멥 데이터 자원 변화를 통한 가상 메모리 기반 FTL 정책의 성능 측정 및 분석 연구

이현섭*
백석대학교 첨단IT학부 교수

A Study on the Performance Measurement and Analysis on the Virtual Memory based FTL Policy through the Changing Map Data Resource

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

요 약 최근 빅데이터를 저장 및 관리하기 위해 대용량 데이터를 안정적으로 접근할 수 있는 고성능의 저장시스템 개발과 연구가 활발하게 진행되고 있다. 특히 데이터센터 및 엔터프라이즈 환경의 저장시스템에서는 대용량의 데이터를 관리하기 위해 대용량의 SSD(solid state disk)가 대량으로 사용되고 있다. 일반적으로 SSD는 미디어인 NAND 플래시 메모리의 특성을 감추고 데이터를 관리를 효율적으로 하기 위해 FTL(flash transfer layer)을 사용한다. 그러나 FTL의 알고리즘은 SSD의 용량이 커질수록 데이터가 저장된 NAND의 위치 정보를 관리하기 위해 DRAM을 많이 사용하는 한계가 있다. 따라서 본 논문에서는 FTL에서 사용하는 DRAM 자원을 줄이기 위한 가상 메모리 (virtual memory)를 적용한 FTL 정책을 소개한다. 본 논문에서 제안하는 가상 메모리 기반 FTL 정책은 LRU(least recently used) 정책을 사용하여 최근 사용된 데이터의 멥핑 정보를 DRAM 공간에 적재하고 이전에 사용된 정보는 NAND에 저장하는 방식으로 멥 데이터를 관리한다. 마지막으로 실험을 통해 가상 메모리 기반의 FTL과 일반 FTL의 데이터 쓰기 처리를 하는 동안 소모되는 성능과 자원의 사용량을 측정하고 분석한다.

주제어 : 낸드 플래시 메모리, 저장시스템, 가상 메모리, 플래시 전환 계층, 빅데이터

Abstract Recently, in order to store and manage big data, research and development of a high-performance storage system capable of stably accessing large data have been actively conducted. In particular, storage systems in data centers and enterprise environments use large amounts of SSD (solid state disk) to manage large amounts of data. In general, SSD uses FTL(flash transfer layer) to hide the characteristics of NAND flash memory, which is a medium, and to efficiently manage data. However, FTL's algorithm has a limitation in using DRAM more to manage the location information of NAND where data is stored as the capacity of SSD increases. Therefore, this paper introduces FTL policies that apply virtual memory to reduce DRAM resources used in FTL. The virtual memory-based FTL policy proposed in this paper manages the map data by using LRU (least recently used) policy to load the mapping information of the recently used data into the DRAM space and store the previously used information in NAND. Finally, through experiments, performance and resource usage consumed during data write processing of virtual memory-based FTL and general FTL are measured and analyzed.

Key Words : NAND flash memory, storage system, virtual memory, flash transfer layer, big data

*이 논문은 2022학년도 백석대학교 학술연구비 지원을 받아 작성되었음
*교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2022년 10월 25일 수정일 2022년 11월 29일 심사완료일 2022년 12월 2일

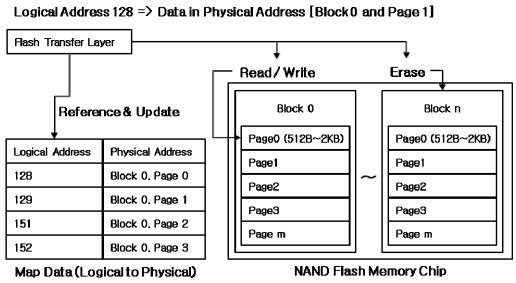
1. 서론

최근 데이터센터 및 엔터프라이즈 환경의 산업과 연구기관에서는 급속도로 증가하고 있는 빅데이터를 안정적으로 저장하고 관리하기 위한 고성능의 저장시스템 기술에 관한 개발과 연구가 활발하게 진행하고 있다.[1-4] NAND 플래시 메모리는 저전력으로 고속의 데이터 접근이 가능한 장점이 있어서 고속의 성능이 필요한 대용량 저장장치 미디어로 활용되고 있다. 그러나 읽기/쓰기 동작과 지우기 동작을 수행할 때 비대칭의 처리 단위로 동작해야 하는 특징이 있고, 데이터 블록을 쓰기 전 지우기 연산을 수행해야 하는 물리적인 특성 (erase-before-write)이 있기 때문에 일반적인 저장 미디어와는 다른 관리 정책이 필요하다.[5-8] 따라서 대표적인 NAND 플래시 메모리 기반 저장장치인 SSD는 NAND 플래시 메모리의 특성을 감추고 데이터를 효율적으로 관리하기 위해 FTL을 사용한다.[9-11] FTL은 멥핑 기법을 이용하여 이러한 특성을 감춘다. 그러나 FTL은 저장장치의 용량이 커질수록 증가하는 멥핑 정보를 관리하기 위한 DRAM 자원이 증가하는 한계가 있다. 결과적으로 이러한 한계는 SSD의 용량이 고용량화될수록 더 많은 자원을 사용해야 하는 문제를 발생시킨다.. 본 논문에서는 FTL에서 사용하는 DRAM 자원을 줄이기 위한 가상 메모리 (virtual memory)[12-14]를 적용한 FTL 정책을 소개한다. 본 논문에서 제안하는 가상 메모리 기반 FTL 정책은 LRU(least recently used)[15, 16] 정책을 사용하여 최근 사용된 데이터의 멥핑 정보를 DRAM 공간에 적재하고 이전에 사용된 정보는 NAND에 저장하는 방식으로 멥 데이터를 관리한다. 마지막으로 실험을 통해 가상 메모리 기반의 FTL과 일반 FTL의 데이터 쓰기 처리를 하는 동안 소모되는 성능과 자원의 사용량을 측정하여 분석한다.

2. 배경

2.1 낸드 플래시 메모리의 구조와 특징

Fig. 1은 낸드 플래시 메모리의 구조와 FTL의 역할을 보여주고 있다. 그림과 같이 NAND 플래시 메모리는 n개의 블록으로 구성되어 있고 각 블록은 m개의 페이지를 가지고 있다. NAND 플래시 메모리의 특징 중 한 가지는 데이터가 있는 페이지에 새로운 정보를 추가로 변경할 때에는 이전 데이터를 지우고 다시 써야 하는 특성

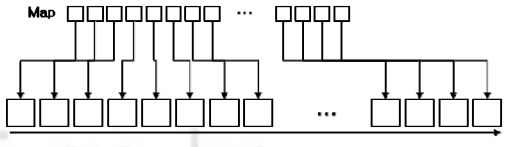


[Fig. 1] Architecture of NAND flash memory

이 있다. 이러한 특성을 쓰기 전 지우기라고 부른다. 그런데 읽기 쓰기 단위는 페이지 단위로 처리되고, 지우기 단위는 블록 단위로 실행된다. 따라서 하나의 페이지를 업데이트해야 할 경우에도 블록 내에 모든 페이지를 함께 지우고 다시 써야 하는 문제점이 있다. 이러한 문제를 해결하기 위해 FTL이 사용되고 있으며 FTL은 논리적인 주소를 물리적인 주소로 멥핑하는 방법을 통해 페이지의 업데이트가 발생해도 대량의 데이터 수정 문제를 지연시킨다. 예를 들어 그림에서 논리적인 주소 129는 물리적인 주소 0번 블록의 1번 페이지에 멥핑되어 있다. 그런데 129번 주소의 데이터가 변경될 경우 FTL은 새로운 물리적인 주소를 할당하여 비어있는 페이지에 업데이트된 데이터를 기록하고 멥핑 정보를 업데이트한다. 만약 0번 블록의 4번 페이지에 업데이트했을 경우 129번 주소의 멥핑 정보는 0번 블록의 4번 페이지로 변경하는 것으로 처리할 수 있기 때문에 대량의 페이지 이동이 발생하지 않는다.

2.2 FTL 알고리즘의 멥 데이터

일반적으로 FTL의 알고리즘은 블록 멥핑 방법과 페이지 멥핑 방법 두 가지로 구분된다. 블록 멥핑은 논리적인 블록을 물리적인 블록으로 멥핑하는 방법이고 페이지 멥핑 방법은 논리적인 페이지를 물리적인 페이지 단위로 멥핑하는 방법이다. 멥핑 단위의 기본적인 크기 차이 때문에 페이지 멥핑 방식이 블록 멥핑의 방식보다 더 큰 멥핑 정보를 유지해야 하는 자원의 한계가 있으나 성능에



[Fig. 2] Map increases as capacity increases

더 최적화된 장점이 있다. 따라서 최근의 SSD 기술은 성능을 향상하기 위해 페이지 멥핑을 기반으로 개발되고 있다. 그러나 페이지 멥핑 방식은 대용량의 빅데이터를 관리하기 위한 대용량의 저장장치 개발을 할 때 더 많은 멥핑 정보를 유지해야 하는 한계가 있다. Fig. 2는 용량 증가에 따른 멥 크기의 증가를 그림으로 보여주고 있다. 그림과 같이 저장장치의 용량이 증가할 경우 페이지들을 멥핑하기 위한 멥의 정보는 같은 비율로 증가하게 되며 페이지 멥핑의 경우 더 많은 DRAM 자원 공간이 필요하다.

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

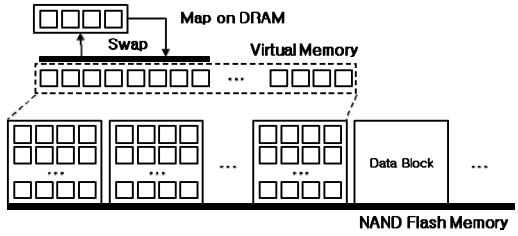
[Fig. 3] Analysis of the map size of 2TB

Fig. 3은 2TB 크기의 저장장치에서 멥 데이터의 의 크기를 분석한 결과이다. 멥 데이터의 각 엔트리는 논리적인 주소에 대한 물리적인 주소를 표현한다. 각 주소는 정수 형태로 관리되는데 일반적으로 하나의 정수형 변수를 유지하기 위해서는 32비트의 공간이 필요하다. 이 공간은 8개의 바이트이고 16진수를 기준으로 0에서 FFFFFFFF의 값까지 표현할 수 있다. 32 Bit(4B)의 최대 값인 FFFFFFFF은 십진수 기준으로 4,294,967,295를 의미한다. 그리고 하나의 페이지가 512B라고 가정한 경우 멥데이터에서 각 엔트리가 표현할 수 있는 최대 주소의 범위는 2TB(2,199,023,255,040 = 4,294,967,295 × 512)이다. 즉 현재의 컴퓨팅 환경과 정수 체제에서 구현할 수 있는 최대 저장장치의 크기는 한계가 있다. 이를 극복하기 위해 페이지의 크기를 2K에서 16K까지 증가하는 방법으로 멥핑의 최대 크기를 32TB까지 확장하는 노력을 하고 있다. 그러나 페이지의 크기를 확장하는 것은 제한이 있기 때문에 32TB를 넘어선 저장공간을 위해서는 멥의 각 엔트리 크기를 64 Bit(8B) 체계로 증가해야 하며, 이 경우 저장장치의 용량 증가와 함께 자원의 폭발적인 증가로 이어지는 문제점 발생할 수 있다.

3. 가상 메모리 기반 FTL 정책

3.1 핵심 아이디어

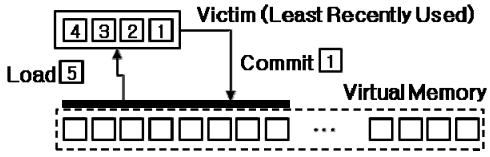
Fig. 4는 본 논문에서 제안하는 가상블록 기반 FTL의 핵심 아이디어를 보여주고 있다. 그림의 예제에서는 NAND 플래시 메모리에서 일부 블록을 멥 데이터를을 유지하기 위한 전용 블록으로 할당하였다. 그리고 멥이 모두 올라온 상태를 가상 메모리로 멥핑하였다. 마지막으로 가상 메모리의 멥에서 실제로 필요한 멥데이터를 선별하여 DRAM의 멥 영역에 로드하고 사용하지 않는 멥데이터는 가상 메모리를 통해 NAND 플래시 메모리에 커밋하는 방법을 통해 멥 데이터를 관리한다.



[Fig. 4] Key idea

3.2 LRU 기반 Swap 정책

본 논문에서 제안하는 알고리즘은 LRU를 기반으로 가상 메모리의 자원을 관리한다.

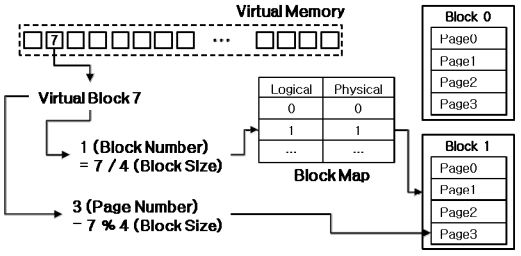


[Fig. 5] LRU based swap policy

Fig. 5는 본 논문에서 제안하는 LRU 기반 Swap 정책을 보여주고 있다. 그림의 예제에서는 DRAM에 최대 4개의 가상 메모리 블록을 적재할 수 있다고 가정하였다. 그리고 가상 메모리로부터 1부터 4까지 4개의 가상 메모리 블록을 DRAM에 적재해 놓은 상태다. 이때 새로운 가상 메모리 블록 5를 로드해야 할 경우 적재된 메모리 블록에서 가장 오래전에 사용되었던 1번 블록을 선택하여 가상 메모리로 커밋한다.

3.3 Block Mapping 기반 가상 메모리 관리 정책

NAND 플래시 메모리는 일반적인 저장장치의 저장 미디어와는 다른 특징이 있기 때문에 FTL과 같은 별도의 모듈을 통해 데이터 블록과 NAND 블록을 멥핑한다. 본 논문에서는 가상 메모리를 NAND 플래시 메모리 블록과 블록 멥핑을 하기 위해 블록 멥핑 기반의 알고리즘을 적용하였다.



[Fig. 6] Virtual memory block management policy

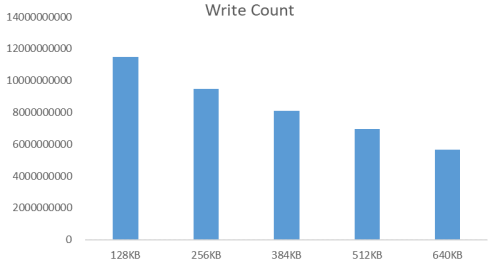
Fig. 6은 본 논문에서 제안하는 블록 멥핑 기반 가상 메모리 관리 정책을 보여주고 있다. 그림과 같이 멥핑을 관리하는 블록 멥에서는 가상 메모리의 논리적 블록 주소를 계산하여 연관된 물리적 블록의 주소를 유지 및 관리한다. 예를 들어 가상 메모리 블록 7의 논리적인 블록 주소는 물리적인 블록의 크기로 나눈 몫으로 계산한다. 그림의 예제에서는 물리적인 블록은 4개의 페이지로 구성되어 있다. 따라서 논리적인 블록 주소는 가상 메모리 블록 7을 4로 나눈 결과인 1이다. 그리고 논리적 블록 주소 1은 블록 멥을 통해 물리적 블록 1에 멥핑되어 있는 것을 확인할 수 있다. 그리고 블록 내 페이지의 위치는 7을 4로 나눈 나머지 결과인 3이다. 즉 가상 메모리 블록 7은 NAND 플래시 메모리의 1번 블록 내에 3번 페이지와 멥핑 된다.

4. 실험 및 평가

본 논문에서는 FTL의 멥데이터를 유지하기위해 필요한 DRAM 자원을 줄이기 위해 가상 메모리 정책을 적용하였다. 이 방법은 DRAM 자원을 줄일 수 있지만 동시에 가상 메모리 블록을 관리하기 위한 추가적인 비용이 발생한다. 본 절에서는 증가하는 DRAM 자원과 환경에서 변화하는 성능을 시뮬레이션을 통해 측정한다. 실험을 위해 저장장치의 미디어인 NAND 플래시 메모리에서 각 페이지 크기는 1024B이고 하나의 블록당 128개의 페이지로 구성하였다. 그리고 총 160MB의 NAND 플래시 메모리 기반 저장장치를 사용하였다. 약 1,000GB에 대한 데이터를 랜덤한 주소의 순서로 쓰기 연산을 수행했다. 실험에서 사용한 NAND 플래시 메모리 블록은 하나의 페이지당 1024B이다. 따라서 페이지당 4B의 멥 데이터를 256개 유지할 수 있다. 이것은 하나의 블록당 128개의 페이지로 구성된 블록 환경에서 블록당 32,768개의 멥 데이터를 유지할 수 있음을 의미한다. 또한, 이 계산의 결과는 하나의 블록당 32MB의 데이터에 대한 멥 데이터를 유지할 수 있음을 의미한다. 따라서 160MB를 유지하기 위한 멥 데이터는 5개의 블록이 필요하고, 실험에서는 가상 메모리 관리를 위한 블록 멥핑 정책을 적용하기 위해 5개의 멥 데이터 블록과 1개의 교체 블록을 사용했다. 즉 1개의 멥 블록을 적재하기 위한 최소 DRAM 자원은 128KB이고 5개의 멥 블록을 적재하기 위해 최대 DRAM 자원은 640KB이다. 따라서 FTL의 멥을 위한 DRAM 자원은 128K에서 640K까지 128K 단위로 변경하며 쓰기 연산 데이터를 처리하는 동안 저장장치의 내부에서 발생하는 읽기, 쓰기, 지우기 및 소모된 시간 성능을 측정하였다.

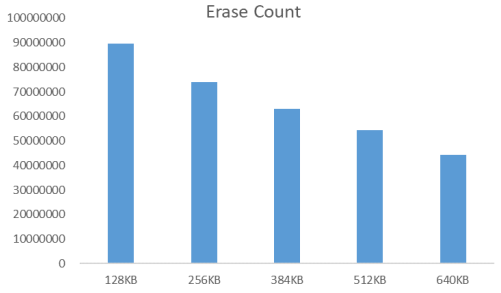
[Fig. 7] Read performance

Fig. 7은 실험의 읽기 성능 결과를 보여주고 있다. 실험에서는 DRAM 자원을 변경해가서 쓰기 연산을 처리하는 동안 발생하는 내부 읽기 요청 횟수를 측정하였다. DRAM 자원이 128KB일 때 읽기 요청 횟수는 총 11,365,325,215회 발생했다. 그리고 256KB일 때 9,356,072,752회 발생했다. DRAM 자원이 두 배 증가했을 때 읽기 성능은 약 17.68% 상승하였다. 그리고 DRAM 자원을 5배 증가시킨 640KB를 사용했을 때 약 51.09% 상승하였다.



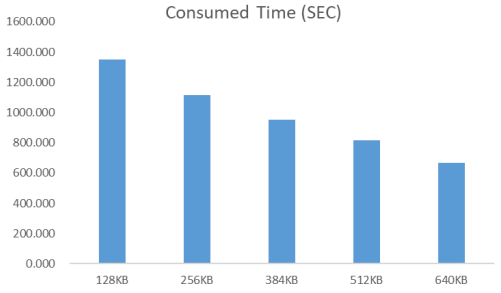
[Fig. 8] Write performance

Fig. 8은 실험의 쓰기 성능의 결과를 보여주고 있다. 그림의 결과 같이 DRAM 자원이 128KB에서 640KB로 증가하는 동안 쓰기 횟수는 11,470,182,815회에서 5,663,481,123회로 감소하였고 17.52%에서 50.62%까지 향상되었다.



[Fig. 9] Erase performance

Fig. 9는 실험의 지우기 성능의 결과를 보여주고 있다. 그림의 결과 같이 DRAM 자원이 128KB에서 640KB로 증가하는 동안 지우기 횟수는 89,573,839회에서 44,208,985회로 감소하였고 17.52%에서 50.65%까지 향상되었다.



[Fig. 10] Consumed time

Fig.10은 실험을 통해 100GB의 데이터를 쓰기 처리하는 동안 소비된 시간 결과를 보여주고 있다. 그림의 결과 같이 DRAM 자원이 128KB일 때 소모된 시간은 약 1350.24초였다. 그리고 640KB일 때 666.14초였다. DRAM 자원이 증가하는 동안 전체적으로 성능이 17.53%에서 50.67% 향상되었다. 실험을 통해 가상 메모리 정책을 적용한 FTL의 정책이 DRAM 자원을 최대 80%까지 절약할 수 있는 것을 확인하였다. 그러나 이 경우 성능은 50%까지 감소하는 것을 확인하였다. 따라서 FTL에서 멥데이터를 줄이는 것은 성능과 자원 사이의 절충(trade-off) 관계인 것을 확인할 수 있었다.

5. 결론

본 논문에서는 가상 메모리 블록 기반 FTL 정책을 제안하였다. 제안하는 핵심 아이디어는 가상 메모리 정책을 사용하여 멥 데이터 중 필요한 데이터만 DRAM에 유지하고 현재 사용하지 않는 멥데이터는 NAND 플래시 메모리와 멥핑되어 있는 가상 메모리에서 관리하는 것이다. 또한, 가상 메모리 블록은 블록 멥핑 정책을 통해 NAND 플래시 메모리 블록에서 관리하였고 DRAM에 유지되는 멥 데이터는 LRU 정책을 기반으로 관리하는 정책을 제안하였다. 또한, 분석과 실험을 통해 멥 데이터를 유지하기 위한 DRAM의 자원을 최대 80%까지 줄일 수 있음을 증명하였다. 그러나 멥 데이터 자원을 관리하기 위해 NAND 플래시 메모리를 사용해야 하는 오버헤드가 있고 실험에서 최대 약 50%의 성능 저하가 발생할 수 있음을 확인하였다. 앞으로는 DRAM 자원을 줄이고 성능 저하를 줄이는 최적화 연구를 수행할 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," The Korea Internet of Things Society, 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," The Society of Digital Policy & Management, 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," The Society of Digital Policy & Management, Vol.20, No.3, pp.343-348, 2022.

[4] H.S.Lee, "A method for optimizing lifetime prediction of a storage device using the frequency of occurrence of defects in NAND flash memory," The Korea Internet of Things Society, Vol.7, No.4, pp.9-14, 2021.

[5] 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.

[6] 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, pp.1032-1039, 2021.

[7] K.Parat and A.Goda, "Scaling Trends in NAND Flash",

2018 IEEE International Electron Devices Meeting (IEDM), pp.211-214, 2018.

[8] G.H.Lee, S.M.Hwang, J.S.Yu and H.J.Kim, "Architecture and Process Integration Overview of 3D NAND Flash Technologies," Open Access Vo.11, No.15, pp.6703, 2021.

[9] 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.

[10] 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.

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

[12] D.Skalatos, A.Kokolis, T.Xu, and J.Torrellas, "Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism," ASPLOS '20: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, pp.1093-1108, 2020.

[13] D.Ganguly, Z.Zhang, Yang and R.Melhem, "Interplay between hardware prefetcher and page eviction policy in CPU-GPU unified virtual memory," ISCA '19: Proceedings of the 46th International Symposium on Computer Architecture, pp.224-235, 2019.

[14] D.Mishra and P.Kulkarni, "A survey of memory management techniques in virtualized systems," Computer Science Review, Vol.29, pp.56-73, 2018.

[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.

[16] 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.

이 현 섭(Hyun-Seob Lee) [종신회원]

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

〈관심분야〉
인공지능, 저장시스템, 임베디드 시스템