일괄처리 기반 향상된 블록 매핑 기법의 설계

이현섭*
백석대학교 컴퓨터공학부 교수

A Design of Enhanced Block Mapping Method Based on Batch Processing

Hyun-Seob Lee*
Professor, Division of Computer Engineering, Baekseok University

요 약 최근 플래시 메모리 기반 저장장치는 다양한 분야에서 널리 활용되고 있다. 그러나 플래시 메모리는 제자리 갱신이 불가능한 물리적 특성 때문에 데이터 수정 시 해당 블록을 먼저 소거한 후 새로운 데이터를 기록해야 한다. 이러한 특성 때문에 논리적 주소와 물리적 주소 간의 매핑을 관리하는 FTL(flash translation layer)이 필수적으로 요구된다. 현재 대표적인 FTL 방식으로는 페이지 매핑 기법이 사용되고 있다. 페이지 매핑 방식은 높은 성능을 제공하지만, 대용량 저장장치에서 매핑 정보 유지를 위한 메모리 오버헤드가 심각한 문제로 대두되고 있다. 이에 대한 대안으로 블록 매핑 기법이 있으며, 이 방식은 상대적으로 작은 매핑 정보를 유지하여 메모리 오버헤드를 줄일 수 있다. 하지만 블록 매핑 방식은 낮은 메모리 사용량이라는 장점에도 불구하고 빈번한 쓰기/지우기 연산으로 인한 성능 저하 문제를 가지고 있다. 본 연구에서는 페이지를 버퍼에 모아서 일괄처리하는 향상된 블록 매핑 기법을 제안한다. 제안하는 방법은 접근 패턴의 지역성을 활용하여 관련 페이지들을 작은 블록 버퍼에 수집한 후 블록 단위로 처리함으로써 기존 블록 매핑의 과도한 쓰기/지우기 연산 문제를 개선한다. 동시에 매핑 테이블 크기를 현저히 줄여 메모리 오버헤드 문제도 해결한다. 마지막으로 실험을 통해 제안하는 방법이 페이지 매핑 기법과의 성능 격차를 효과적으로 줄일 수 있음을 검증한다.

주제어 : 낸드 플래시 메모리, 플래시 전환 계층, 블록 매핑 기법, 페이지 매핑 기법, 대용량 저장장치

Abstract In recent years, flash memory-based storage devices have been widely used in a variety of applications. However, the physical characteristics of flash memory make it impossible to update in-place; when modifying data, the block must first be erased before new data can be written. This makes it essential to have FTL (flash translation layer) that manages the mapping between logical and physical addresses. Currently, the most common FTL method is the page mapping technique. While page mapping provides high performance, the memory overhead of maintaining mapping information in large storage devices is becoming a serious issue. An alternative is block mapping, which can reduce memory overhead by maintaining relatively small mapping information. However, despite its advantage of low memory usage, block mapping suffers from performance degradation due to frequent write/erase operations. In this work, we propose an improved block mapping technique that batches pages together in a buffer. The proposed method improves the problem of excessive write/erase operations in traditional block mapping by utilizing the locality of the access pattern to collect related pages into a small block buffer and process them in blocks. At the same time, it also solves the memory overhead problem by significantly reducing the mapping table size. Finally, we verify through experiments that the proposed method can effectively reduce the performance gap with page mapping techniques.

Key Words : NAND Flash Memory, Flash Translation Layer, Block Mapping Method, Page Mapping Method, Large Capacity Storage

This paper was supported by 2025 Baekseok University Research Fund
*교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2025년 07월 11일 수정일 2025년 08월 06일 심사완료일 2025년 08월 15일

1. 서론

플래시 메모리 기반 저장장치(flash memory based storage devices)는 기존의 하드 디스크 드라이브(HDD, hard disk drive)를 대체하여 컴퓨터 시스템의 주요 저장 매체로 자리 잡고 있다. 특히 SSD (solid state drive)와 같은 NAND 플래시 메모리 기반 저장장치는 높은 접근 속도, 낮은 전력 소비, 그리고 향상된 내구성 등의 장점으로 데이터 센터, 개인용 컴퓨터, 그리고 모바일 기기 등 다양한 분야에서 사용되고 있다. 플래시 메모리는 기존의 자기 디스크 저장장치와 달리 제자리 갱신(in-place update)이 불가능하다는 물리적인 특성을 가지고 있다[1-4]. 이는 데이터를 수정하기 위해서는 해당 블록을 먼저 소거(erase)한 후 새로운 데이터를 쓰는 과정이 필요하기 때문이다. 이러한 특성 때문에 플래시 메모리 기반 저장장치는 논리적 주소(logical address)와 물리적 주소(physical address) 간의 매핑을 관리하는 FTL(flash translation layer)[5, 6]를 사용한다. FTL에서 사용하는 대표적인 매핑 방법은 블록 매핑(block mapping) 기법[7, 8]과 페이지 매핑(page mapping) 기법[9, 10]이 있다. 블록 매핑 기법은 블록 단위로 논리적인 주소와 물리적 주소를 매핑하여 매핑 정보가 작은 반면 플래시 메모리의 기본 쓰기 연산은 페이지 단위로 동작하기 때문에 내부적으로 많은 횟수의 쓰고/지우기 연산이 발생한다는 단점이 있다. 따라서 기존의 FTL에서는 매핑 정보를 관리하기 위해 페이지 매핑(page mapping) 기법을 사용해 왔다. 페이지 매핑 방식은 논리적 페이지 번호와 물리적 페이지 번호 간의 일대일 매핑을 제공하여 높은 성능과 효율적인 가비지 컬렉션(GC, garbage collection)[11, 12]을 가능하게 한다. 그러나 저장장치의 용량이 테라바이트급으로 증가하면서 페이지 매핑 방식의 한계가 뚜렷하게 나타나고 있다. 특히 대용량 저장장치에서 페이지 매핑 방식은 매핑 정보를 유지하기 위한 메모리 오버헤드 문제를 야기하며, 저장장치의 용량이 증가할수록 심각한 성능 저하와 비용 증가를 초래한다. 따라서 더 효율적인 매핑 기법의 연구가 필요하다.

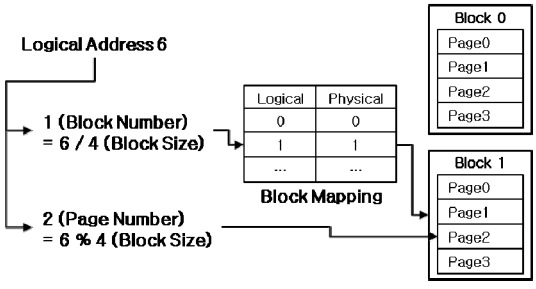
본 연구는 대용량 플래시 메모리 기반 저장장치에서 매핑 공간의 오버헤드를 효과적으로 줄이기 위해 페이지를 블록 단위로 일괄처리(batch processing)[13, 14]하는 블록 매핑 기법을 제안한다. 이 방법은 접근 패턴의 지역성이 있는 페이지들을 작은 버퍼에 모아 블록 단위 쓰기 동작을 수행한다. 이 방법을 통해 블록 매핑을 통해 발생하는 많은 횟수의 쓰기/지우기 연산 문제를 개선할 수 있다. 또한, 블록 매핑 방식은 페이지 단위가 아닌 블록 단위로 매핑을 관리함으로써 매핑 테이블의 크기를 현저히 줄일 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 플래시 메모리 기반 저장장치의 특성과 기존 매핑 기법들에 대한 관련 연구를 검토한다. 3장에서는 제안하는 일괄처리 기반 블록 매핑 기법의 구조와 알고리즘을 상세히 설명한다. 4장에서는 실험 환경과 성능 평가 결과를 제시하고, 5장에서는 연구 결과를 종합하여 결론을 도출한다.

2. 배경 및 관련 연구

2.1 블록 매핑 기법

플래시 메모리는 여러 개의 블록으로 구성되어 있고 각 블록은 여러 개의 페이지로 이루어져 있다. 이러한 구조에서 메모리의 주소를 식별하고 관리하기 위해 블록 매핑 기법은 블록 단위로 논리적인 주소와 물리적인 주소를 매핑한다. 그리고 논리적인 블록의 주소는 블록 크기로 나눈 몫을 통해 식별하고 블록 내의 페이지 위치는 블록 크기로 나눈 나머지 값을 이용한 오프셋(offset) 결과를 통해 식별한다.



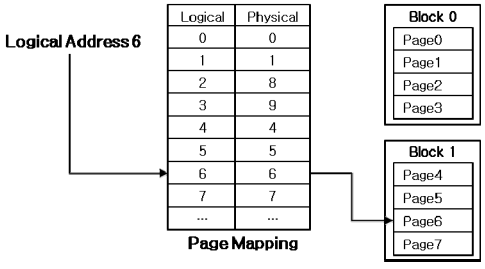
[Fig. 1] Block Mapping Method

Fig. 1은 블록 매핑 기법의 알고리즘을 보여주고 있다. 그림의 예에서 각 블록은 4개의 페이지로 구성되어 있다. 여기서 논리적 페이지 주소 6의 블록 번호는 블록 크기 4로 나누는 연산을 통해 1이라는 것을 확인할 수 있다. 그다음 매핑 정보를 통해 블록 1은 물리적 블록 1에 매핑되어 있는 것을 확인할 수 있다. 또한, 블록 내의 페이지 번호는 오프셋 연산을 통해 2임을 확인할 수 있다. 이러한 방법을 통해 블록 매핑 기법은 논리적인 주소를 물리적인 주소로 변환한다. 그러나 이 방법은 동일한

주소의 페이지에 쓰기 연산을 수행해야 할 때 제자리 갱신을 하지 못하는 플래시 메모리의 특성 때문에 블록 내의 유효한 데이터들을 다른 블록으로 이전해야 한다. 따라서 이러한 데이터 이전 과정을 수행하기 위해 대량의 추가적인 연산을 수행해야 하는 단점이 있다.

2.2 페이지 매핑 기법

페이지 매핑 기법은 페이지 단위로 주소를 관리한다. 따라서 이미 데이터가 기록된 페이지에 데이터가 갱신되면, 변경된 페이지 데이터를 새로운 페이지에 쓰고 매핑 테이블에 정보를 유지한다. 페이지 매핑 기법은 이러한 방법을 통해 데이터 갱신이 발생해도 내부적으로 확산될 수 있는 읽기/쓰기/지우기 연산을 지연시키고 성능을 최적화하는 장점이 있다.



[Fig. 2] Page Mapping Method

Fig. 2는 페이지 매핑 기법의 알고리즘을 보여주고 있다. 페이지 매핑은 모든 논리적 주소에 대한 물리적 페이지 주소를 매핑한다. 그림의 예에서 페이지 주소 6은 플래시 메모리의 페이지 6에 매핑되어 있는 것을 확인할 수 있다. 따라서 추가적인 연산 없이 매핑이 되어 있는 페이지에 빠르게 접근하는 장점이 있다. 또한, 페이지 6의 데이터에 변경된 데이터 쓰기 요청이 발생해도, 비어 있는 새로운 페이지에 갱신된 데이터를 쓰고 매핑 정보를 업데이트한다. 예를 들어 논리적인 주소 페이지 2는 업데이트가 발생하여 플래시 메모리 내에 비어 있는 페이지 8에 데이터를 저장하였다. 그리고 저장된 페이지 주소 정보를 매핑 테이블에 반영하여 이후 페이지 6의 데이터를 찾는 경우 플래시 메모리의 페이지 8에서 데이터를 읽어온다. 만약 페이지 매핑 기법 대신 블록 매핑을 사용하는 상황에서 2번 페이지에 갱신이 발생했을 경우, FTL은 페이지 2번의 데이터를 유지하고 있는 블록의 모든 페이지를 비어 있는 블록으로 이전하는 대량의 읽기/쓰기 동작을 수행했을 것이다. 즉, 페이지 매핑 기법은 데이터 갱신이 발생하면 해당 페이지만 비어 있는 물리적 페이지로 이전하기 때문에 블록 매핑 기법을 적용하였을 때 발생할 수 있는 대량의 추가적인 연산을 지연시킬 수 있는 장점이 있다. 그러나 페이지 매핑은 페이지 단위로 매핑 정보를 관리하기 때문에 매핑 테이블을 유지하는 비용이 큰 단점이 있다.

2.3 매핑 정보 유지의 오버헤드

앞에서 살펴본 페이지 매핑 테이블은 데이터 쓰기 동작에 최적화된 성능을 보여주지만 매핑 테이블을 관리하는 비용은 저장장치의 용량이 증가할수록 더 큰 부담을 주는 문제가 있다. 예를 들어 하나의 페이지가 유지할 수 있는 데이터를 512B로 가정하였을 때, 1TB의 데이터를 저장하기 위해서는 약 8GB의 매핑 정보가 필요하다. 또한, 저장장치의 페이지의 수가 증가하는 것은 매핑 정보를 관리하는 체계에 또 다른 관리 오버헤드를 야기한다. 이러한 메모리 오버헤드는 저장장치의 용량과, 구성하는 페이지의 개수가 증가할수록 비례하여 증가한다.

Bit	Byte	HEX	DEC	Max Size
32	4	FFFFFFFF	4,294,967,295	2TB (512B)

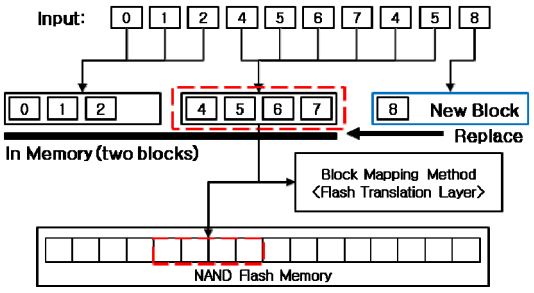
[Fig. 3] Analysis of the map size of 2TB

Fig. 3은 512B의 페이지로 구성된 2TB 저장장치에서 맵 데이터의 크기를 분석한 결과이다. 맵 데이터의 각 엔트리는 논리적인 주소에 대한 물리적인 주소를 표현한다. 그리고 각 주소는 정수 형태로 관리되는데 일반적으로 하나의 정수형 변수를 유지하기 위해 4B(32b)의 메모리 자원이 필요하다. 32 Bit 크기의 각 변수는 16진수로 변환하였을 때 0에서 0xFFFFFFFF의 값까지 표현할 수 있다. 그리고 최댓값인 0xFFFFFFFF는 십진수 기준으로 4,294,967,295를 의미한다. 그리고 하나의 페이지가 512B라고 가정한 경우 맵 데이터에서 각 매핑 정보가 표현할 수 있는 최대 주소의 범위는 2TB(2,199,023,255,040 = 4,294,967,295 × 512)이다. 즉 현재의 컴퓨팅 환경과 변수 체제에서 구현할 수 있는 최대 저장장치의 크기는 2TB 이하라는 한계가 있다. 이러한 한계를 극복하고 2TB를 초과하는 매핑 정보를 유지하기 위해서는 각 매핑 정보의 크기를 64B로 확장해야 한다. 이러한 확장은 대량의 메타데이터 수정과 추가적인 메모리 오버헤드의 문제가 있다. 2TB 데이터를 매핑하기 위한 또다른 방법은 페이지의 크기를 2K에서 16K까지 확장하는 방법이다. 이 경우 각 매핑 정보는 증가한 페이지 크기의 비율

만큼 매핑 정보의 총량을 줄일 수 있는 장점이 있다. 그러나 페이지의 크기를 확장하는 것은 제한이 있기 때문에 맵의 크기를 줄이기 위한 추가적인 연구가 필요하다.

3. 일괄처리 기반 향상된 블록 매핑 기법

3.1 핵심 아이디어

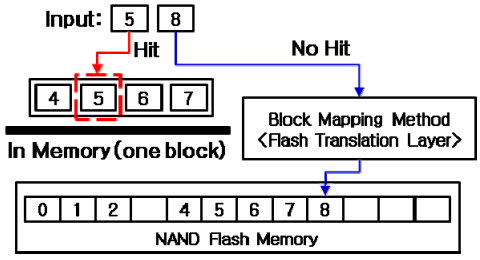


[Fig. 4] Key Idea

Fig. 4는 제안하는 아이디어의 핵심 아이디어를 보여주고 있다. 그림의 예에서는 하나의 블록은 4개의 페이지로 구성된다고 가정하였다. 그리고 메모리 버퍼에 블록 단위로 총 2개의 데이터량을 유지할 수 있는 블록 버퍼 공간을 할당하였다. 이때 데이터 쓰기 요청을 위해 입력된 페이지 0, 1, 2는 메모리 버퍼의 첫 번째 블록 버퍼에 할당하였다. 그다음 페이지 4, 5, 6, 7은 두 번째 블록 버퍼에 할당하였다. 이후 두 번째 블록 버퍼에 유지하고 있는 페이지 4, 5와 동일한 주소로 갱신된 데이터의 쓰기 요청이 입력되었을 때 아직 플래시 메모리에 반영하지 않고 블록 버퍼에 유지되고 있는 페이지를 찾아서 데이터를 갱신한다. 마지막으로 페이지 8에 쓰기 요청이 들어왔을 때 이 페이지를유지할 수 있는 블록 버퍼가 없기 때문에 기존의 블록을 커밋하고 새로운 블록을 버퍼에 유지한다. 커밋하는 블록은 블록 버퍼 중 가장 많은 데이터를 유지하고 있는 두 번째 블록 버퍼를 선택한다. 그리고 커밋하는 블록은 FTL에서 블록 매핑 기법에 따라 블록 단위로 플래시 메모리에 저장된다. 마지막으로 새로운 블록 버퍼에 수 있는 블록 버퍼를 할당하여 새로운 데이터를 버퍼에 유지한다. 이 방법은 블록 버퍼를 통해 전체 매핑 데이터의 크기를 최적화할 수 있다. 또한, 일부 블록에 집중된 데이터 쓰기 요청이 있을 경우, 중복된 페이지 주소에 대한 요청을 감소시킬 뿐만 아니라 쓰기 요청을 지연시켜서 성능을 향상할 수 있는 장점이 있다.

3.2 데이터 접근 정책

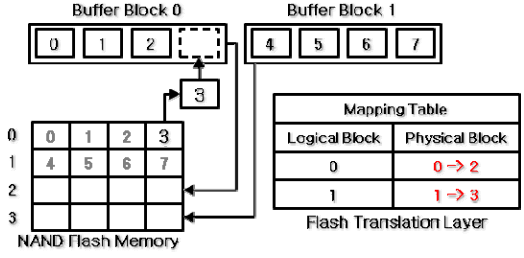
본 연구에서 제안하고 있는 향상된 블록 매핑 기법은 읽기 및 쓰기 요청이 있는 특정 페이지가 버퍼에 있을 경우 플래시 메모리의 직접 접근을 지연시킨다. 특히 쓰기 요청의 경우 플래시 메모리의 특성에 의해 내부적으로 확산되는 추가적인 연산을 줄일 수 있는 장점이 있다.



[Fig. 5] Data Access Policy

Fig. 5는 데이터 접근 정책을 보여주고 있다. 그림에서는 블록 버퍼를 1개만 유지하고 있다. 그리고 버퍼는 4개의 페이지를 유지하고 있다. 이때 페이지 5에 대한 탐색을 요청하면 데이터 접근 정책은 버퍼 내에 유지되어 있는 최신 데이터를 탐색한다. 그러나 페이지 8에 대한 접근 요청이 있을 경우, 해당 페이지가 버퍼에 없기 때문에 FTL의 매핑 정보를 참조하여 요청된 데이터를 탐색한다.

3.3 블록 버퍼 커밋 정책

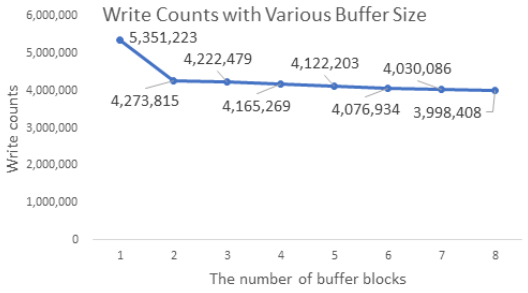


[Fig. 6] Block Buffer Commit Policy

Fig. 6은 버퍼 블록 0과 1을 교체하는 블록 버퍼 교체 정책의 예를 보여주고 있다. 논리적 블록 0과 1이 물리적 블록 0과 1에 각각 매핑되어 있는 상황을 가정하였다. 이때 버퍼 블록 0에 유지되고 있는 논리적 블록 0은 플래시 메모리에 저장된 페이지 중 중복되지 않은 페이지 0을 읽어서 하나의 데이터 블록 형태를 구성한다. 그다음 비어 있는 물리적 블록 2에 저장하고 매핑 정보를 갱신

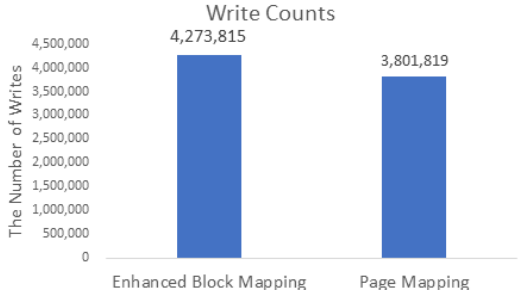
한다. 그다음 버퍼 블록 1에 유지되고 있던 논리적 블록 1은 모든 페이지를 유지하고 있다. 따라서 버퍼 블록의 데이터를 비어 있는 블록 3에 저장하고 매핑 테이블을 갱신한다.

4. 실험 및 성능 평가



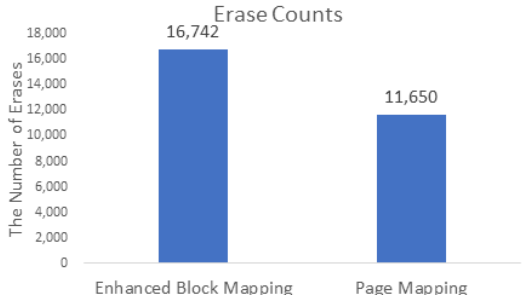
[Fig. 7] Write Counts with Various Buffer Size

제안하는 일괄처리 기반 블록 매핑 기법의 성능을 평가하기 위해 최적화된 성능을 보여주는 페이지 매핑 기법과 성능을 비교하는 시뮬레이션을 진행하였다. 실험에서 블록당 페이지의 수는 256개로 하였고, 저장장치의 용량은 400GB로 설정하였다. 실험에 사용한 데이터는 엔터프라이즈 환경에서 추출된 데이터 입출력 패턴[15]이다. 또한, 버퍼의 크기를 정하기 위해 버퍼 블록의 개수를 조정하며 Fig. 7과 같은 평가를 수행하였다. 블록이 증가할수록 이전 성능과 비교하여 20.13% 0.7 ~ 1.35% 향상되었다. 이 중, 가장 최적화된 블록의 개수는 2개였다. 따라서 버퍼 블록의 개수는 2로 정했다.



[Fig. 8] Write Performance

Fig. 8은 페이지 매핑 기법과 비교한 쓰기 성능의 결과이다. 쓰기 횟수는 총 4,273,815회 발생하였고, 이 결과는 3,801,819회 발생한 페이지 매핑 기법과 비교하여 약 11.04%의 성능 감소가 발생하였다.



[Fig. 9] Erase Performance

Fig. 9는 지우기 성능의 결과이다. 지우기 횟수는 총 16,742회 발생하였고, 이 결과는 11,650회 발생한 페이지 매핑 기법과 비교하여 약 30.41%의 성능 차이가 발생하였다. 실험을 통해 일반적으로 페이지 매핑 기법과 비교하여 많은 성능 차이가 발생하는 블록 매핑을 최대 약 30.41% 이하로 성능의 차이를 개선할 수 있음을 확인하였다. 이러한 원인은 버퍼를 통해 일괄처리하는 방식이 성능 향상에 영향을 줄 수 있는 것으로 분석되었다.

5. 결론

본 논문에서는 대용량 플래시 메모리 기반 저장장치에서 매핑 공간의 오버헤드를 효과적으로 줄이기 위한 향상된 블록 매핑 기법을 제안하였다. 이 방법은 매핑 정보를 유지하기 위한 메모리 자원을 최적화할 수 있는 블록 매핑 기법을 기반으로 동작한다. 또한, 버퍼 블록에 데이터를 모아놓고, 블록 단위로 일괄처리하여 성능을 개선할 수 있도록 하였다. 실험을 통해 제안하는 방법이 페이지 매핑과 비교하여도 성능을 약 30% 이내로 줄일 수 있음을 확인하였다. 향후에는 버퍼 블록에서 커밋 방법을 개선하여 성능을 향상하는 방법을 연구할 예정이다.

REFERENCES

[1] H.S.Lee, "A Prediction-Based Data Read Ahead Policy using Decision Tree for improving the performance of NAND flash memory based storage devices," *Journal of Internet of Things and Convergence*,

Vol.8, No.4, pp.9-15, 2022.

[2] H.S.Lee, "A Safety IO Throttling Method Inducting Differential End of Life to Improving the Reliability of Big Data Maintenance in the SSD based RAID," *Journal of Digital Convergence*, Vol.20, No.5, pp.593-598, 2022.

[3] H.S.Lee, "Performance analysis and prediction through various over-provision on NAND flash memory based storage," *Journal of Digital Convergence*, Vol.20, No.3, pp.343-348, 2022.

[4] H.S.Lee, "High Efficiency Life Prediction and Exception Processing Method of NAND Flash Memory-based Storage using Gradient Descent Method," *Journal of Internet of Things and Convergence*, Vol.11, No.11, pp.44-50, 2021.

[5] H.S.Lee, "A Memory Mapping Technique to Reduce Data Retrieval Cost in the Storage Consisting of Multi Memories," *Journal of Internet of Things and Convergence*, Vol.9, No.1, pp.19-24, 2023.

[6] H.S.Lee, "A Study on the Performance Measurement and Analysis on the Virtual Memory based FTL Policy through the Changing Map Data Resource," *Journal of Internet of Things and Convergence*, Vol.9, No.1, pp.71-76, 2023.

[7] Q.He, G.Bian, W.Zhang and Zhen.Li, "RTFTL: design and implementation of real-time FTL algorithm for flash memory," The Journal of Supercomputing, Vol.78, pp.18959-18993, 2022.

[8] J.H.Park, D.J.Park, T.S.Chung and S.W.Lee, "Journal of Parallel and Distributed Computing," Electronics, Vol.10, No.3, 2021.

[9] L.Su, M.Lin, J.Zhang and Y.Pan, "CCFTL: A novel continuity compressed page-level flash address mapping method for SSDs," Journal of Parallel and Distributed Computing, Vol.191, 2024.

[10] J.Sun, S.Li, Y.Sun, Chao.Sun, D.Vucinic and J.Huang, "LeaFTL: A Learning-Based Flash Translation Layer for Solid-State Drives," ASPLOS 2023, Vol.2, pp.442-456, 2023.

[11] H.S.Lee, "An Efficient Resource Optimization Method for Provisioning on Flash Memory-Based Storage," *Journal of Internet of Things and Convergence*, Vol.9, No.4, pp.9-14, 2023.

[12] H.S.Lee, "A Study on Characteristics and Techniques that Affect Data Integrity for Digital Forensic on Flash Memory-Based Storage Devices,"*Journal of Internet of Things and Convergence*, Vol.9, No.3, pp.7-12, 2023.

[13] J.W.Fowler and L.Monch, "A survey of scheduling with parallel batch (p-batch) processing," European Journal of Operational Research, Vol.298, pp.1-24, 2022.

[14] D.K.Shukla, D.Kumar and D.S.Kushwaha, "An efficient tasks scheduling algorithm for batch processing heterogeneous cloud environment," International Journal of Advanced Intelligence Paradigms, Vol.23, No.1, pp.203-216, 2022.

[15] SNIA, http://iotta.snia.org/traces/block-io/388.

이 현 섭(Hyun-Seob Lee) [종신회원]

- 2013년 2월 : 한양대학교 컴퓨터공학과 (공학 박사)
- 2012년 3월 ~ 2021년 2월 : 삼성전자 책임연구원
- 2021년 3월 ~ 현재 : 백석대학교 컴퓨터공학부 조교수

〈관심분야〉
인공지능, 저장시스템, 임베디드 시스템