사물인터넷융복합논문지 Vol. 8, No. 5, pp. 33-39, 2022 https://doi.org/10.20465/KIOTS.2022.8.5.033

데이터 갱신요청의 연속성과 빈도를 고려한 개선된 핫 데이터 검증기법

이승우*
영남이공대학교 소프트웨어콘텐츠계열 교수

Improved Hot data verification considering the continuity and frequency of data update requests

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

요 약 모바일 컴퓨팅 분야에서 사용되는 저장장치는 저전력, 경량화, 내구성 등을 갖추어야 하며 사용자에 의해 생성되는 대용량 데이터를 효과적으로 저장 및 관리할 수 있어야 한다. 낸드 플래시 메모리는 모바일 컴퓨팅 분야에서 저장장치로 주로 사용되고 있다. 낸드 플래시 메모리는 구조적 특징 때문에 데이터 갱신요청 시 제자리 덮어쓰기가 불가능하여 데이터 갱신요청이 자주 발생하는 요청과 그렇지 않은 요청을 정확히 구분하여 각 블록에 저장 및 관리함으로써 해결할 수 있다. 이러한 데이터 갱신요청에 분류기법을 핫 데이터 식별 기법이라고 하며 현재 다양한 연구가 진행되었다. 본 논문은 더 정확한 핫 데이터 검증을 위해 카운팅 필터를 사용하여 데이터 갱신요청 발생을 연속적으로 기록하고 또한 특정 시간 동안 요청된 갱신요청이 얼마나 자주 발생하는지를 고려하여 핫 데이터를 검증한다.

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

Abstract A storage device used in the mobile computing field should have low power, light weight, durability, etc., and should be able to effectively store and manage large-capacity data generated by users. NAND flash memory is mainly used as a storage device in the field of mobile computing. Due to the structural characteristics of NAND flash memory, it is impossible to overwrite in place when a data update request is made, so it can be solved by accurately separating requests that frequently request data update and requests that do not, and storing and managing them in each block. The classification method for such a data update request is called a hot data identification method, and various studies have been conducted at present. This paper continuously records the occurrence of data update requests using a counting filter for more accurate hot data validation, and also verifies hot data by considering how often the requested update requests occur during a specific time.

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

1. 서론

SSD(Solid State Drive)는 낸드플래시메모리 기반 저장장치로 빠른 데이터 입출력 속도와 낮은 전력 소모 그리고 가벼운 무게로 인한 휴대성 등에 다양한 장점이 있다[1]. 낸드 플래시 메모리는 페이지와 블록 단위로 구성된다. 데이터 쓰기와 읽기 명령에 경우 페이지 단위로 이루어지며 데이터 삭제 명령에 경우 블록 단위로 동작하

*교신저자 : 이승우(zpa007@gmail.com)
접수일 2022년 8월 4일 수정일 2022년 9월 12일 심사완료일 2022년 9월 19일

는 구조적 특징이 있다. 특히 낸드 플래시 메모리는 데이터 쓰기, 읽기 단위와 삭제 단위가 다른 구조적 특징으로 인해 데이터 갱신요청 시 제자리 덮어쓰기(in-place updates)가 불가능하여 쓰기 전 지우기(erase-before-write) 동작이 반드시 먼저 요구된다. 앞서 언급한 것처럼 낸드 플래시 메모리는 데이터 삭제 시 블록 단위로 동작하기 때문에 데이터 갱신요청 시 갱신요청 페이지를 포함한 블록 내의 모든 페이지를 새로운 블록에 복사하는 오버헤드가 발생하며 저장장치 내에 비어있는 블록의 수가 줄어들수록 쓰기 전 지우기 동작으로 인한 부가 연산이 증가하게 된다[2-8].

쓰기 전 지우기 동작을 최소화하기 위해서는 특정 페이지를 대상으로 발생하는 빈번한 데이터 갱신요청과 그렇지 않은 갱신요청을 정확히 구분할 필요가 있다. 이러한 갱신요청 분류기법을 핫 데이터 판단기법이라 하며 현재까지 다양한 핫 데이터 판단기법이 알려졌다[9-11].

본 논문은 더 정확한 핫 데이터 검증을 위해 데이터 갱신요청을 카운팅 필터에 연속적으로 기록하고 또한 연속적인 갱신요청에 발생 패턴을 함께 기록한다. 즉 데이터 특정 데이터 갱신요청에 대한 핫 데이터 판단 시 갱신요청의 연속성과 특정 갱신요청에 발생 패턴을 고려한 상대적으로 더 정확한 핫 데이터 판단기법이며 제안기법을 낸드 플래시 메모리를 저장매체로 주로 사용하는 IoT 분야 기기에서 적용하여 블록 단위 삭제로 인한 오버헤드를 줄일 수 있다.

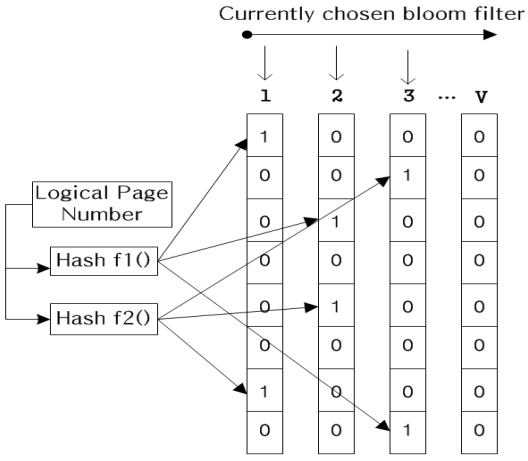
2. 관련 연구

2.1 기존 핫 데이터 식별 기술의 문제점

지금까지 알려진 핫 데이터 판단기법은 데이터 갱신요청 횟수를 단순히 누적 기록한 값을 기준으로 판단하는 MHF(Multi Hash Function Framework)기법과 데이터 갱신요청의 연속성을 기준으로 하는 BHF(Bloom filter Hash Function Framework) 기법으로 구분할 수 있다.

다수의 해시함수를 사용하는 MHF 기법은 데이터 갱신요청 시 갱신요청 LPN을 해싱하고 해싱 값에 대응하는 메모리에 순차적으로 0에서 1로 기록함으로 갱신요청 발생을 기록한다. 더 이상 갱신요청이 없다면 기록된 값은 정해진 시간 값에 맞춰 가장 먼저 기록된 값부터 순차적으로 0으로 초기화된다. 갱신요청에 누적 기록값을 0으로 초기화하여 메모리에 기록된 값이 항상 마지막 갱신요청을 기록한 값임을 나타낸다. 이러한 MHF 기법은 일반적으로 메모리상에 4bit로 구성된 메모리 배열로써 상위 2bit 값에 변화가 핫 데이터 판단 기준이 된다.

MHF 기법은 알고리즘이 단순하여 쉽게 적용할 수 있는 장점이 있다. 하지만 더 정확한 핫 데이터 판단에 있어 문제점이 존재한다. MHF 기법에 문제점은 특정 데이터 갱신요청이 짧은 시간에 집중적으로 발생한 후 더 이상 갱신요청이 발생하지 않은 경우에도 이를 핫 데이터로 판단하는 것이다. 그 이유는 핫 데이터 판단에 기준으로 상위 2bit의 갱신요청이 발생한 횟수만을 고려하기 때문이다. 단순히 특정 시간 동안 기록된 데이터 갱신요청에 발생빈도만을 고려하면 집중적인 특정 갱신요청 패턴이 더 이상 발생하지 않는 경우까지 핫 데이터로 판단하는 문제를 발생시킨다.



[Fig. 1] Bloom filter hash function framework

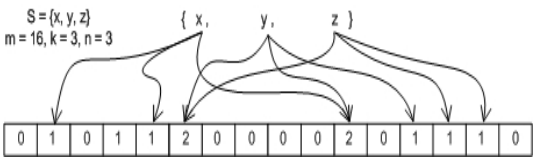
BHF 기법은 데이터 갱신요청 시 해당 LPN을 해시함수를 이용하여 해싱하고 그 값에 대응하는 블룸필터 값을 1로 변경한다. 또한 더 이상 데이터 갱신요청이 없는 경우 현재 선택된 블룸필터의 값은 0으로 초기화된다. BHF 기법은 데이터 갱신요청 사실을 기록하기 위해 다수의 블룸필터를 라운드 로빈 방식으로 운용한다. BHF 기법은 별도의 기록값 삭제 과정 없이 가장 오래전 기록된 블룸필터 부터 순차적으로 삭제하며 삭제 후 새로운 블룸필터를 계속해 생성한다. BHF 기법은 다수의 블룸필터에 기록된 데이터 갱신요청의 연속성을 기준으로 핫 데이터를 판단한다. 하지만 핫 데이터 판단 시 갱신요청 발생의 연속성만을 고려하면 짧은 시간 연속적으로 발생한 갱신요청 패턴이 자주 발생하는 경우 이를 정확히 핫 데이터로 판단하지 못하는 단점이 있다.

2.2 Bloom Filter와 Counting Filter

블룸필터(Bloom filter)란 n개 요소를 갖는 집합 N에서 s개의 요소로 이루어진 부분집합 S의 요소별 키값을 표현하기 위해 사용되는 확률 기반 자료구조이며 1970년 Burton Howard Bloom에 의해 발표되었다[12-14]. 블룸필터는 특정 요소를 삭제하는 것은 불가능하다. 특히 특정한 요소가 부분집합에 속한 것으로 판단될 경우라도 실제로는 부분집합에 속하지 않을 수 있는데 이것을 긍정오류라고 하며 반대 경우로 특정 요소가 부분집합에 속하지 않는 것으로 판단되는 경우 실제 해당 요소가 부분집합에 속할 수도 있는 경우를 부정오류라고 한다. 블룸필터는 부정오류가 절대 발생하지 않는다는 특징이 있다.

한편 블룸필터를 활용하는 경우 긍정오류 발생확률을 줄이기 위해 비트 배열의 크기를 적어도 부분집합에 요소의 개수를 15배 크기로 설정하고 추가로 해시함수의 개수 또한 3개 이상으로 설정할 때 긍정오류 발생확률은 전체 1% 미만인 것으로 알려져 있다.

블룸필터는 빠른 속도로 특정 요소의 부분집합 포함 여부를 조사할 수 있는 알려진 방법이다. 하지만 블룸필터는 부분집합의 항목을 수정할 수 없다는 단점이 있다. 또한 항목 제거가 불가능하여 항목 집합 S가 동적으로 변화는 경우 항목이 변경될 때마다 새로운 비트 배열을 생성해야 하는 단점이 있다. 이러한 블룸필터의 단점을 극복한 것이 카운팅 블룸필터이다.



[Fig. 2] Counting filter

카운팅 필터(Counting filter)는 1998년 L. Fan에 의해 제안되었다[15]. 블룸필터에 단점 중 항목 변경 시 필터를 재생성해야 하는 단점을 해결하기 위해 n비트로 확장한 카운터를 사용하여 항목 추가 시 해당 n비트의 값을 1씩 증가시키고 삭제 시 값을 1씩 감소시킬 수 있다. 또한 항목 검색 시 해당 n비트의 값이 0이 아닌지 확인하게 된다. 카운팅 필터는 항목 삽입과 삭제에 대해 블룸필터와 비교해 상대적으로 자유로우며 카운터 값을 저장하는 n비트에 항목 삽입 삭제에 대한 사실을 표현할 수 있게 된다.

본 논문에서는 핫 데이터 판단을 위해 전체 LPN 중 핫 데이터로 판정된 LPN을 표현하기 위해 카운팅 필터를 이용한다. 또한 핫 데이터 판정 역시 카운팅 필터의 n비트를 이용하여 핫 데이터 기록과 판정을 동시에 진행함으로써 효과적인 메모리 절약과 핫 데이터 판정과정의 단순성을 동시에 확보하였다.

3. 제안기법

제안기법의 동작 과정은 크게 두 가지 과정으로 진행된다. 첫째 데이터 갱신요청 시 핫 데이터 판단에 필요한 데이터를 기록하는 과정과 둘째 기록된 데이터를 바탕으로 핫 데이터 여부를 판단하는 과정으로 진행된다.

제안기법은 핫 데이터 판단에 필요한 데이터를 기록하기 위해 카운팅 필터를 이용한다. 이전 기법에서 핫 데이터 판단에 필요한 데이터를 기록하기 위해 주로 단순 비트 배열 또는 블룸필터를 이용한다. 하지만 카운팅 필터 사용 시 메모리 사용량을 더 줄일 수 있고 더 많은 핫 데이터 판단에 필요한 데이터를 기록할 수 있으며 빠른 속도로 검색을 수행할 수 있다. 또한 제안기법은 카운팅 필터 검색성능 향상을 위해 기존에 제안한 기법과 비교해 해시함수 사용개수를 줄였으며 카운팅 필터 운용방식을 더욱 효율적으로 개선하였다.

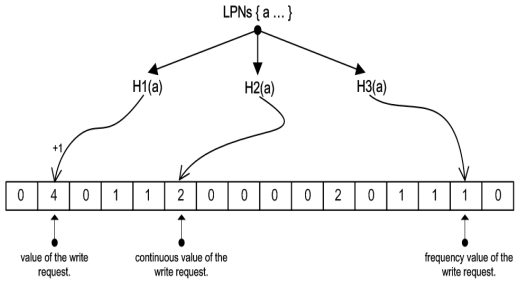
3.1 데이터 갱신요청 패턴

데이터 갱신요청에 패턴은 다양하기에 더 정확한 핫 데이터 판단을 위해서는 데이터 갱신요청에 연속성과 특정 패턴에 데이터 갱신요청이 발생하는 빈도를 함께 고려해야 한다. 연속적인 데이터 갱신요청은 크게 짧은 시간 동안 연속하여 발생하는 경우와 상대적으로 긴 시간 동안 연속하여 데이터 갱신요청이 발생한 경우로 구분할 수 있다. 또한 특정 패턴의 갱신요청이 얼마나 자주 발생하는 횟수를 데이터 갱신요청에 빈도라고 할 수 있다. 일반적으로 핫 데이터 판단 시 데이터 갱신요청의 연속성만을 고려하는 경우 데이터 갱신요청이 한 번이라도 연속적인 경우에도 이를 핫 데이터로 판단하는 문제가 있다. 특히 데이터 갱신요청 패턴 중 짧은 연속성에 갱신요청이 빈번하게 발생하는 경우와 긴 연속성에 갱신요청이 단순히 한번 발생하는 경우를 정확히 판단해야 한다. 제안기법은 더 정확한 핫 데이터 판정을 위해 데이터 갱신요청의 연속성과 특정 패턴의 발생빈도를 함께 고려한다.

3.2 핫 데이터 판단을 위한 데이터 기록 과정

제안기법은 핫 데이터 판단에 필요한 데이터를 기록하기 위해 데이터 갱신요청에 해당하는 LPN을 해시함수를 이용해 해싱되고 반환된 값에 해당하는 카운팅 필터에 핫 데이터 판단과정에 필요한 데이터를 기록한다. 또한 카운팅 필터에 기록된 값은 특정 시점에 핫 데이터 판단 과정에 사용되며 핫 데이터로 판단되는 경우 갱신요청에 해당하는 LPN은 카운팅 필터에 계속 기록되며 핫 데이터로 판단 되지 않는 경우 카운팅 필터에서 삭제한다.

제안기법은 데이터 갱신요청 시 특정 LPN을 H1 해시함수로 해싱하여 대응하는 카운팅 필터에 갱신요청 사실을 누적 기록한다. 즉 갱신요청 시 해당 카운팅 필터에 값을 1씩 증가시키며 더 이상 갱신요청이 발생하지 않는다면 설정된 시간 간격으로 1씩 감소하게 된다. 하지만 특정 LPN에 데이터 갱신요청을 단순히 누적 기록하는 것만으로는 데이터 갱신요청에 연속성을 판단할 수 없다. 예를 들어 현재 H1에 기록된 값이 2일 경우 데이터 갱신요청이 연속적으로 발생하여 기록된 값인지 더 이상 데이터 갱신요청이 발생하지 않고 감소 중에 기록된 값인지 판단할 수 없다.

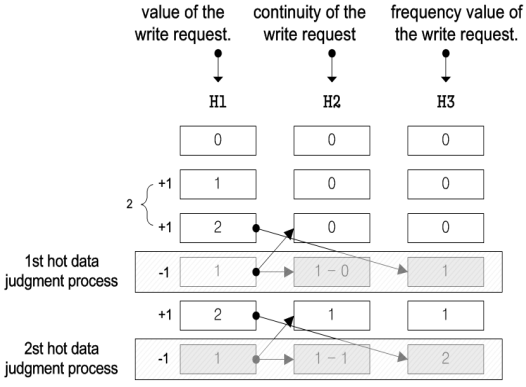


[Fig. 3] Operation process of the proposed method

[그림 3]에서 제안기법은 이를 해결하기 위해 해시함수 H2에 대응하는 카운팅 필터에 데이터 갱신요청에 연속성을 기록한다. 이를 위해 H1에 대응하는 카운팅 필터에 값이 최초 감소할 때 직전 H2에 기록된 연속성 값과 현재 H1에 기록된 연속성 값을 뺄셈하여 H2에 대응하는 카운팅 필터에 값이 갱신되며 이 값은 계속하여 유지된다. 이는 H1에 기록된 현재 연속성이 최초 감소하기 직전에 H2에 기록된 이전 데이터 갱신요청에 연속성 값에 차이를 통해 현재 데이터 갱신요청이 짧은 연속성에 갱신요청인지 긴 연속성에 갱신요청인지 판단하는 데이터로 활용한다.

또한 제안기법은 더 정확한 핫 데이터 판단을 위해 H1에 기록된 현재 연속성이 최초 감소할 때 특정 연속성에 갱신요청이 발생하는 빈도를 H3에 대응하는 카운팅 필터에 값을 1로 기록하여 해당 패턴 발생 횟수를 기록한다. 만약 더 이상 데이터 갱신요청이 발생하지 않는 경우 H1에 대응하는 카운팅 필터에 값이 연속 감소할 때 H3에 대응하는 카운팅 필터에 값 또한 연속하여 1씩 감소시킨다. 이러한 이유는 해당 연속성에 갱신요청 패턴이 더 존재하지 않기 때문이다.

3.3 핫 데이터 판단과정



[Fig. 4] Judgment process

제안기법의 핫 데이터 판단과정은 특정 LPN에 대한 데이터 갱신요청을 누적 기록한 H1에 대응하는 카운팅 필터에 값이 최초 감소할 때 동작한다. 제안기법에 핫 데이터 판단 기준은 특정 LPN에 데이터 갱신에 대한 짧은 연속성을 가진 데이터 갱신요청에 빈도가 높은 경우와 상대적으로 긴 연속성을 가진 데이터 갱신요청에 빈도가 이전 갱신요청에 기록된 값보다 큰 경우 핫 데이터로 판단하며 그 외에 데이터 갱신요청은 단순 데이터 업데이트 동작으로 판단한다.

제안기법은 데이터 갱신요청에 연속성을 판단하기 위해 특정 데이터 갱신요청에 연속성이 감소하기 직전에 H2 카운팅 필터에 기록된 값과 감소 직후에 H2 카운팅 필터에 기록된 값을 서로 비교한다. 이는 직전 갱신요청에 연속성과 현재 갱신요청에 연속성을 서로 비교하여 현재에 갱신요청이 이전보다 짧은 연속성에 갱신요청인지 아닌지를 판단하기 위함이다. 더불어 H3 카운팅 필터에 기록된 현재 갱신요청에 연속성에 빈도값으로 현재 연속성에 갱신요청이 단순히 한번 발생한 것인지 아니면 자주 발생하는 연속성에 갱신요청인지를 판단하게 된다.

결론적으로 직전까지의 데이터 갱신요청에 대한 H2와 H3에 기록된 값과 현재의 데이터 갱신요청에 대한 H2와 H3에 기록된 값을 비교함으로 해당 LPN에 대한 데이터 갱신요청의 연속성과 빈도에 변화를 검증하여 핫 데이터 여부를 판단하게 된다.

[그림 4]을 통해 제안기법에 핫 데이터 판단 데이터 기록 및 핫 데이터 판단과정을 설명한다. 현재 각 n비트의 값은 모두 0으로 초기화된 상태이며 이후 특정 LPN에 대한 데이터 갱신요청이 두 번 연속하여 발생하였고 이후 갱신요청이 더 이상 없어 갱신요청을 누적 기록하는 H1에 n비트값을 1 감소한다. 이때 해당 갱신요청에 연속성을 기록하기 위해 직전 H2에 기록된 연속성 값인 0과 현재 H1에 기록된 연속성 값인 1에 차이를 현재 H2에 기록한다. 이는 해당 갱신요청이 끝나는 시점에 해당 갱신요청에 연속성을 H2에 기록한 것이다. 또한 해당 갱신요청이 종료되었기에 해당 갱신요청 패턴을 H3에 1로 기록한다. 갱신요청이 감소하는 시점은 하나의 갱신요청 패턴이 끝났음을 의미함으로 핫 데이터 판단 절차가 진행된다.

핫 데이터 판정에 필요한 데이터는 H2와 H3에 값이며 현재 H2에 값이 1이므로 이를 통해 해당 갱신요청은 짧은 시간 연속하여 갱신요청이 발생했음을 알 수 있으며 현재 H3에 값이 1이므로 짧은 시간 연속한 갱신요청이 한번 발생했음을 알 수 있다. 즉 직전 각 H2와 H3에 값 0, 0과 현재 H2와 H3에 값은 1, 1과 비교했을 때 갱신요청의 연속성과 요청 패턴의 횟수 모두 증가함을 의미함으로 해당 LPN은 핫 데이터로 판단한다.

이후 한 번에 갱신요청이 발생하고 이후 갱신요청이 더 이상 없어 H1에 값을 1 감소시킨다. 현재 H2와 H3에 값은 0, 2이고 직전 H2와 H3에 값은 1, 1이므로 현재 갱신요청은 직전 갱신요청에 비해 연속성이 짧다. 하지만 갱신요청 빈도는 더 크기 때문에 짧은 시간 동안 지속적인 갱신요청을 핫 데이터로 정확히 판단할 수 있다.

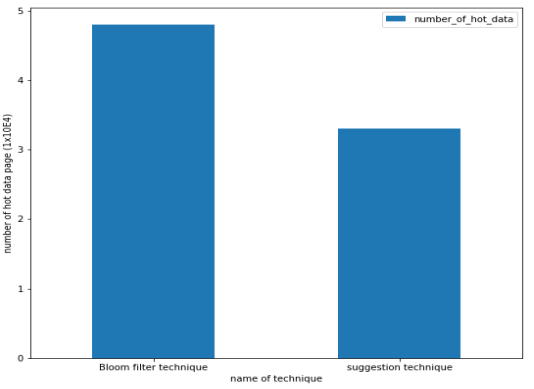
4 실험 결과

이 장에서는 기존기법과 제안기법을 시뮬레이션 프로그램으로 구현하고 일반적인 사용 환경에서 생성한 워크로드를 각각 적용하여 핫 데이터 판단 비율을 분석하여 성능을 검증한다.

4.1 실험환경 및 Trace 파일

실험에 사용된 시스템은 전체 SSD의 크기는 8GB가 되도록 65,536개의 블록을 할당하였으며 읽기와 쓰기 연산 중 데이터 갱신요청 발생 횟수는 약 480,000회이다. 또한 실험 진행에 있어 쓰기 요청 이외 다른 요소는 고려하지 않는다.

4.2 핫 데이터 판단 비율



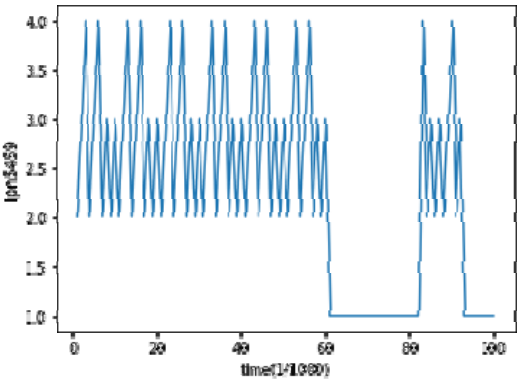
[Fig. 5] Number of hot data pages

[그림 5]에 경우 전체 데이터 갱신요청 중 블룸필터를 이용하여 핫 데이터로 판단된 갱신요청은 약 48,000회이고 비율은 약 11%이다. 또한 제안기법 적용 시 핫 데이터로 판단된 데이터 갱신요청은 약 33000회이고 비율은 약 7%이다.

결과적으로 기존기법과 비교해 제안기법에 경우 약 4% 정도 핫 데이터 판단 비율에 차이가 발생하였다. 이는 블룸필터를 이용한 기존기법에 경우 핫 데이터 검증 시 갱신요청에 연속성만을 고려하여 특정 시간 구간 동안 갱신요청이 발생한 후 더 이상 갱신요청이 발생하지 않는 단순 데이터 갱신에 대해서도 핫 데이터로 판정하기 때문이다. 반면 제안기법의 경우 데이터 갱신에 연속성과 더불어 특정 패턴의 갱신요청에 빈도를 함께 고려하기 때문에 단순 데이터 갱신요청에 경우 핫 데이터로 판단하지 않는다.

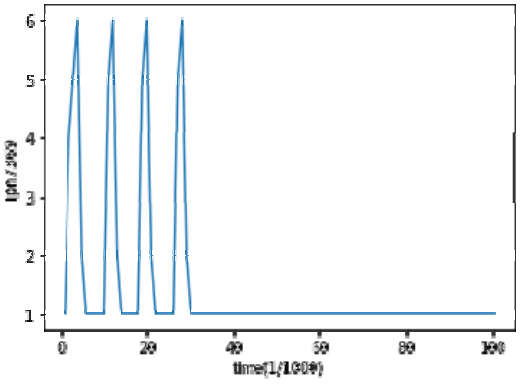
[그림. 6]에 경우 lpn 5459에 데이터 갱신요청 패턴을 볼 수 있다. 이 경우 짧은 시간 간격으로 연속하여 데이터 갱신이 요청되는 패턴이 지속하여 발생하는 것을 알 수 있다. 단순히 갱신요청에 연속성만을 판단 기준으로 하는 기존기법은 단순히 갱신요청이 충분히 연속적이지 않음으로 핫 데이터로 판정하지 않게 된다. 제안기법은 핫 데이터 판정 시 갱신요청에 빈도를 함께 고려하기

때문에 해당 lpn을 핫 데이터로 판정하여 별도 관리함으로 블록 단위 삭제 횟수를 감소시킬 수 있다.



[Fig. 6] Judgment process of lpn 5459

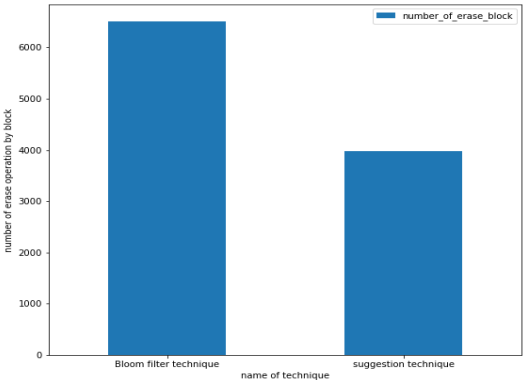
[그림 7]에 경우 lpn 7369에 데이터 갱신요청 패턴을 볼 수 있다. 이 경우 연속적인 갱신요청 발생하였으나 이후 갱신요청이 발생하지 않는다. 제안기법은 이러한 패턴을 정확히 핫 데이터로 판단하기 위해서는 데이터 갱신요청에 빈도를 고려한다. 즉 특정 시점에 순간적으로 발생한 갱신요청은 핫 데이터로 판단하지 않는다.



[Fig. 7] Judgment process of lpn 7369

4.3 블록 단위 삭제 비율

더 정확한 핫 데이터 판단은 핫 데이터 판단 실패로 인해 발생하는 블록 단위 삭제 횟수를 확인하는 것이다. [그림 8]에 경우 각 기법 적용 시 블록 단위 삭제 횟수이며 블룸필터 기법 적용 시 상대적으로 제안기법보다 더 많은 블록 단위 삭제가 발생하였다. 이는 블룸필터를 이용한 핫 데이터 검증기법 적용 시 단순히 연속적인 데이터 갱신요청에 대한 검증만으로 핫 데이터를 판단하기 때문에 짧은 연속성의 데이터 갱신요청을 핫 데이터로 판단하지 못하여 블록 내 무효페이지 비율이 증가하여 이로 인한 블록 단위 삭제요청이 증가한 것을 알 수 있다. 결과적으로 제안기법이 블룸필터를 이용한 기법보다 더 효과적으로 핫 데이터 판단하여 이를 처리함으로 블록 단위 삭제 비율이 낮음을 알 수 있다.



[Fig. 8] Number of opration by block

5. 결론

본 논문은 낸드 플래시 메모리 사용 시 해결해야 하는 제자리 덮어쓰기 불가능 문제를 발생시키는 데이터 갱신요청을 더 효과적으로 처리하기 위해 카운팅 필터를 이용하여 데이터 갱신요청에 연속성과 빈도를 함께 고려한 개선된 핫 데이터 판단기법을 제안하였으며 기존 제안기법과 비교해 해시함수 수를 감소하였으며 핫 데이터 판단 절차를 개선하였다. 특히 이전 기법에 핫 데이터 검증기법에 문제점인 데이터 갱신요청에 연속성만을 고려하는 문제점을 해결하였고 제안기법은 데이터 갱신요청을 연속성과 함께 빈도를 함께 고려하기 때문에 상대적으로 블록 단위 삭제 횟수가 상대적으로 더 적게 발생하였다. 이러한 이유는 짧은 연속성의 갱신요청이 지속하여 발생하는 경우를 더 정확히 핫 데이터로 검증한 결과로 해석할 수 있다.

REFERENCES

[1] 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.

[2] T.S.Chung, D.J.Park, D.H.Lee, S.W.Lee, and H.J.Song, "System Software for Flash Memory: A Survey," EUC 2006: Embedded and Ubiquitous Computing, pp.394-404, 2006.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] O.Rottenstreich, and I.Keslassy, "The Bloom Paradox: When Not to Use a Bloom Filter," IEEE/ACM Transactions on Networking, Vol.23, Issue.3, 2012.

[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] H.S.Lim, J.W.Lee and C.H.Yim, "Complement Bloom Filter for Identifying True Positiveness of a Bloom Filter," IEEE Communications Letters, Vol.19, Issue.11, pp.1905-1908, 2015.

[15] P.Lin, F.Wang, W.Tan, and H.Deng, "Enhancing Dynamic Packet Filtering Technique with d-Left Counting Bloom Filter Algorithm," International Workshop on Intelligent Networks and Intelligent Systems (ICINIS), pp.530-533, 2009.

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 소프트웨어콘텐츠계열 조교수

〈관심분야〉
임베디드 시스템, Nand Flash Memory