모리 공간 사용이 적고, 쉽게 구현이 가능하다. 하지만 리스트의 모든 페이지들이 dirty페이지고 cold flag가 설정되어 있지 않은 worst case의 경우 리스트 안에서 모든 페이지들의 flag를 검사하고, 위치를 변경해야 하는 상황이 발생하기 때문에 수행속도가 $O(n^2)$으로 느리다. 또한 CF-LRU과 마찬가지로 지연 쓰기로 인해 적중률이 저하되기 때문에 worst case의 경우에 대한

대책과 적중률에 대하여 고려해야 한다.

2.3 FLRU

FLRU는 CF-LRU, LRU-DPL-CD와 마찬가지로 LRU 정책을 기반으로 변형된 캐시 정책으로 제안되었다. 이 정책은 NAND 플래시 메모리의 특성인 읽기, 쓰기, 지우기 비용이 서로 다르고 덮어쓰기를 할 경우 높은 오버헤드가 발생하는 부분을 고려하여 가중치를 적용한다. 또한 참조된 시간을 고려하여 참조되었던 시간들의 합과 평균을 계산하고 계산된 결과에 의해 각 페이지마다 가치(*C*)를 달리하는 방법을 사용한다. 교체가 되는 대상 페이지는 계산한 결과 중 *C*의 값이 가장 높은 페이지를 선택한다. (그림 3)에서의 (1)은 페이지에서 *C*를 구하는 방법을 수식화하여 나타내고 있다. t_c는 현재 시간, t_{fivag}는 참조된 시간들의 평균, f_i는 참조횟수, ERC(Expected Replacement Cost)는 페이지의 교체 예상 비용을 의미한다. ERC는 FTL(Flash Transition Layer)[6][7][8]라는 주소 변환 계층에서 플래시 메모리의 물리적 공간 상태를 파악하여, 읽기 연산을 기준으로 읽기/쓰기/덮어쓰기의 평균 수행 시간을 각각 비율로 나타내어 1/7/65의 가중치 값으로 표현된다.

(1) $$C = \left(\frac{t_c - t_{fiavg}}{f_i \times ERC} \right)$$

(2)

(그림 3) FLRU 페이지 가치와 시간

(그림 3)에서 (2)는 (1)에서 분자부분($t_c - t_{fivag}$)을 계산하는 예를 그림으로 표현하였다. 1에서 참조가 되고 2에서 다시 참조가 발생하여 총 2번의 참조가 되었다면, 1에서의 참조 시간과 2에서의 참조 시간을 더하여 총 참조 횟수인 2로 나눈 값을 t_{fivag}라고 한다.

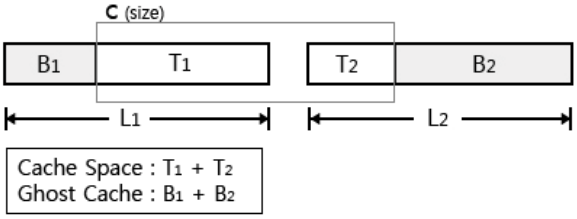
NAND 플래시 메모리에서의 덮어쓰기 연산은 지움 연산과 쓰기 연산이 동시에 이루어지기 때문에 연산 비용이 가장 비싸다. 이 점을 고려하여 dirty페이지에 대한 지연 쓰기를 하지만 dirty페이지의 덮어쓰기에 의한 지연이 캐시의 적중률을 저하시키는 원인이 될 우려가 있다. 그리고 페이지 교체가 발생할 때마다 참조된 시간을 모두 계산해야만 하고, 페이지를 교체하는데 예상되는 비용을 계산하기 위해 FTL에 접근해야 하기 때문에 수행 속도가 느리다. 또한 참조된 시간을 저장하는 공간과 참조횟수에 대한 공간 비용이 추가적으로 필요하게 되며, 캐시의 크기에 따라 수행속도 및 공간 비용은 더욱 늘어나게 되는 단점을 가져 이를 보완해야만 한다.

3. 플래시 메모리 기반의 효율적인 캐시 관리 정책 설계

본 논문에서는 NAND 플래시 메모리의 성능 향상을 위해 기존의 ARC(Adaptive Replacement Cache)[9] 정책을 개선한 DLD-ARC 교체 정책을 제안한다. 본 장에서는 기존의 ARC 정책에 대해 알아보고, 이를 NAND 플래시 메모리의 특성을 고려하여 설계한다. 그리고 제안하고자 하는 캐시 관리 정책에 대한 전체적인 특성을 기술한다.

3.1 ARC

ARC는 이전에 참조된 페이지의 정보를 이용하여 캐시 공간을 효율적으로 관리하는 정책으로 제안되었다. 이 알고리즘에는 한번 참조된 페이지들을 관리하는 리스트(L_1)와 두 번 이상 참조된 페이지들을 관리하는 리스트(L_2)가 존재하며, Ghost Cache 또는 Ghost Buffer[9]라고 불리는 공간에서 참조된 페이지의 리스트에 따라 두 리스트의 캐시 공간 비율이 점차 변화에 적응하는 캐시 교체 정책이다.



(그림 4) ARC 구조

(그림 4)는 ARC의 구조를 나타낸 모습으로 실질적인 Real Cache ($T_1 + T_2$)의 크기는 c이며 Ghost Cache ($B_1 + B_2$)을 합하면 전체 캐시 크기가 2c의 공간을 사용하는 더블 캐시이다. 페이지 교체가 발생하게 되면 캐시에서 교체되는 페이지를 바로 제거하지 않고 T_1에서 교체가 이루어지면 Ghost Cache 공간인 B_1으로, T_2에서 교체가 이루어지면 B_2로 이동한다. 이렇게 이동된 페이지 정보를 바탕으로 요구 페이지가 Ghost Cache공간에 존재하게 되면 앞으로 요구될 페이지들이 자신이 속한 리스트에서 참조될 가능성이 높다고 판단하여 리스트의 길이를 점차 증가시키고 다른 리스트인 Real Cache의 리스트 길이를 점차 감소시킨다. 한 예로 B_1에 요구 페이지가 존재한다면 T_1의 리스트 길이를 증가시키고 T_2의 리스트 길이는 감소시키게 된다. 캐시는 Ghost Cache를 사용함으로써 미래의 참조될 페이지들의 패턴 형태가 한번만 참조될 페이지인지 지속적으로 참조될 페이지인지를 구분하여 관리하게 되기 때문에 캐시의 성능에 유용하게 사용된다. 이러한 Ghost Cache를 사용하는 캐시 정책에는 2Q[10]와 적응형 LRFU(적응형 Least Recently Used and Least Frequently Used)[11]가 있다.

ARC 정책만으로는 NAND 플래시 메모리에서의 쓰기

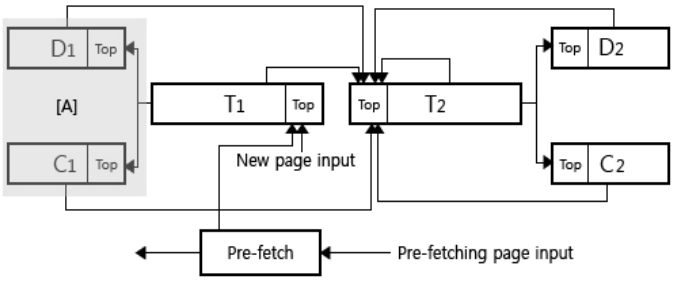
비용을 고려하지 않기 때문에 NAND 플래시 메모리 기반 캐시 정책으로 사용하기에 적합하지 않다. 이에 NAND 플래시 메모리의 특성을 고려하여 DLD-ARC 정책을 설계한다.

3.2 DLD-ARC (Double List for Dirty page - ARC)

DLD-ARC 정책은 NAND 플래시 메모리의 특성에 맞게 지연 쓰기를 고려해야 하며, 기존의 ARC 정책의 특성을 유지해야 한다. 따라서 Real Cache에서의 지연 쓰기를 하지 않고 Ghost Cache 공간을 활용하여 dirty페이지에 대한 지연 쓰기를 하여 적중률을 최대한 유지하고, dirty페이지를 효율적으로 관리해야 한다. 또한 dirty페이지를 어떤 방식으로 처리할지를 고려해야 한다.

3.2.1 clean리스트와 dirty리스트

DLD-ARC는 L_1과 L_2리스트로 존재하며 다시 L_1 리스트는 T_1과 B_1로, L_2는 T_2와 B_2로 구분한다. 그리고 dirty페이지를 효율적으로 관리하기 위한 목적으로 Ghost Cache 공간인 B_1를 dirty페이지가 유지되는 리스트 D_1과 clean페이지가 유지되는 리스트 C_1으로 구분한다. 같은 방식으로 B_2를 D_2와 C_2로 구분한다.

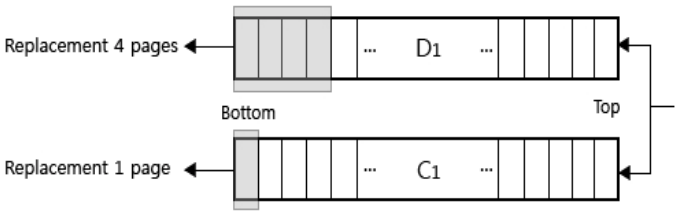


(그림 5) DLD-ARC 구조

NAND 플래시 메모리의 구조상 파일을 순차적으로 읽기, 쓰기 연산을 할 경우 임의 읽기, 쓰기 연산 보다 성능이 20~30% 좋아진다[12]. 이러한 이유로 읽기 연산 시에는 순차적인 선반입 정책을 사용하여 읽기 성능을 높일 수 있다. (그림 5)는 DLD-ARC에서 사용하는 캐시의 구조를 나타내며 각 화살표들은 페이지의 이동경로를 나타낸다. 새로운 요구 페이지가 캐시로 진입하면 T_1의 top위치로 진입하게 된다. 요구 페이지에 의해 선반입 된 페이지들은 하나의 리스트(Pre-fetch)로 관리하여 페이지가 잘못 선반입 되었을 경우 캐시에서 빠르게 제거함으로써 공간 낭비를 줄일 수 있도록 한다. Real Cache 공간이 full인 상황이 되면 T_1이나 T_2의 마지막 위치에서 Ghost Cache 공간으로 이동하게 된다. 이때 dirty bit를 확인하여 dirty페이지는 dirty리스트의 top위치로 clean페이지는 clean리스트의 top위치로 이동한다. 요구 페이지가 발생하면 모든 리스트를 탐색하게 되는데 이때 Pre-fetch리스트에서 요구 페이지의 찾으면 T_1의 top위치로 이동하고, T_1, T_2, C_1, C_2, D_1, D_2리스트에서 요구 페이지를 찾으면 두 번 이상 참조되었기 때문에 T_2의 top위치로 이동한다.

DLD-ARC는 dirty리스트와 clean리스트로 관리하기 위해서 페이지를 식별할 수 있는 정보가 필요하다. 이 정보를 유지하기 위해 LRU-DPL-CD정책에서와 같이 dirty bit를 사용한다. 그러나 LRU-DPL-CD에서의 worst case와 같은 경우가 발생하는 문제점을 개선하기 위해 페이지의 이동을 없애고 dirty counter라는 정보를 유지하도록 한다. dirty counter는 L_1과 L_2에 각각 존재하며 초기 값은 '0'이고 페이지 교체가 이루어지는 리스트에 지연 쓰기가 발생할 때마다 하나씩 증가한다. dirty counter는 (0 ≤ dirty counter ≤ cache size)의 범위를 가지며, 최대 cache size와 같은 값을 가질 수 있다. 만약 cache size와 같은 값을 가지게 되면 해당 dirty리스트의 LRU 페이지를 교체 대상 페이지로 선택한다. 이는 dirty페이지가 Ghost Cache 공간으로 이동 후 캐시 크기의 한 주기 동안 다시 참조 되지 않으면 앞으로 사용할 가능성이 낮다는 점을 고려하였다. DLD-ARC의 지연 시간은 LRU-DPL-CD의 지연 시간과 동일한 지연 시간을 가진다. 만약 dirty페이지에서 교체가 이루어져 쓰기 연산이 발생하게 되면 dirty counter는 다시 '0'으로 초기화 된다. 이런 방식을 사용하면 dirty가 된 페이지의 이동 시간이 없어지기 때문에 worst case를 방지할 수 있다.

DLD-ARC는 온라인 방식으로 수행 과정에서 요구 페이지의 패턴에 따라 점차 변화하는 적응형 캐시 정책인 ARC 정책의 특성을 그대로 유지한다. 따라서 CF-LRU에서의 오프라인 방식으로 w을 결정하여 사용하는 정책보다 다양한 환경에서 바로 적용할 수 있는 이점을 가진다. 또한 LRU-DPL-CD보다 공간 비용이 다소 높지만 수행속도가 $O(n^2)$되는 worst case를 $O(n)$로 유지하여 수행 속도를 개선할 수 있으며 페이지 교체가 발생할 때마다 모든 페이지에 대한 C를 계산 하는 방식의 FLRU와 비교하여 공간적으로나 시간적으로 오버헤드가 적은 특성을 가진다.



(그림 6) Ghost Cache 공간 A에서 페이지 교체의 예

3.2.2 교체된 페이지 쓰기 연산

NAND 플래시 메모리의 쓰기 성능은 순차적으로 쓰기 연산을 할 경우 가장 성능이 좋아진다[12]. (그림 6)에서와 같이 clean리스트 에서의 교체가 발생하면 쓰기 연산이 수행되지 않기 때문에 한 페이지를

교체하고, dirty리스트에서 페이지 교체가 발생하는 상황이 되면 dirty페이지를 한 블록 크기인 4개의 페이지가 교체되어 쓰기 연산을 수행한다. 이로 인해 한 블록 안에서 사용되지 않는 공간 낭비와 지움 횟수를 줄일 수 있게 된다. (그림 6)은 (그림 5)에서의 A공간을 한 블록이 4개의 페이지로 구성되어 있는 것으로 예를 들어 표현하였다. dirty와 clean페이지로 구분하여 관리하게 되면 리스트에서 dirty페이지 확인, 탐색을 필요로 하지 않게 되어 dirty페이지에서의 블록 단위 쓰기 기법을 사용하는데 큰 이점을 가진다.

4. 결론 및 향후 계획

본 논문에서는 NAND 플래시 메모리 기반 파일 시스템을 위하여 ARC 정책을 개선한 DLD-ARC 정책을 설계하였다. ARC 정책을 모델로 개선한 이유는 더블 캐시의 Ghost Cache 공간을 활용하면 적중률의 감소를 줄일 수 있기 때문이다. 더블 캐시의 Ghost Cache에서 지연 쓰기를 하기 위하여 dirty페이지와 clean페이지를 다른 리스트로 구분하고, dirty bit와 dirty counter를 이용하여 dirty페이지들을 효율적으로 관리할 수 있도록 하였다. 또한 NAND 플래시 메모리의 특성인 쓰기 성능을 고려하여 dirty페이지를 블록단위로 빠르게 쓰기 연산이 수행되도록 하였다.

향후 DLD-ARC 알고리즘을 기존에 제안된 플래시 메모리 기반의 캐시 정책들과 비교하여 다양한 환경을 토대로 테스트를 수행할 계획이며 더블 캐시를 이용하는 다른 캐시 정책에도 NAND 플래시 메모리의 특성에 맞게 dirty페이지와 clean페이지 관리 기법을 적용하여 테스트를 수행할 계획이다.

참 고 문 헌

[1] Needham & Company, LCC, “NAND vs. Hard Disk Drives: Hype, Myth and Reality,” October, 2005.

[2] Greg Gagne, Abraham Silberschatz, Peter Baer Galvin, “Operating System Concepts sixth edition,” Wiley, 2003

[3] Chanik Park, Jeong-Uk Kang, Seon-Yeong Park, Jin-Soo Kim, “Energy-aware demand paging on NAND flash-based embedded storages,” Proceedings of the 2004 international symposium on Low power electronics and design table of contents, pp.338-343, 2004.

[4] 정호영, 박성민, 차재혁, 강수용, “플래시 메모리를 위한 Not-cold-page 쓰기지연을 통한 LRU 버퍼교체 정책 개선,” 정보과학회논문지 : 시스템 및 이론 제 33권 제 9호, pp. 634-641, 2006

[5] 박종민, 박동주, “플래시 메모리상에서 시스템 소프트웨어의 효율적인 버퍼 페이지 교체 기법,” 정보과학회논문지 : 데이터베이스 제 34권 제 2호, pp. 133-140, 2007

[6] Li-Pin Chang, Tei-Wei Kuo, “An Efficient Management Scheme for Large-Scale Flash-Memory Storage Systems,” In Proceedings of the ACM Symposium on Applied Computing (SAC), pp. 862-868, 2004.

[7] Jesung Kim, Jong Min Kim, S. H. Noh, Sang Lyul Min, Yookun Cho, “A Space-Efficient Flash Translation Layer for CompactFlash Systems,” IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, pp. 366-375, 2002.

[8] Michael Wu, Willy Zwaenepoel “eNVy: a Non-Volatile, Main Memory Storage System,” In Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 86-97, 1994.

[9] N. Megiddo, D.S. Modha, “ARC: A Self-Tuning, Low Overhead Replacement Cache,” Proc. Usenix Conf. File and Storage Technologies, Usenix, pp 115-130, 2003.

[10] T. Johnson and D. Shasha, “2Q: A low overhead high performance buffer management replacement algorithm,” in Proc. VLDB Conf., pp. 297-306, 1994.

[11] 이종민, 현철승, 이동희, “적응형 LRFU 블록 교체 정책의 설계,” 한국컴퓨터종합학술대회논문집, Vol. 34, No. 1(A), 2007

[12] CommDesign, “Flash memory 101: An Introduction to NAND Flash,” 2006