사물인터넷융복합논문지 Vol. 8, No. 5, pp. 77-83, 2022 https://doi.org/10.20465/KIOTS.2022.8.5.077

낸드 플래시 메모리 시스템 기반의 지속성을 고려한 핫 데이터 식별 경량 기법

이승우*
영남이공대학교 소프트웨어콘텐츠계열 교수

A lightweight technique for hot data identification considering the continuity of a Nand flash memory system

Seungwoo Lee*
Professor, Division of Mechanical Engineering Technology, Yeungnam University College

요 약 낸드 플래시 메모리는 구조적으로 쓰기 전 지우기(Erase-Before-Write) 동작이 요구된다. 이것을 해결하기 위해서는 데이터 업데이트 동작이 빈번히 발생하는 페이지(Hot data page)를 구분하여 별도에 블록에 저장함으로 해결할 수 있으며 이러한 Hot data를 분류하는 기법을 핫 데이터 판단기법이라 한다. MHF(Multi Hash Function Framework)기법은 데이터 갱신요청의 빈도를 시스템 메모리에 기록하고 그 기록된 값이 일정 기준 이상일 때 해당 데이터 갱신요청을 Hot data로 판단한다. 하지만 데이터 갱신요청에 빈도만을 단순히 카운트하는 방법으로는 정확한 Hot data로 판단에 한계가 있다. 또한 데이터 갱신요청의 지속성을 판단 기준으로 하는 기법의 경우 갱신요청 사실을 시간 간격을 기준으로 순차적으로 기록한 뒤 Hot data로 판단하는 방법이다. 이러한 지속성을 기준으로 하는 방법의 경우 그 구현과 운용이 복잡한 단점이 있으며 갱신요청에 빈도를 고려하지 않는 경우 부정확하게 판단되는 문제가 있다. 본 논문은 데이터 갱신요청에 빈도와 지속성을 함께 고려한 경량화된 핫 데이터 판단기법을 제안한다.

주제어 : Nand Flash Memory, FTL, Hot Data Identification, Garbage Collection, Mapping Algorithm

Abstract Nand flash memory requires an Erase-Before-Write operation structurally. In order to solve this problem, it can be solved by classifying a page (hot data page) where data update operation occurs frequently and storing it in a separate block. The MHF (Multi Hash Function Framework) technique records the frequency of data update requests in the system memory, and when the recorded value exceeds a certain standard, the data update request is judged as hot data. However, the method of simply counting only the frequency of the data update request has a limit in judging it as accurate hot data. In addition, in the case of a technique that determines the persistence of a data update request, the fact of the update request is recorded sequentially based on a time interval and then judged as hot data. In the case of such a persistence-based method, its implementation and operation are complicated, and there is a problem of inaccurate judgment if frequency is not considered in the update request. This paper proposes a lightweight hot data determination technique that considers both frequency and persistence in data update requests.

Key Words : Nand Flash Memory, FTL, Hot Data Identification, Garbage Collection, Mapping Algorithm

*교신저자 : 이승우(zpa007@gmail.com)
접수일 2022년 7월 24일 수정일 2022년 9월 2일 심사완료일 2022년 9월 11일

1. 서론

낸드 플래시 메모리는 기타 저장장치와 비교해 더 빠른 데이터 입출력 속도와 저전력 등 여러 장점이 있다[1-2]. 하지만 낸드 플래시 메모리는 하드웨어 특성으로 인해 발생하는 몇 가지 문제점을 가지고 있으며 이를 해결하기 위해 일반적으로 FTL(Flash Translation Layer)이 사용된다[3-6]. FTL은 낸드 플래시 메모리에 데이터 업데이트 동작 시 제자리 덮어쓰기(in-place updates)가 불가능하여 추가적인 쓰기 전 지우기(erase-before-write) 동작이 발생하는 것을 해결하는 것이 핵심이며 이를 위해 데이터 업데이트 동작이 빈번히 발생하는 페이지(Hot data page)와 상대적으로 그렇지 않은 페이지(Cold data page)를 구분하여 별도에 블록에 저장할 필요가 있다. 이러한 Hot data를 분류하는 기법을 핫 데이터 판단기법이라 한다.

현재까지 알려진 기법으로는 쓰기 데이터 갱신요청의 빈도를 기준으로 Hot data를 판단하는 MHF(Multi Hash Function Framework) 기법과 데이터 갱신요청에 지속 여부 기준으로 판단하는 기법 등이 있다[7-11].

MHF기법은 데이터 갱신요청의 빈도를 시스템 메모리에 기록하고 그 기록된 값이 일정 기준 이상일 때 해당 데이터 갱신요청을 Hot data로 판단한다. 하지만 데이터 갱신요청에 빈도만을 단순히 카운트하는 방법으로는 정확한 Hot data로 판단에 한계가 있다. 또한 데이터 갱신요청의 지속성을 판단 기준으로 하는 기법의 경우 갱신요청 사실을 시간 간격을 기준으로 순차적으로 기록한 뒤 Hot data로 판단하는 방법이다. 이러한 지속성을 기준으로 하는 방법의 경우 그 구현과 운용이 복잡한 단점이 있으며 갱신요청에 빈도를 고려하지 않는 경우 부정확하게 판단되는 문제가 있다.

본 논문은 데이터 갱신요청에 빈도와 지속성을 함께 고려한 경량화된 Hot data 판단기법을 제안한다.

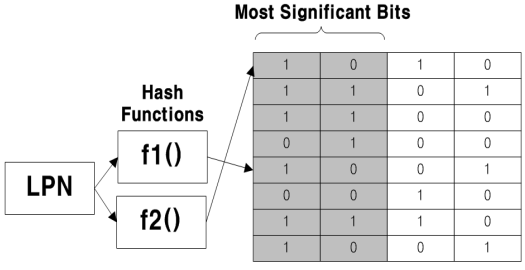
2. 관련 연구

2.1 낸드 플래시 메모리의 동작 특징

낸드 플래시 메모리의 데이터 읽기와 쓰기에 단위는 페이지 단위이며 삭제 단위는 블록 단위이다. 낸드 플래시 메모리는 구조적 특징으로 인해 데이터 갱신요청 시 덮어쓰기가 불가능한 특징이 있다. 이를 해결하는 가장 간단한 방법은 기존 데이터가 기록된 페이지를 포함한 블록을 삭제하지 않고 비어있는 페이지에 즉시 기록하고 이전 페이지는 무효(invalid) 상태로 표시한다. 이러한 무효화 페이지가 점차 늘어나는 문제는 시스템에서 적절한 시점에 가비지 컬렉션에 의해 삭제 연산이 실행되며 해결되지만 빈 페이지가 부족할 경우 블록삭제 후 쓰기 동작에 오버헤드가 크다. 이러한 무효페이지 발생을 최소화하기 위해서는 데이터 갱신을 자주 발생시키는 갱신요청을 특정하여 별도 처리할 수 있으며 이를 위해 핫 데이터 판단기법이 필수적으로 필요하다.

2.2 핫 데이터 식별 기술의 문제점

Hot data란 특정 페이지를 대상으로 데이터 갱신요청이 상대적으로 자주 발생하는 것을 말하고 또한 상대적으로 그렇지 않은 경우를 cold data라고 한다[12-15]. 블록에 Hot data 페이지로 인해 빈번한 업데이트 동작 발생으로 인한 무효페이지 증가 및 부가적인 오버헤드가 발생하게 된다. 이러한 문제를 해결하기 위해서는 데이터 갱신요청 패턴을 분석하여 잦은 갱신요청에 Hot data를 판단하여 cold data와 별도 저장하여 관리함으로 해결할 수 있다. 이와 관련된 많은 연구가 진행되었고 대표적인 기법으로 멀티해시함수를 이용한 MHF 기법이 있다. 이 기법은 하나 이상의 해시함수를 이용하여 데이터 갱신요청 된 페이지의 주소값을 해싱하고 해싱값과 대응하는 메모리에 갱신요청을 누적 기록한 뒤 Hot data 판단을 한다.



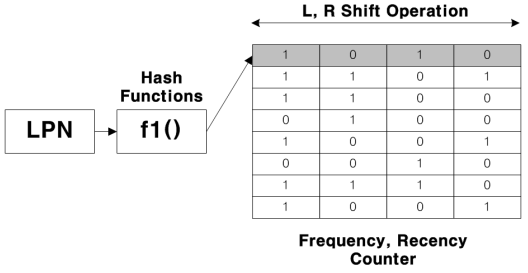
[Fig. 1] Multi hash function framework

[그림. 1]은 MHF의 전체 동작 방식이다. 플래시 변환계층으로부터 특정 논리페이지를 대상으로 데이터 갱신이 요청되면 해당 논리페이지 주소값을 해싱하고 결과값과 매칭된 특정 메모리에 bit값을 1bit씩 증가시킨다. 이후 메모리에 기록된 값을 기준으로 Hot data 판단을 진행한다. 기록된 값 중 상위 2bit에 기록된 값이 0이 아니면 해당 논리페이지 주소에 기록된 데이터를 Hot data로 판단하는 구조이다.

MHF 기법은 알고리즘이 단순하여 운영의 오버헤드가 적은 장점이 있으나 Hot data 판단 시 단순히 데이터 갱신요청에 빈도만을 고려하기 때문에 특정 시간 동안 집중적인 데이터 갱신요청이 발생한 뒤 더 이상 데이터 갱신요청이 발생하지 않은 경우에도 이를 Hot data로 판단하는 단점이 있다.

3. 제안기법

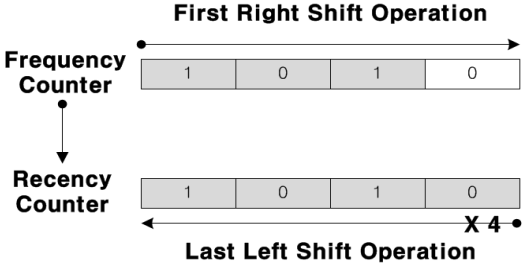
3.1 제안기법 구조



[Fig. 2] Our framework and it's operations

제안기법의 전체 구조는 [그림. 2]와 같다. 데이터 갱신요청이 발생했을 때 해당 LPN 값을 해싱하기 위한 해시함수와 해싱 값에 대응하는 4bit 메모리로 이루어진 전체 메모리 테이블로 구성된다.

제안기법의 동작 과정은 다음과 같다. 최초 데이터 갱신요청 시 빈도를 기록하기 위해 해당 LPN을 해시함수 H1을 이용하여 해싱하고 해당 값에 대응하는 4bit 메모리에 첫째 비트에 값을 1 증가시킨다. 해당 4bit 메모리는 기본적으로 설정된 시간 단위로 right shift 동작하게 된다. 만약 4bit 메모리에 가장 왼쪽 비트에 값이 1이 되는 경우 [그림. 3]처럼 해당 데이터 갱신요청에 지속성을



[Fig. 3] Our process when shift operation

기록하기 위해 동작 방식을 변경한다. 즉 갱신요청에 지속성을 기록하는 4bit 메모리는 기존 갱신요청에 빈도를 기록한 4bit 메모리를 이용하기 때문에 추가적인 메모리 사용량을 감소시킬 수 있다.

지속성을 기록하기 위한 동작 방식으로 전환된 4bit 메모리는 설정된 시간 단위로 left shift 방식으로 동작한다. 이때 4bit 메모리에 첫째 비트에 left shift 동작에 설정된 시간을 4배 빠르게 변경한다. 데이터 갱신요청에 지속성을 측정하기 위한 4bit 메모리는 4차례 left shift 이후 다시 빈도 측정을 동작 방식으로 전환되고 이때 데이터 갱신요청에 대한 Hot data 여부를 판단하는 과정이 시작된다.

3.2 데이터 갱신요청의 빈도

```
Algorithm 1. Frequency of write operation(lpn).
Input: lpn
Output: none
1 k := the number of hash functions
2 threshold := threshold value for
Hot data identification
3 for i=1 to k
4    entry=hashing[i] (lpn)
5    increase hashing_table[entry] by 1
6 end for
7 left shift operation
end
```

제안기법은 특정 데이터 갱신요청에 빈도를 기록하기 위해 Algorithm 1.과 같이 갱신요청 시 해당 LPN 주소를 해시함수를 이용하여 해싱하고 그 값과 매칭되는 4bit 메모리에 첫 번째 bit에 값을 1로 변경한다. 4bit 메모리는 right shift 동작한다. 더 이상 갱신요청이 없는 경우 기록된 값은 시간이 지남에 따라 4bit 모두 0으로 초기화된다. 또한 4bit 메모리에 첫 번째 bit는 설정된 시간 간격으로 자동 초기화된다.

이러한 동작은 데이터 갱신요청에 빈도를 측정하기 위함이며 초기화 시간보다 더 빠르게 갱신요청이 발생하는 경우 이를 두 번째 bit에 값을 1로 변경하여 갱신요청에 발생빈도를 기록한다. 즉 데이터 갱신요청에 빈도를 기록하는 4bit 메모리에 첫 번째 bit의 기록값을 설정된 시간 간격으로 초기화하여 단순 갱신요청에 대한 부가 동작을 방지할 수 있으며 설정 시간보다 더 빠르게 갱신이 요청될 때 해당 갱신요청에 빈도를 나머지 bit에 누적 기록하게 된다.

데이터 갱신요청에 빈도를 단순 기록하여 특정 갱신요청을 Hot data로 판단하는 것은 충분하지 않다. 특히 특정 시간 구간에 집중적인 데이터 갱신요청이 발생한 뒤 더 이상 갱신이 발생하지 않은 경우에도 갱신요청에 빈도만을 고려할 경우는 Hot data로 판단되는 문제가 발생한다. 이러한 문제점을 해결하기 위해 본 논문에서는 데이터 갱신요청의 발생빈도를 카운터하고 그 결과를 미리 정한 시간 단위로 별도의 메모리에 누적 기록하여 이를 바탕으로 Hot data 여부를 판단하는 알고리즘을 제안한다.

3.3 데이터 갱신요청의 지속성

제안기법은 더 정확한 Hot data 판단을 위해 데이터 갱신요청에 빈도와 더불어 해당 갱신요청에 지속성을 함께 고려하기 위해 Algorithm 2.와 같이 갱신요청에 빈도를 기록한 4bit 메모리에 4번째 bit의 값이 1이 되는 순간 동일 4bit 메모리의 작동 방식을 변경하여 지속성 여부를 기록하게 된다. 지속성 기록을 위한 동작 방식은 최초 4bit 메모리에 bit 값을 모두 0으로 초기화되며 shift 방향은 left로 변경된다. 또한 shift 횟수 또한 4배 빠르게 조정하게 된다. shift 횟수를 4배 빠르게 조정함으로써 갱신요청에 빈도를 별도 메모리에 기록할 필요 없이 동일 메모리에 지속성 여부를 기록할 수 있는 장점이 있다. 데이터 갱신요청에 빈도를 4배 빠르게 shift 처리하여 빈도가 낮아진다면 지속성 역시 낮아지는 결과로 판단할 수 있다. 이후 left shift 동작이 종료되면 Hot data page 여부를 판단하기 위한 Hot data 판단과정을 시작한다.

Algorithm 2. *Continuity Counter(lpn).*

```
Input: lpn
Output:none
1 k := the number of hash functions
2 threshold := threshold value for
  Hot data identification
3 for i=1 to k
4    entry=hashing[i] (lpn)
5    if 4bit value 1
6       right shift operation, x4 time rate
7 end for
end
```

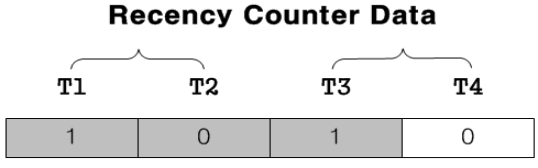
3.4 제안기법의 핫 데이터 판단과정

제안기법은 Hot data 판단을 위해 특정 데이터 갱신요청에 빈도와 지속을 함께 고려한다. 이를 위해 앞서 설명한 것처럼 4bit 메모리에 빈도를 우선 기록한 뒤 이후 동일 4bit 메모리에 shift 방향과 속도를 변경하여 지속성을 기록하였다. 제안기법은 갱신요청에 빈도를 기록한 4bit 메모리에 4번째 bit에 값에 따라 지속성을 기록하기 위한 동작 방식을 변경하기 때문에 갱신요청에 발생 빈도가 낮은 경우 지속성 측정을 생략함으로 부가적인 연산을 줄일 수 있다. 반면 갱신요청 빈도를 기록한 4bit 메모리에 4번재 bit값이 1로 변경되면 해당 갱신요청은 충분한 빈도로 요청이 발생했음을 의미함으로 지속성을 기록한 뒤 Hot data 판단과정을 시작한다.

제안기법은 데이터 갱신요청에 지속성을 판단하기 위해 4bit 메모리에 상위 2bit와 하위 2bit의 값이 모두 1일 때 Hot data로 판단한다.

[그림. 4]와 같이 빈도 측정값을 누적 기록하기 위해 각 bit를 T1, T2, T3, T4로 구분하였다. T1의 기록된 값은 T2의 기록된 값보다 지속성 측면에서 더 높은 가중치를 가진다. 왜냐하면 지속성이란 갱신요청에 빈도가 오래 지속되었는지를 의미하기 때문에 더 이전에 빈도 측정값에 대한 연속성이 기록된 T1의 값은 T2의 값보다 갱신요청을 기록한 시점이 더 앞서기 때문이다. 또한 상위 2bit와 하위 2bit를 구분하여 평가함으로써 갱신요청의 짧은 지속성과 긴 지속성을 판단할 수 있다.

이를 위해 [그림. 4]와 같이 T1과 T2를 하나의 긴 연속적인 시간 구간으로 설정하고 T3와 T4를 하나의 짧은 연속적인 시간 구간으로 설정한다. 예를 들어 4bit 메모리에 값이 1 1 0 0 이라면 이러한 데이터 갱신요청은 긴 지속성 구간을 기준으로 쓰기연산 요청이 발생하였고 이후 데이터 갱신요청이 짧은 연속적인 구간에 지속적으로 발생하지 않는 갱신패턴으로 판단하여 핫 데이터로 판단하지 않는다. 또한 4bit 메모리에 값이 1 0 1 0 이라면 이러한 데이터 갱신요청은 연속하는 지속성 구간에 모두 데이터 갱신이 지속됨으로 핫 데이터로 판단한다.



[Fig. 4] Duration measurement data

4. 실험 결과

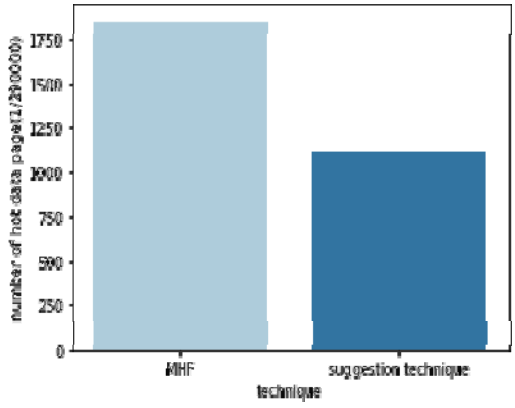
4.1 트레이스 파일

MHF 기법과 제안기법을 각 시뮬레이션 프로그램으로 구현하고 일반적인 PC 사용 환경에서 발생하는 워크로드를 기록한 트레이스 파일을 대상으로 데이터 갱신요청에 대한 Hot data 발생비율과 낸드플래시메모리 블록 내 무효화 페이지 발생비율에 대해 실험하고 그 결과를 분석한다. 실험에 사용된 트레이스 파일로 [표. 1]과 같이 데이터 갱신요청 횟수는 약 약 290,000회이다. 본 실험에서 데이터 갱신요청 이외에 부가적인 연산에 대해서는 고려하지 않는다.

〈Table 1〉 Trace file characteristics

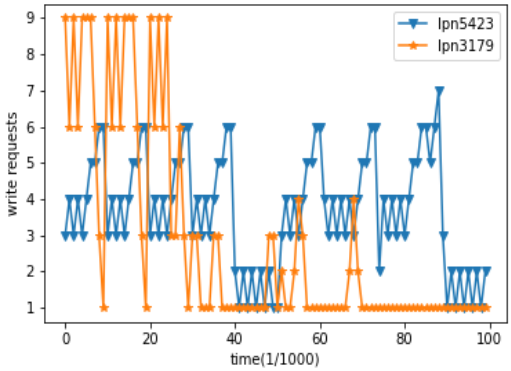
Properties	Values
Number Of Write Requests	290,400
Minimum Offset	8
Max Offset	326,960
Minimum Size	32
Max Size	4,096

4.2 핫 데이터 판단 비율



[Fig. 5] Number of hot data pages

[그림. 5]에서 MHF 기법 적용 시 데이터 갱신요청 중 Hot data 판단 비율은 전체 데이터 갱신요청 약 290,400회 중 약 0.6%이고 제안기법 적용 시 0.3%이다. 실험환경에 사용된 트레이스 파일에 따른 결과로 전체적으로 제안기법이 MHF 기법보다 데이터 갱신요청에 대한 Hot data 판단 비율이 낮음을 알 수 있다. 이는 앞서 설명한 것처럼 제안기법은 데이터 갱신요청에 단순 빈도만을 고려하지 않고 지속성을 함께 평가하기 때문이며 특정 시간 집중적인 갱신요청이 발생한 이후 더 이상 갱신요청이 발생하지 않는 갱신요청을 Hot data로 판단하지 않은 결과이다.



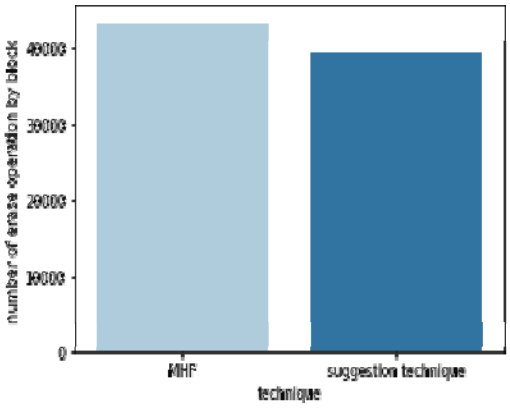
[Fig. 6] Example of hot data judgment in update request

[그림. 6]에서 lpn3179에 경우 짧은 시간 동안 갱신요청이 발생한 이후 더 이상 갱신요청이 발생하지 않는다. 이러한 갱신요청 패턴에 대해 단순 빈도만을 기준으로 판단하는 경우 단순 갱신요청을 핫 데이터로 판단하여 추가적인 시스템 메모리 사용량이 증가하는 문제점이 발생한다. 제안기법은 갱신요청에 빈도와 지속성 여부를 함께 판단하기 때문에 lpn3179에 경우 단순 갱신요청으로 판단하게 된다. 또한 lpn5423에 경우 갱신요청에 빈도가 낮아 핫 데이터로 판단되지 않을 수 있으나 제안기법은 짧은 빈도에 지속적인 갱신요청을 핫 데이터로 판단하게 된다.

4.3 블록 단위 삭제 횟수

[그림. 7]은 MHF 기법과 제안기법을 각각 적용했을 때 블록 삭제 횟수를 보여준다. MHF 기법의 경우 블록 삭제는 약 4300회이고 제안기법의 경우 약 3900회이다. 제안기법 적용 시 블록 단위 삭제 횟수가 더 낮은 것은 데이터 갱신요청에 빈도와 함께 지속성을 측정하여 Hot data를 더 정확히 판단하고 해당 데이터 갱신요청에 LPN을 별도 블록에 저장함으로 블록 내 무효화 페이지 발생 수가 감소했기 때문이며 이로 인해 블록에 대한 삭제 횟수가 감소하였기 때문이다.

반면 MHF 기법은 Hot data 판단 비율은 제안기법보다 더 높지만 정확한 Hot data를 판단하지 못함으로 데이터 갱신요청을 자주 요청하는 특정 LPN으로 인해 블록 내 무효페이지 발생비율이 더 높게 발생하였음을 알 수 있다.



[Fig. 7] Number of eraser operation by block

5. 결론

본 논문에서는 데이터 갱신요청에 대한 효율적인 핫 데이터 판단을 위해 데이터 갱신요청의 빈도와 지속성을 함께 고려하고 갱신요청에 빈도를 기록한 메모리에 동작 방식을 변경하여 지속성을 기록하는 경량 핫 데이터 판단기법을 제안하였다. 실험결과 기존기법은 데이터 갱신요청에 빈도만을 고려하기 때문에 핫 데이터로 판단된 횟수는 상대적으로 많으나 지속적인 데이터 갱신요청을 Hot data로 판단하지 못하여 제안기법 보다 블록 단위 삭제가 더 많이 발생하였다. 결론적으로 제안기법은 핫 데이터 판단 횟수는 적지만 오히려 블록삭제 횟수는 감소하였고 이는 제안한 핫 데이터 판단기법이 더 정확한 핫 데이터를 판단하여 무효페이지 발생을 최소화함으로 결과적으로 블록단위 삭제 횟수를 줄일 수 있었다.

REFERENCES

[1] Y.Takai, M.Fukuchi, R.Kinoshita, C.Matsui, and K.Takeuchi, "Analysis on Heterogeneous SSD Configuration with Quadruple-Level (QLC) NAND Flash Memory," 2019 IEEE 11th International Memory Workshop (IMW), pp.1-4, 2019.

[2] D.Ma, J.Feng, and G.Li, "A Survey of Address Translation Technologies for Flash Memories," ACM Computing Surveys (CSUR), Vol.46, No.36, pp.1-39, 2014.

[3] S.Jiang, L.Zhang, X.Yuan, H.Hu, and Y.Chen, "S-ftl: An efficient address translation for flash memory by exploiting spatial locality," IEEE/NASA Goddard Conference on Mass Storage Systems and Technologies (MSST), pp.1-12, 2011.

[4] J.Kim, J.M.Kim, S.H.Noh, S.L.Min, and Y.K.Cho, "A space-efficient flash translation layer for compact flash systems," IEEE Transactions on Consumer Electronics, Vol.48, Issue.2, pp.366-375, 2002.

[5] S.W.Lee, D.J.Park, T.S.Chung, D.H.Lee, S.W.Park, and H.J.Song, "A log buffer-based flash translation layer using fully-associative sector translation," ACM Transactions on Embedded Computing Systems, Vol.6, No.3, pp.18, 2007.

[6] D.W.Jung, J.U.Kang, H.S.Jo, J.S.Kim, and J.W.Lee, "Superblock FTL: A superblock-based flash translation layer with a hybrid address translation scheme," ACM Transactions on Embedded Computing Systems, Vol.9, Issue.4, 2010.

[7] J.Liu, S.Chen, T.Wu, and H.Zhang, "A Novel Hot Data Identification Mechanism for NAND Flash Memory," IIEEE Transactions on Consumer Electronics, Vol.61, Issue.4, pp.463-469, 2015.

[8] J.W.Hsieh, T.W.Kuo, and L.P.Chang, "Efficient identification of hot data for flash memory storage systems," ACM Transactions on Storage (TOS), Vol.2, Issue.1, pp.22-40, 2006.

[9] H.S.Lee, H.S.Yun, and D.H.Lee, "HFTL:Hybrid Flash Translation Layer based on Hot Data Identification for Flach Memory," IEEE Transactions on Consumer Electronics, Vol.55, Issue.4, pp.2005-2011, 2009.

[10] S.O.Park, and S.J.Kim, "An efficient file system for large-capacity storage with multiple NAND flash memories," 2011 IEEE International Conference on Consumer Electronics (ICCE), pp.399-400, 2011.

[11] Y.J.Lee, H.W.Kim, H.J.Kim, T.Y.Huh, S.H.Jung, and Y.H.Song, "Adaptive Mapping Information Management Scheme for High Performance Large Sale Flash Memory Storages," Journal of the Institute of Electronics and Information Engineers, Vol.50, Issue.3, pp.78-87, 2013.

[12] D.C.Park, and David H.C.D, "Hot data identification for flash-based storage systems using multiple bloom filters," 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies(MSST), pp.1-12, 2011.

[13] L.P.Chang, "On efficient wear leveling for large-scale flash-memory storage systems," SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, pp.1126-1130, 2007.

[14] S.H.Jung, Y.S.Lee, and Y.H.Song, "A process-aware hot/cold identification scheme for flash memory storage systems," IEEE Transactions on Consumer Electronics, Vol.56, Issue.2, pp.339-347, 2010.

[15] K.W.Kim, S.H.Jung, and Y.H.Song, "Compression ratio based hot/cold data identification for flash memory," 2011 IEEE International Conference on Consumer Electronics (ICCE), pp.33-34, 2011.

이 승 우(Seung-Woo Lee) [정회원]

- 2013년 2월 : 경북대학교 IT대학 컴퓨터공학 (공학석사)
- 2020년 8월 : 경북대학교 IT대학 컴퓨터공학 (공학박사)
- 2020년 3월 ~ 2021년 2월 : 경운대학교 연구교수
- 2021년 3월 ~ 현재 : 영남이공대학교 소프트웨어콘텐츠 계열 조교수

〈관심분야〉
임베디드 시스템, Nand Flash Memory