사물인터넷융복합논문지 Vol. 8, No. 4, pp. 9-15, 2022 https://doi.org/10.20465/KIOTS.2022.8.4.009

낸드 플래시 메모리 기반 저장 장치의 성능 향상을 위해 결정트리를 이용한 예측 기반 데이터 미리 읽기 정책

이현섭*
백석대학교 첨단IT학부 교수

A Prediction-Based Data Read Ahead Policy using Decision Tree for improving the performance of NAND flash memory based storage devices

Hyun-Seob Lee*
Professor, Division of Advenced IT, Baekseok University

요 약 낸드 플래시 메모리는 저전력 소비와 빠른 데이터 처리 속도 때문에 다양한 저장 장치의 미디어로 사용되고 있다. 그러나 데이터의 읽기 처리 속도가 쓰기 처리 속도와 비교하여 약 10배 빠른 비대칭 속도의 특징이 있기 때문에 속도차이를 개선하기 위한 다양한 연구가 진행되고 있다. 특히 플래시 전용 버퍼 관리 정책은 대부분 쓰기 속도를 개선하기 위해 연구되어 왔다. 그러나 최근에 다양한 목적으로 사용되고 있는 플래시 메모리로 구성된 SSD(solid state disk)는 쓰기 성능보다 읽기 성능에 취약한 문제가 있다. 본 논문에서는 낸드 플래시 메모리로 구성된 SSD에서 쓰기 성능보다 읽기 성능이 더 좋지 않은 이유를 밝히고 이를 개선하기 위한 버퍼 관리 정책을 연구한다. 본 논문에서 제안하는 버퍼 관리 정책은 읽기 데이터의 패턴을 분석하고 미래에 요청될 데이터를 낸드 플래시 메모리에서 미리 읽어두는 정책을 적용하여 플래시 기반 저장 장치의 속도를 개선하는 방법을 제안한다. 또한, 시뮬레이션을 통해 미리 읽기 정책의 효과를 증명한다.

주제어 : 낸드 플래시 메모리, 저장시스템, 패턴 분석, 버퍼 관리 정책, 미리 읽기

Abstract NAND flash memory is used as a medium for various storage devices due to its high data processing speed with low power consumption. However, since the read processing speed of data is about 10 times faster than the write processing speed, various studies are being conducted to improve the speed difference. In particular, flash dedicated buffer management policies have been studied to improve write speed. However, SSD(solid state disks), which has recently been used for various purposes, is more vulnerable to read performance than write performance. In this paper, we find out why read performance is slower than write performance in SSD composed of NAND flash memory and study buffer management policies to improve it. The buffer management policy proposed in this paper proposes a method of improving the speed of a flash-based storage device by analyzing the pattern of read data and applying a policy of pre-reading data to be requested in the future from NAND flash memory. It also proves the effectiveness of the read-ahead policy through simulation.

Key Words : NAND flash memory, storage system, pattern analysis, buffer management policy, read ahead

*이 논문은 2022학년도 백석대학교 학술연구비 지원을 받아 작성되었음
*교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2022년 6월 7일 수정일 2022년 7월 18일 심사완료일 2022년 7월 22일

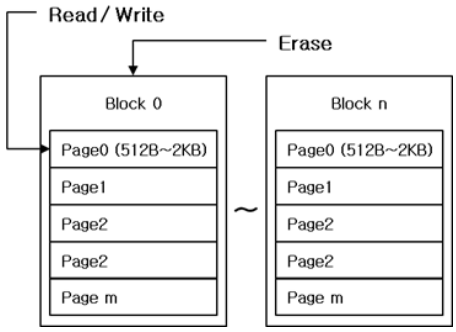
1. 서론

최근 데이터 생산량의 급격한 증가는 대용량 저장장치 기술의 발전을 이끌어왔고, 이와 못지않게 대용량 데이터를 고속으로 읽기 위한 성능향상의 요구사항이 지속해서 대두되고 있다. 이러한 성능향상의 요구를 맞추기 위해 과거 대부분의 저장시스템에서 사용해오던 저장 시스템들은 새로운 고속 성능의 저장장치로 빠르게 대체되고 있다. 특히 고속 저장장치의 미디어로 사용되고 있는 낸드 플래시 메모리는 저전력 소비와 빠른 데이터 처리 속도 때문에 다양한 분야의 저장 장치에서 활용되고 있다. 그러나 비대칭의 데이터 읽기 쓰기 처리 속도의 독특한 특징이 있다.[1, 2] 일반적으로 플래시 메모리의 읽기와 쓰기 속도의 차이는 약 1:10 비율의 차이가 있기 때문에 읽기 처리가 지연되더라도 쓰기 처리를 개선하기 위한 다양한 연구가 진행되어 왔다. 특히 플래시 메모리 전용 버퍼 관리 정책은 대부분 쓰기 처리 성능을 개선하기 위해 연구되어 왔다. 그러나 최근에는 낸드 플래시 메모리로 구성된 SSD(solid state disk)에서 쓰기 속도보다 읽기 속도에 취약한 특징을 보인다. 따라서 플래시 메모리 기반 저장장치에서도 읽기 속도를 개선하기 위한 버퍼 관리 정책의 연구가 필요하다. 본 논문에서는 SSD에서 읽기 속도보다 쓰기 속도가 느린 이유를 밝히고 이를 개선하기 위한 버퍼 관리 정책을 연구한다. 본 논문에서 제안하는 버퍼 관리 정책은 읽기 데이터의 패턴을 분석하고 미래에 요청될 데이터를 낸드 플래시 메모리에서 미리 읽어두는 정책을 적용하여 플래시 기반 저장 장치의 속도를 개선하는 방법을 제안한다. 또한 시뮬레이션을 통해 미리 읽기 정책의 효과를 증명한다.

2. 배경

2.1 낸드 플래시 메모리의 구조와 특징

Fig.1은 낸드 플래시 메모리의 구조를 보여주고 있다. 그림과 같이 낸드 플래시메모리는 여러 개의 블록으로 구성이 되어 있고 각 블록은 여러 개의 페이지로 구성이 되어 있다. 플래시 메모리의 주요 특징은 데이터를 쓰기 전 해당 공간을 지워야 하는(erase-before-write) 것과 읽기/쓰기 동작은 페이지 단위로 처리되고 지우기 동작은 블록 단위로 처리되는 것이다. 이러한 특징을 감추기 위해 일반적으로 FTL(flash transfer layer)[3-7]을 사용한다. FTL은 논리적인 주소와 물리적인 주소를 맵핑하

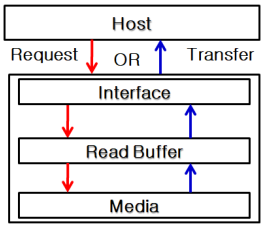


[Fig. 1] Structure of NAND Flash Memory

는 방식으로 이러한 특징을 감춘다. 또 한 가지의 특징은 읽기 쓰기 지우기 동작의 시간적 비용이 약 1:10:100 정도의 차이가 있다는 점이다. 따라서 그동안 연구되어 온 대부분의 플래시 메모리용 버퍼 관리 정책은 읽기 성능이 저하되어도 쓰기(지우기) 성능을 개선하기 위한 연구가 진행되어 왔다.

2.2 엔터프라이즈 환경에서 저장장치의 읽기 처리

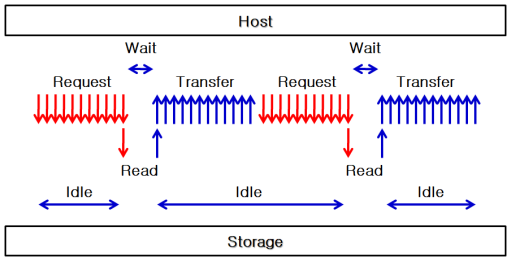
플래시 메모리의 기술 발전으로 과거에 사용하던 미디어보다 빠른 데이터 속도를 제공하는 SSD는 엔터프라이즈 서버와 데이터 센터의 데이터 저장장치로 활용되고 있다. 또한 기술의 급격한 성능 향상으로 내부 데이터 처리 속도가 호스트와 데이터 전송을 담당하는 인터페이스 모듈의 성능을 앞지르게 되어 인터페이스의 성능을 추가로 개선하는 새로운 연구[8-10]가 진행되고 있다.



[Fig. 2] Processing Read Request on the Storage

Fig.2는 저장장치의 읽기 처리 과정을 보여주고 있다. 그림과 같이 저장장치는 호스트와 SATA, SAS, NVME와 같은 프로토콜을 통해 데이터 처리에 대한 요청받아서 수행한 후 호스트로 처리된 데이터를 전송하는 구조로 설계되어 있다. 그림에서는 호스트에서 요청받은 읽기 요청을 수행하기 위해 인터페이스가 요청받아 저장장치의 내부 미디어로부터 버퍼로 데이터를 읽은 후 인터페이스가 다시 호스트로 DMA를 통해 전송하는 예제를

보여주고 있다. 그런데, 최근에는 미디어로부터 데이터를 읽는 속도가 향상되어, 이전의 인터페이스 모듈을 사용할 경우 데이터를 호스트로 전송하는 인터페이스에서 성능 병목현상을 불러일으킬 수 있다.



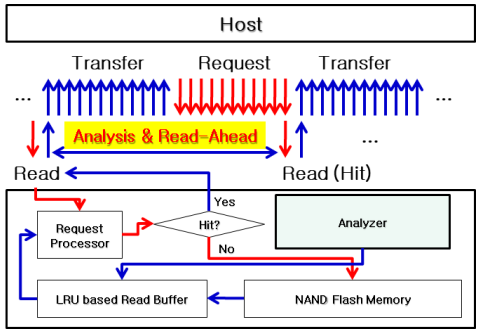
[Fig. 3] Limitation of Processing

Fig.3은 고속 데이터 처리가 가능한 저장장치에서 느린 전송속도로 인해 발생하는 한계점을 보여주고 있다. 저장장치에서는 데이터 전송과 데이터 처리는 독립된 하드웨어 모듈로 구성되어 있기 때문에 데이터를 전송하면서 내부적으로 데이터를 처리할 수 있다. 그러나 이러한 특징을 고려하지 않고 동기화된 데이터 처리하면, 플래시 메모리 기반 저장장치와 같이 고속의 데이터 읽기 처리가 가능한 저장장치의 경우 저속의 프로토콜을 사용하는 시스템에서 호스트와 저장장치 간에 데이터전송 속도보다 내부 데이터 처리 속도가 빠르기 때문에 유휴시간이 발생할 수 있다. 즉 그림의 예제와 같이 호스트와 저장장치 사이에서 호스트는 저장장치로부터 데이터 전송을 기다려야 하는 전송지연 시간이 발생하고 저장장치는 데이터 전송이 끝나고 호스트로부터 다음 요청받아서 새로운 작업을 수행하기까지 응답을 기다려야 하는 시간이 발생할 수 있다.

3. 데이터 패턴 학습 기반 예측 데이터 미리 읽기 정책

3.1 예측 데이터 미리 읽기 정책

본 논문에서는 제안하는 데이터 패턴 학습 기반 예측 데이터 미리 읽기 정책은 읽기 요청을 호스트로부터 받아 처리할 때 요청된 데이터를 호스트로 전송하는 동안 분석기를 통해 학습된 다음 주소의 데이터를 미리 읽어두는 것이다. 그다음 호스트로부터 새로운 요청을 받았을 때 미리 읽어둔 데이터가 일치할 경우 전송지연 시간 없이 미리 읽어 두었던 데이터를 호스트로 전송한다.

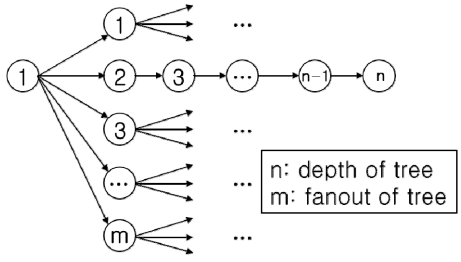


[Fig. 4] Predictive Data Read Policy

Fig.4는 논문에서 제안하는 예측 데이터 미리 읽기 정책의 예제이다. 그림의 예제와 같이 호스트에서 요청받은 읽기 요청을 처리할 때 만약 데이터가 미리 읽어두었던 데이터와 일치하면 호스트로 데이터 전송을 수행한다. 그러나 만약 일치하지 않을 경우 미디어로부터 데이터를 읽어서 호스트에 전송한다. 그리고 미리 읽어둔 데이터의 적중과 관계없이 분석기의 학습 데이터를 갱신하고 학습된 다음 읽기 요청이 들어올 확률이 높은 데이터를 미리 읽어두는 과정을 반복한다.

3.2 결정트리 기반 데이터 패턴 분석기

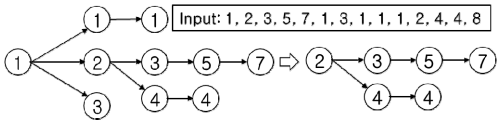
본 논문에서 제안하는 데이터 패턴 분석기는 결정트리(decision tree)[11-13] 기술을 기반으로 호스트로부터 전송된 데이터의 주소 패턴으로부터 관계성을 학습하고, 이를 기반으로 이전에 요청된 데이터를 전송하는 동안 호스트의 다음 읽기 요청을 예측한다.



[Fig. 5] Data Pattern Analyzer

Fig.5는 결정트리 기반 데이터 패턴 분석기를 보여주고 있다. 그림과 같이 결정 트리는 특정 주소로부터 시작하여 m개의 데이터 가짓수로 확산되어 n개의 길이까지 유지하는 형태로 관리된다. 최초 주소에서 n개의 데이터는 순차적으로 기록이 되고 이후 새로운 데이터 패턴은

연결된 데이터로부터 최초 중복이 없는 n개의 데이터 패턴으로부터 가지가 확산되도록 삽입이 된다. 만약 새로운 주소에 대해 연결된 데이터 패턴이 없으면 최초 데이터를 제거하고 남아있는 가지 데이터 중 가장 긴 주소 패턴이 첫 번째 주소부터 자리를 차지하게 된다.



[Fig. 6] Example of Analyzer

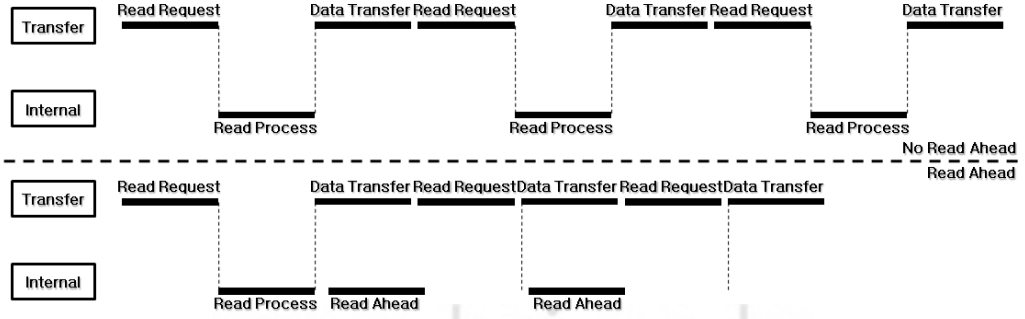
Fig.6는 데이터 패턴의 종류와 깊이인 m과 n을 각각 5로 설정한 패턴 분석기의 예제이다. 그림에서 Input의 순서대로 페이지 번호에 대한 읽기 요청이 들어왔을 때 최초 호스트의 요청에 따른 페이지 주소 1, 2, 3, 5, 7의 패턴은 중복이 없는 순서대로 하나의 패턴을 만들었다. 이후 요청된 주소의 순서대로 1, 3에 대한 패턴과 1, 1, 1에 대한 패턴 3. 4. 4에 대한 패턴을 추가하였다. 이후 새로운 주소 8이 사용되었을 때 이것을 처리할 수 없기 때문에 최초의 데이터 1과 가지 패턴이 제거되고 가지 패턴 중 길이가 가장 긴 2번 패턴이 주요 패턴으로 변경된다. 단, 요청된 n개의 패턴이 모두 순차적인 경우 이후 들어오는 패턴은 순차적인 패턴으로 판단한다. 본 논문에서는 n개의 패턴을 200으로 가정하였다.

3.3 예제를 통한 미리 읽기 정책 성능 비교

Fig.7은 예측 미리 읽기 정책을 적용하였을 때 이 정책을 적용하지 않은 일반적인 관리정책과 비교하여 성능이 어떻게 변화하는지를 예제를 이용하여 보여주고 있다. 그림의 예제에서는 중간의 경계선을 중심으로 미리 읽기 정책을 적용한 경우와 그렇지 않은 경우를 구분하였고 각각 하드웨어적으로 분리 가능한 전송시간의 오버헤드와 내부 동작의 오버헤드를 구분하여 표기하였다. 그리고 편의상 읽기 요청과 데이터전송 그리고 읽기요청 처리를 모두 동일한 단위의 수행시간으로 가정하였다. 마지막으로 미리 읽기 정책의 경우 첫 번째 읽기 처리 이후 모두 미리 읽기 데이터와 요청한 데이터가 동일한 것을 가정하였다. 예제의 결과와 같이 3번의 읽기 요청을 처리하는 동안 미리 읽기 정책을 적용한 방법은 7단계의 동작이 필요했고, 미리 읽기 정책을 적용하지 않은 방법은 9단계의 동작이 필요했다. 즉 미리 읽기 정책을 적용한 경우 적중률이 올라가고 적중이 반복될수록 성능이 향상되는 것을 확인할 수 있다.

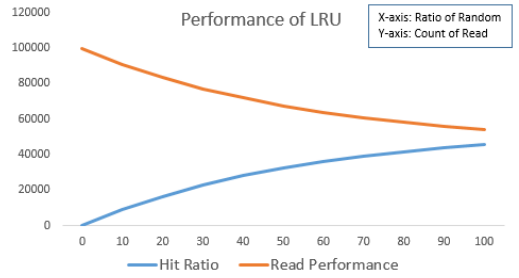
4. 실험 및 평가

본 논문에서는 연구의 우수성을 평가하기 위해 다양한 분야에서 일반적으로 사용되고 있는 LRU(least recently used)[14, 15] 기법과 비교하여 실험 평가하였다. 또한, 연구 효과의 분석을 위해 결정트리를 기본으로 사용하는 미리 읽기 정책(dTree)과 LRU 기반 정책에 dTree를 융합 한 정책(dTree based LRU)을 실험 및 분석하였다. 실험에서는 다양한 패턴의 데이터를 실험하기 위해 랜덤한 패턴과 순차적인 패턴의 비율을 나타내는 RR (ratio of random) 값을 사용하였다. 이 값은 0에 가까울수록 데이터의 패턴이 순차적이고 100에 가까울수록 랜덤한 패턴의 데이터 전송 요청을 의미한다. 예를 들어 RR이 100인 경우 실험에서 사용된 모든 데이터의 패턴이 랜덤한 것을 의미한다. 반면 RR이 0인 경우 모든 데이터가 순차적인 것을 의미한다. 만약 RR이 50인 경우 실험에서 사용된 절반의 데이터는 순차적인 패턴을 가지고 있고 나머지는 랜덤한 패턴의 데이터를 가지고 있는



[Fig. 7] Example of Comparison

것을 의미한다. 본 논문에서는 실험의 객관적인 평가를 위해 여러 RR 값 비율의 환경에서 일반적으로 많이 사용되고 있는 LRU(least recently used) 기법과 비교 실험을 진행하였다. 실험에서는 100,000개의 데이터 세트를 사용하였고, 하나의 데이터당 512B를 가정하였다. 또한 버퍼의 크기는 2MB로 하였고, 분석할 수 있는 데이터 패턴의 종류와 깊이인 m과 n을 각각 4,096으로 하였다. 그리고 호스트에서 한 번에 요청되는 데이터 크기를 2MB라고 하였고, 데이터 전송 시간은 2MB를 전송하는 시간과 동일하다고 가정하였고 이 시간 단위를 100us로 가정하였다.

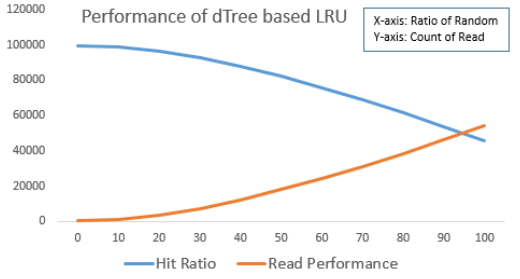


[Fig. 8] Performance of LRU

Fig.8은 RR에 따라 변화하는 다양한 데이터 패턴에서 LRU의 적중률과 읽기 성능을 보여주고 있다. 그림에서 RR이 0인 경우 모든 데이터가 순차적인 것을 의미하고, LRU의 특성상 적중률이 0이기 때문에 성능의 장점이 없다. 따라서 10만 개의 데이터에 대한 읽기 요청에 대해 10만 번의 읽기 연산을 수행해야 했다. RR이 100인 경우 모든 데이터가 중복을 허용하는 랜덤한 순서이다. 이 경우 적중률이 약 45.55% 발생했고, 10만 번의 읽기 요청에 대해 약 54450회의 읽기 연산을 수행했다. 마지막으로 RR이 50인 경우 적중률은 약 32% 발생했고, 67,661회의 읽기 연산을 수행했다.

[Fig. 9] Performance of dTree

Fig.9는 10만 개의 데이터를 처리하는 동안 RR에 따른 dTree의 적중률과 읽기 성능을 보여주고 있다. 그림의 결과와 같이 dTree는 순차적 패턴을 인식하여 미리 읽기 정책을 수행하기 때문에 RR이 0인 경우 99.8%의 적중률을 보여주었으나, RR이 100인 경우 0%의 적중률을 보여주었다. RR에 따라 200~100,000회의 읽기 동작을 수행하였다.

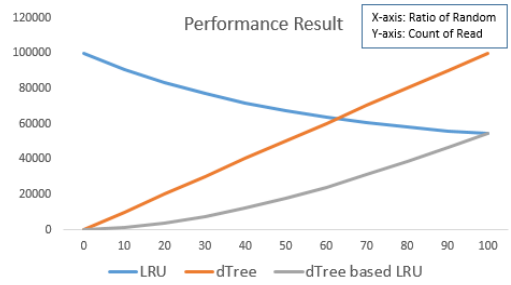


[Fig. 10] Performance of dTree based LRU

Fig.10은 LRU based dTree 정책의 결과를 보여주고 있다. 그림의 결과와 같이 RR에 따라 적중률은 99.8%~45.55%를 보였고, 읽기 동작은 200~54,449회 수행하였다.

[Fig. 11] Hit Ratio

Fig.11은 실험에서 사용한 각 정책의 적중률을 비교하였다. RR이 0인 경우, 결정트리를 이용한 미리 읽기 동작을 수행한 dTree와 LRU based dTree가 기본적인 LRU보다 99.8%의 우수한 적중률을 보였다. RR이 100인 경우 LRU를 기반으로 하는 정책이 LRU를 사용하지 않는 정책 대비 45.55%의 적중률을 보였다. 결과적으로 논문에서 제안한 LRU와 dTree 기반 미리 읽기 정책을 융합한 정책이 데이터 패턴과 관계 없이 우수한 적중률을 보였다.



[Fig. 12] Performance Result

Fig.12는 각 정책의 읽기성능 결과이다. 그림의 결과와 같이 RR이 0인 경우 LRU based dTree 정책이 LRU보다 약 98.8% 우수한 성능을 보였고, RR이 100인 경우 기본적인 결정트리를 적용한 정책보다 45.55% 성능이 향상되는 것을 보였다. 즉 본 논문에서 제안하는 결정트리를 LRU와 융합한 방법이 데이터의 패턴과 관계 없이 평균적으로 우수한 성능을 보이는 것을 확인 하였다.

5. 결론

본 논문에서는 저장장치와 호스트 간에 데이터 전송이 이루어지는 유휴시간에 예측 기반 미리 읽기를 통해 읽기 성능을 향상시키는 방법을 제안하였다. 또한 결정트리를 기반으로 패턴을 분석기를 제안하였고, 이를 LRU에 적용하여 다양한 데이터 패턴에서 안정적인 성능을 보이는 것을 증명하였다. 그러나 실험에서 사용한 데이터는 다양한 패턴을 생성하기 위해 융합된 데이터이다. 따라서 랜덤한 패턴에서는 제안한 데이터 분석기로 패턴을 분석할 수 없는 한계가 있다. 따라서 종합적인 성능이 LRU보다 우수하지만 랜덤한 데이터 패턴이 증가할수록 성능의 증가폭이 감소하고 100% 랜덤한 패턴에서는 LRU의 성능을 넘어설 수 없는 한계가 있다. 향후 리얼 데이터를 이용한 패턴 분석 방법을 연구하여 랜덤한 패턴에서도 우수한 성능을 보이는 패턴 분석 방법을 연구할 예정이다.

REFERENCES

[1] M.K.Kim, I.J.Kim and J.S.Lee, "CMOS-compatible ferroelectric NAND flash memory for high-density, low-power, and high-speed three-dimensional memory," *Science Advances*, Vol.7, No.3, 2021.

[2] P.Kumari, U.Surendranathan, M.Wasiolek, K. Hattar, N.P.Bhat and B.Ray, "Radiation-Induced Error Mitigation by Read-Retry Technique for MLC 3-D NAND Flash Memory," *IEEE Transactions on Nuclear Science*, Vol.68, No.5, 1032-1039, 2021.

[3] S.S.Chae, R.Mativenga, J.Y.Paik, M.Attique, and T.S.Chung, "DSFTL: An efficient FTL for flash memory based storage systems." *Electronics* Vol.9, No.1, pp.145, 2020.

[4] W.Xie, Y.Chen, and P.C.Poth, "ASA-FTL: An adaptive separation aware flash translation layer for solid state drives," *Parallel Computing*, Vol.61, pp.3-17, 2017.

[5] I.B.Zion, "Key-value FTL over open channel SSD," *12th ACM International Conference on Systems and Storage*. pp.192-192, 2020.

[6] S.Kim and Y.Son, "Optimizing Key-Value Stores for Flash-Based SSDs via Key Reshaping," *IEEE Access 9*, pp.115135-115144, 2021.

[7] J.H.Park, D.J.Park, T.S.Chung, and S.W.Lee, "A Crash Recovery Scheme for a Hybrid Mapping FTL in NAND Flash Storage Devices," *Electronics*, Vol.10, No.3, pp.1-20, 2021.

[8] H.Litz, J.Gon, A.Klimovic, and C.Kozyrakis, "RAIL: Predictable, Low Tail Latency for NVMe Flash," *ACM Transactions on Storage.*, Vol.18, No.5, pp.1-21, 2022.

[9] Y.Zou, A.Awad, and M.Lin, "DirectNVM: Hardware-accelerated NVMe SSDs for High-performance Embedded Computing," *ACM Transactions on Embedded Computing Systems*, Vol.21, No.9, pp.1-24, 2022.

[10] S.Kim, H.Park, and J.Choi, "Direct-Virtio: A New Direct Virtualized I/O Framework for NVMe SSDs," *Electronics*, Vol.10, No.17, pp.1-12, 2021.

[11] C.S.Lee, P.Y.S.Cheang, and M.Moslehpour, "Predictive Analytics in Business Analytics: Decision Tree," *Advances in Decision Sciences*, Vol.26, pp.1-29, 2022.

[12] B.Charbuty, and A.Adnan, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, Vol.2, No.1, pp.20-28, 2021.

[13] G.Pappalardo, S.Cafiso, A.D.Graziano, and A.Severino, "Decision Tree Method to Analyze the Performance of Lane Support Systems," Sustainability, Vol.13, No.2, pp.1-13, 2021.

[14] Q.Zheng, T.Yang, Y.Kan, X.Tan, J.Yang, and X.Jiang, "On the Analysis of Cache Invalidation With LRU Replacement," *IEEE Transactions on Parallel and Distributed Systems*, Vol.33, No.3, pp.654-666, 2022.

[15] A.A.Titinchi and N.Halasa, "FPGA implementation of simplified Fuzzy LRU replacement algorithm," *16th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp.657-662, 2019.

이 현 섭(Hyun-Seob Lee) [종신회원]

- 2007년 2월 : 한양대학교 컴퓨터공학과 (공학 석사)
- 2013년 2월 : 한양대학교 컴퓨터공학과 (공학 박사)
- 2012년 3월 ~ 2021년 2월 : 삼성전자 책임연구원
- 2021년 3월 ~ 현재 : 백석대학교 첨단IT학부 조교수

〈관심분야〉
인공지능, 저장시스템, 임베디드 시스템