Management Policy of Hybrid Buffer Memory for Performance Improvement of Cloud Solid State Drives

Cheol-Woo Jung†/Researcher · Sungho Kim‡/Researcher ·
Sang-Ho Hwang†/Researcher · Chang-Hyeon Park*/Professor
†ICT R&D Division, Gyeongbuk Institute of IT Convergence Industry Technology (GITC)
‡Future Automotive Research Division, Gyeongbuk Institute of IT Convergence Industry Technology (GITC)
*Dept. of Computer Engineering, Yeungnam University

클라우드 SSD 성능 향상을 위한 하이브리드 버퍼 메모리 관리 정책

정철우†/연구원 · 김성호‡/연구원 · 황상호†/연구원 · 박창현*/교수
†(재)경북IT융합산업기술원 ICT연구본부 · ‡미래차연구본부 · *영남대학교 컴퓨터공학부

Abstract

In this paper, we proposes CTP-SSD, an SSD-based hybrid buffer memory policy for managing large amounts of information data in cloud-based server systems. The proposed CTP-SSD was designed with three policy directions: DRAM characteristics, buffer characteristics, and operation patterns for the characteristics of the hybrid buffer memory and SSD. Based on this design direction, CTP-SSD proposes five procedures based on page allocation, pattern tracking, victim page selection, and DRAM/NVMs migration strategies. Through performance evaluation, CTP-SSD reduced the number of write operations on SSD by 19.23 times, 19.21 times, 19.09 times, and 9.34 times compared to LRU, CLOCK, CLOCK-DNV, and AC-CLOCK. Through these results, CTP-SSD was experimentally proven to improve SSD performance compared to other policies.

Keywords : Solid State Drive, Garbage Collector, Flash write reduction, Hybrid buffer, Non-volatile memories

† To whom correspondence should be addressed.
Corresponding Author: park@yu.ac.kr

Received 10 June 2024; Revised 21 June 2024; Accepted 28 June 2024

1. 서 론

클라우드 시스템은 사물인터넷(IoT, Internet of Things), 센서, PC, 서버 등 폭발적인 시스템 활용 증대로 인해 고성능, 대용량에 대한 사용자 요구가 증대되고 있는 추세이다[1-2]. 최근 구축중인 클라우드 시스템에서는 많은 사용자들이 데이터를 접근하고 저장하는 특성을 가지고 있어 고성능에 대한 요구가 증가하고 있다. 이러한 고성능 이슈를 해결하기 위해 클라우드 시스템에서는 기존의 하드디스크(HDD, Hard Disk Drive)의 속도 문제를 비휘발성 메모리 기반의 SSD(Solid State Drive) 활용한 많은 연구가 진행 중에 있다[3]. SSD는 대용량, 높은 집적도 등의 특성을 가지고 있는 낸드 플래시 메모리(NAND flash memory) 소자를 활용하여 여러 개의 셀 단위를 하나의 단위로 묶어 구성하는 저장 장치이다[4].

그러나 SSD는 기존의 HDD와 비교하여 세 가지 치명적인 단점이 존재한다[5]. 첫 번째, SSD는 쓰기 연산을 수행한 후 반드시 소거 연산을 수행해야 한다. 즉 제자리에 덮어쓰기가 불가능한 특성이 있다. 두 번째, SSD의 연산 속도는 쓰기, 읽기, 소거 연산 별 상이하다. 세 번째, SSD의 수명은 HDD와 비교하여 현저하게 낮다. 이러한 문제를 플래시 전환 계층(FTL, Flash Translation Layer) 또는 SSD 전용 파일시스템을 활용하여 해결하고자 노력하였다[6].

최근에는 SSD의 주요 단점을 버퍼 메모리에서 해결하고자 하는 많은 연구를 진행 중에 있다[7]. 기존의 SSD 버퍼 메모리는 주로 DRAM(Dynamic Random Access Memory)을 활용하였다. DRAM은 낮은 비용과 높은 집적도로 오랜 기간 동안 캐시, 메인 메모리, 버퍼 메모리 등 다양한 메모리 분야에서 폭넓게 활용하고 있다[8]. 이러한 장점에도 불구하고 DRAM은 집적도 문제로 인해 대용량화가 어려워, 기존의 많은 연구자들은 대용량화가 적합한 비휘발성 메모리(NVMs, non-volatile memories)를 활용한 다양한 연구가 진행 중에 있다[9].

비휘발성 메모리 버퍼는 대용량화가 적합한 메모리의 장점이 존재하는 반면, DRAM 대비 연산속도 측면에서 낮은 성능을 보여준다는 단점이 있다. 이러한 문제점을 해결하고자, 기존 연구자들은 DRAM과 비휘발성 메모리를 결합한 하이브리드 버퍼를 제안하였다[10]. 하이브리드 버퍼는 DRAM 낮은 비용과 높은 성능의 장점과 비휘발성 메모리에 데이터 저장할 수 있는 장점을 가질 수 있다. 기존의 대표적인 하이브리드 버퍼 정책은 CLOCK-DNV(clock with DRAM and NVM hybrid write buffer)이다[10]. CLOCK-DNV는 하이브리드 버퍼를 효과적으로 활용하기 위해 더티비트가 발생한 페이지들을 병합하여 DRAM으로 이주시킨다. 이후 DRAM에서 공간이 부족할 경우 CLOCK-DNV는 비휘발성 메모리로 이주하는 정책을 수행한다.

그러나 CLOCK-DNV는 하이브리드 버퍼의 특성을 고려하지 않고 설계하여 비효율적인 측면이 있다. 예를 들어, CLOCK-DNV는 DRAM의 공간이 부족할 경우 참조 비트와 더티 비트만을 활용하여 비휘발성 메모리에 이주 여부를 결정한다. 즉 최근에 접근한 패턴만을 고려하여 이주를 하는 측면이 있어, 낮은 용량의 하이브리드 버퍼에 비효율적인 측면이 존재한다.

따라서 본 논문에서는 하이브리드 메인 메모리에서 고안된 AC-CLOCK(adaptive classification CLOCK)을 하이브리드 버퍼에 적용하고자 한다[10]. AC-CLOCK은 하이브리드 메인 메모리에 최적화하여 설계하였기 때문에 하이브리드 버퍼(DRAM, 비휘발성 메모리)와 SSD 특징을 고려하여 새롭게 설계할 필요가 있다.

첫 번째, SSD는 하이브리드 버퍼에서 수행하는 연산과 비교하여 매우 긴 쓰기 지연시간이 발생한다. 즉 SSD에서의 쓰기 연산은 시스템 전반에 지연시간을 증가시킬 수 있다.

두 번째, 하이브리드 버퍼는 DRAM과 비휘발성 메모리 결합한 구조로 연산 별 지연시간 차이가 존재한다. 따라서 하이브리드 버퍼에서는 메모리 별 연산 지연시간 차이로 인해 시스템 성능 저하에 지대한 영향이 발생할 수 있다.

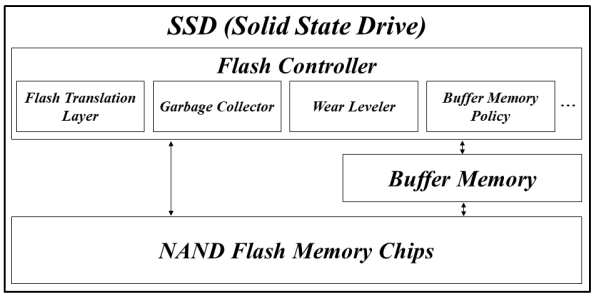
세 번째, 클라우드 서버 시스템에서는 데이터 저장함에 있어 시간 및 공간에 대한 지역적인 특성을 가지고 있다. 즉 이러한 특성은 하이브리드 버퍼 및 SSD 연산 수행 횟수에 영향을 미치고, 결국 시스템 전반에 영향을 미칠 수 있다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 배경지식과 관련연구에 대해 설명한다. 제안하는 기법은 3장에서 상세하게 서술하고, 4장에서는 기법에 대한 실험환경 설정 및 성능 비교 분석을 진행한다. 마지막 5장에서는 본 논문의 결론을 맺는다.

2. 배경지식 및 관련연구

2.1 배경지식

SSD는 크게 플래시 컨트롤러(flash controller), 낸드 플래시 메모리 칩, 버퍼 메모리로 구성한다. 그림 1은 SSD의 구조를 보여주고 있다.



[Figure 1] Architecture of SSD

SSD에서 낸드 플래시 메모리 칩은 실제 읽기와 쓰기 연산을 수행하는 주체로 페이지가 최소 단위이다. 낸드플래시 메모리는 하드 디스크와 다르게 소거 연산(erase operation)이라는 추가적인 연산이 존재한다[6]. 이는 최소 연산 단위인 페이지 단위에 쓰기 연산을 수행한 이후 바로 쓰기 연산을 하지 못하기 때문에 소거 연산이 반드시 수행되어야 하며, 이를 제자리 덮어쓰기 문제라고 한다. 또한 소거 연산은 최소 연산 단위인 페이지의 묶음인 블록 단위로 구성되어 있어, 기존 데이터들을 이동시키고 연산을 수행해야 한다. 따라서 낸드 플래시 메모리 칩은 기존 하드 디스크와는 다르게 낮은 쓰기와 읽기 지연시간의 장점이 있으나, 추가적인 문제에 대한 해결책이 필요하며, 이러한 역할을 플래시 컨트롤러가 수행한다.

버퍼 메모리는 시스템의 메인 메모리와 낸드 플래시 메모리 칩에 읽기와 쓰기 연산을 최종적으로 수행하기 전 버퍼 역할을 수행한다. 일반적으로 버퍼 메모리는 DRAM을 사용한다[10]. 최근 SSD에서는 작은 버퍼 메모리 용량을 극복하고자 DRAM과 비휘발성 메모리를 결합한 하이브리드 버퍼 메모리에 대한 연구가 활발하게 진행 중이며, 이를 시스템에 적용하고자 하는 많은 연구가 진행 중에 있다[10-11].

플래시 컨트롤러는 다수의 낸드 플래시 메모리 칩과 버퍼 메모리를 관리하고 인터페이스 역할을 한다. SSD 플래시 컨트롤러는 크게 네 가지로 구성되어 있다.

첫 번째, 플래시 전환 계층은 낸드 플래시 메모리 칩에서 제자리 덮어쓰기 문제를 해결하기 위해 컨트롤러에서 관리하는 논리 주소와 낸드 플래시 메모리 칩에서 관리하는 물리 주소를 상호 매핑 한다.

두 번째, 가비지 컬렉터(Garbage Collector)는 플래시 전환 계층에서 관리하는 논리 주소와 물리 주소를 매핑 하는 과정에서 과거에 사용하여 현재 사용하지 않고, 새로운 공간을 확보한다. 이때 낸드 플래시 메모리 칩에서는 고유 연산인 소거 연산을 수행한다.

세 번째, 마모도 평준화(Wear Leveler)는 낸드 플래시 메모리 칩 내 모든 공간을 고르게 수명을 마모시켜 수명 연장한다. 낸드 플래시 메모리 칩은 모든 한계 수명을 활용하기 위해 고르게 마모를 시키기 위해 수행한다.

마지막 네 번째는 버퍼 메모리 관리 정책이다. 이 정책은 메인 메모리와 낸드 플래시 메모리 칩에 연산을 수행하기 전 단계로 활용함으로써 SSD 전반의 읽기와 쓰기 연산속도를 향상시킬 수 있다.

다음 표 1은 SSD를 구성하는 DRAM, 비휘발성 메

모리, 낸드 플래시 메모리 칩의 특성이다. 표 1에서 낸드 플래시 메모리 칩, 비휘발성 메모리, DRAM 순으로 내구성과 집적도 차이를 확인할 수 있다. 특히 주목해 볼만한 사실은 DRAM과 비휘발성 메모리는 읽기 지연시간이 유사한 반면에 쓰기 지연시간이 7배 정도 차이가 발생함을 알 수 있다. 기존의 연구자들은 이와 같은 특성을 SSD에서 버퍼 메모리를 하이브리드로 활용하는 구조로 착안했다[12].

<Table 1> Characteristics of memories in SSD

Description		DRAM	NVMs	NAND flash
operation latency	read	50ns	50ns	100us
	write	50ns	350ns	2.4ms
	earse	-	-	3.0ms
density		very low	low	high
duration		-	108	103

2.2 관련연구

이 절에서는 메모리 정책들에 대해 서술한다. 메모리 관리 정책에서 가장 대표적인 정책들은 LRU(Least Recently Used)와 CLOCK이 존재한다[8].

LRU는 버퍼 메모리에서 가장 오랫동안 참조되지 않은 페이지들을 공간이 부족할 경우 SSD로 교체 연산을 수행한다. 해당 정책에서는 페이지들의 가장 오랫동안 참조하지 않은 페이지들을 관리하기 위한 리스트가 필요하므로 소프트웨어적으로 접근 가능한 영역에서 적은 크기의 버퍼에 주로 활용한다.

CLOCK은 모든 페이지들에 접근 여부만을 확인할 수 있는 참조 비트를 활용하여 접근 여부를 확인하고, 공간이 부족할 경우 시계 방향으로 회전하면서 참조 비트가 설정되지 않은 페이지를 쓰기 연산을 통해 SSD로 내보낸다. 해당 정책은 LRU와 달리 소프트웨어, 하드웨어 둘 다 설계가 가능한 특성이 있으며 별도에 리스트 관리 연산을 수행하지 않아도 되기 때문에 대용량 버퍼 메모리나 메인 메모리에서 주로 활용한다.

BPLRU(Block Padding Least Recently Used)는 SSD에서 발생하는 가비지 컬렉터 발생 횟수를 줄이기 위한 정책이다[13]. 이 정책은 가비지 컬렉터 횟수를 줄이기 위에 버퍼 메모리에서 가장 많은 페이지를 보유한 블록들을 우선적으로 SSD로 내보낸다. 이때 내보내는 블록 중에 버퍼 메모리에 존재하지 않은 페이지들은 SSD에서 읽기 연산을 통해 새로운 블록을 만들어 내보내며 이를 블록 패딩(block padding) 기법이라고 한다. 해당 정책은 기존 버퍼 메모리를 기준으로 설계되어 있어 하이브리드 버퍼에서는 제대로 활용하기 힘들며, 단독 기법이라기보다는 구성 요소로 활용하기에 용이하다.

AC-CLOCK은 메인 메모리에서 하이브리드 메모리를 적용한 정책이다[10]. 이 정책은 DRAM과 비휘발성 메모리의 특성을 고려하여, 각기 다른 정책을 수행한다. DRAM 정책은 쓰기 연산의 횟수 및 읽기 연산 여부에 따라 DRAM에 유지할 페이지들인지 비휘발성 메모리에 유지할 페이지인지 판단한 후, DRAM에 공간이 부족할 경우 비휘발성 메모리로 페이지를 이주한다. 비휘발성 메모리에서는 읽기 연산 여부를 판단하여, DRAM과 비휘발성 메모리 둘 다 공간이 부족할 경우 SSD로 내보낸다. 해당 정책은 메인 메모리의 주요 특성에 맞춰 설계한 정책으로 버퍼 메모리에 그대로 적용하기에는 부적합한 측면이 존재한다.

CLOCK-DNV는 2개의 참조 비트(읽기와 쓰기 연산)와 더티 비트(쓰기 연산)로 페이지들을 특성을 구성하고, 하이브리드 버퍼의 특성을 고려하여 각기 다른 CLOCK을 수행한다[11]. 이 정책에서는 DRAM에서 여유 공간이 부족할 경우 더티 비트가 설정한 페이지를 비휘발성 메모리로 이주하고, 더티 비트가 설정되지 않은 비트는 폐기 처리한다. 해당 정책은 더티 비트가 설정된(쓰기 연산이 발생) 페이지들만 하이브리드 메모리 버퍼에 유지하는 특성으로 인하여 단순 쓰기 연산인 작업부하에는 효과가 있으나, 쓰기와 읽기 연산이 존재할 경우 잦은 스와핑 연산으로 인해 성능저하가 발생할 위험성이 크다.

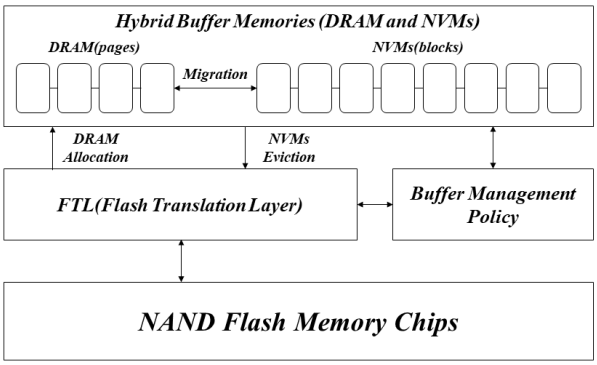
3. 제안 기법

이 장에서는 하이브리드 버퍼에서 메모리 별로 특성을 고려하고 효율적으로 활용할 수 있는 CTP-SSD(Clock with Time Pattern in SSD) 정책을 제안한다. 해당 정책은 크게 두 가지 목적으로 설계한다.

첫 번째, 하이브리드 버퍼에서는 DRAM과 비휘발성 메모리의 특징들을 반영하여 페이지들을 유지할 수 있도록 한다.

두 번째, SSD에서는 많은 부하를 발생시키는 소거 연산에 대한 절감에 초점을 둔다.

제안하는 정책이 위 두 가지 목적에 주목하는 이유는 표 1의 SSD 측면에서 시스템 부하가 가장 크게 영향을 미치는 부분이 낸드 플래시 메모리 칩에서 수행하는 연산(읽기, 쓰기, 소거)이기 때문이다. 따라서 시스템에서는 낸드 플래시 메모리 칩에 대한 부하를 줄일 경우 전체 SSD에 대한 성능 향상이 가능하다. 그림 2는 이러한 목적을 반영하고 있는 CTP-SSD의 구조를 보여주고 있다.



[Figure 2] Architecture of CTP-SSD

그림 2에서 제시한 CTP-SSD는 위에 언급한 설계 목적을 기반으로 다음과 같은 기준으로 정책을 제시한다.

첫 번째, 하이브리드 버퍼에서 DRAM은 쓰기 연산에 대한 낮은 부하를 가지고 있으며 효율성이 증가하는 특성이 있다. 따라서 DRAM은 쓰기 연산에 대한 효율성을 가질 수 있도록 한다.

Algorithm 1: Clock with Time Pattern in SSD

```
input: p // page according to a requested operation
       op // read or write operation
       d_pts // current clock pointer in DRAM
       n_pts // current clock pointer in NVMs
1 if there is no free space in DRAM then
2 |  GetDRAMFreeSpace(d_pts, n_pts);
3 |  ReclaimDRAMSpace(d_pts); // page unit
4 end
5 AllcateToDRAM(p);
6 SetWaitState(p);
```

Algorithm 2: GetDRAMFreeSpace

```
input: d_pts // current clock pointer in DRAM
       n_pts // current clock pointer in NVMs
output: None
1 while there is no free space in DRAM do
2 |  if NVMs migration condition then
3 |  |  MigrateToNVMs(d_pts, n_pts);
4 |  |  break;
5 |  else
6 |  |  ChangePageTypeInDRAM(d_pts);
7 |  |  NextClockPointerInDRAM(d_pts);
8 |  end
9 end
```

두 번째, 시스템은 읽기와 쓰기 연산이 1회 이상 하이브리드 버퍼에 연산이 발생할 경우 최대한 버퍼에서 활용할 수 있도록 하며, 페이지가 활용되지 않을 경우 SSD로 내보낸다.

세 번째, 하이브리드 버퍼에서의 페이지들은 읽기와 쓰기 연산을 수행하는 패턴을 파악하고 패턴에 따라 공간적 분리를 통해 효율성을 가질 수 있도록 한다.

위와 같은 정책적 기준으로 토대로 CTP-SSD는 DRAM과 NVMs의 기준 정책을 판단하기 위해 표 2의 정보를 활용한다.

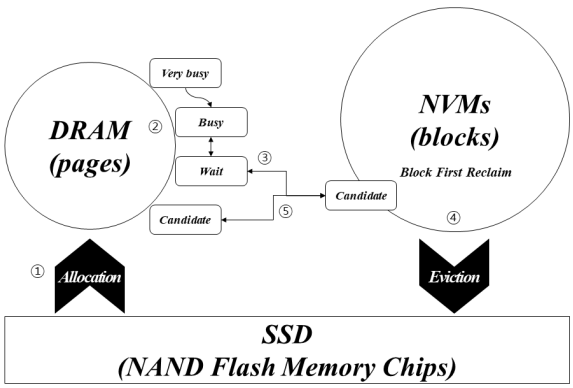
<Table 2> Information of DRAM and NVMs

memory	unit	information
DRAM	page	type bit(2bit), reference bit(1bit)
NVMs	page	dirty bit(1bit), reference bit(1bit)

Algorithm 3: MigrateToNVMs

```
input: d_pts // current clock pointer in DRAM
       n_pts // current clock pointer in NVMs
output: None
1 while there is no free space in NVMs do
2   if reference bit in n_pts is unset then
3     EvictAndReclaimNVMsSpace(n_pts); // block unit
4     break;
5   else
6     UnsetReferenceBitAndDirtybit(n_pts);
7     NextClockPointerInNVMs(n_pts);
8   end
9 end
10 AllcateToNVMs(d_pts);
```

표 2에서 DRAM은 상세한 정보를 판단하고 구분하기 위해 페이지 단위로 type 2bit 정보와 읽기 연산 발생 유무를 판단하는 reference 1bit 정보를 사용한다. NVMs는 페이지 단위로 쓰기 연산 발생 유무를 판단하는 dirty 1bit와 읽기 연산 발생 유무를 판단하는 reference 1bit 정보를 사용한다. 표 2의 정보를 기반으로 CTP-SSD는 그림 3과 같이 동작하며, 제안하는 CTP-SSD의 기본 전략은 다음과 같다.



[Figure 3] The flowchart of CTP-SSD

CTP-SSD는 DRAM에서 쓰기 연산 중점 페이지들을 NVMs에서 읽기 연산 중점 페이지들을 유지한다. 페이지 부재로 인해 하이브리드 버퍼 메모리에 할당할 경우 그 페이지는 시간지역성에 의해 읽기와 쓰기 연산의 발생빈도가 높기 때문에 DRAM에 우선 할당한다. 다시 참조될 가능성이 낮거나 읽기와 쓰기 연산 패턴 변화에 따라 일부 페이지들은 NVMs 이주 또는 SSD로 내보냄으로써, 참조 확률이 높은 다른 페이지들을 위한 공간을 확보한다.

앞서 언급한 기본 전략을 토대로 CTP-SSD는 알고리즘 1-3과 5단계 절차를 통해 세부 과정을 수행한다.

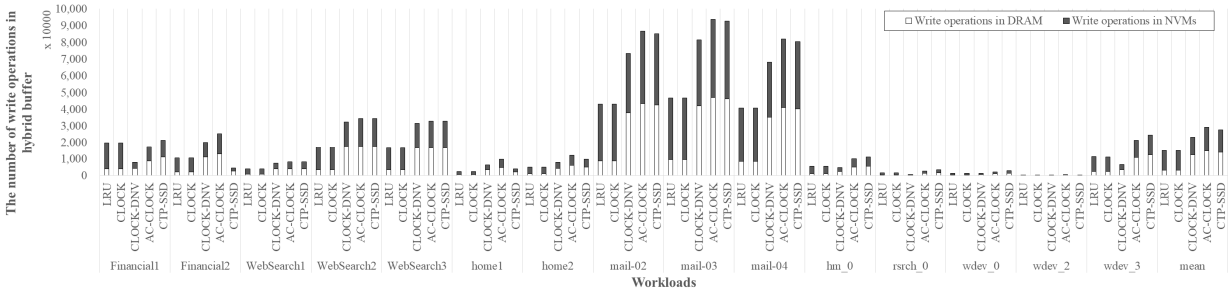
절차 1. 시스템에서 읽기 또는 쓰기 연산으로 인해 페이지 부재가 발생할 경우 CTP-SSD는 DRAM에 페이지를 할당하고 type 상태를 wait 상태로 설정한다. 이는 DRAM이 낮은 연산 비용을 가지고 있어 읽기와 쓰기 연산 중심인 페이지들을 구분하기에 유리하기 때문이다(그림 3 ①, 알고리즘 1 라인 5-6).

절차 2. DRAM에 할당한 페이지들은 표 2를 기반으로 다음 페이지 부재가 발생할 때까지 페이지들에 대한 정보를 기록한다. 기록한 정보는 다음 페이지 부재가 발생할 때 표 3에서 type 조건에 따라 NVMs로 이주한다(그림 3 ②).

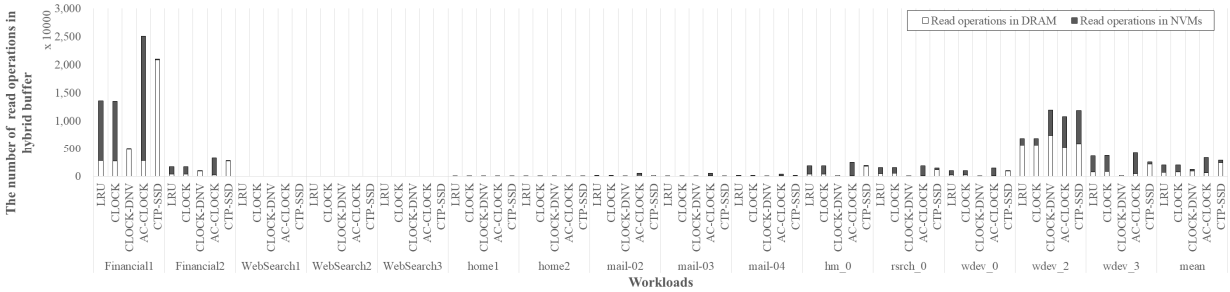
<Table 3> Types of DRAM page

Type(2bit)	Condition	Reference bit
Very busy(3)	DRAM maintenance	-
Busy(2)		
Wait(1)	DRAM maintenance	set
	NVMs migration candidate	unset
Candidate(1)	NVMs migration candidate	-

절차 3. DRAM에 공간이 부족해 페이지 부재가 발생할 경우 CTP-SSD는 표 3의 type 조건 중 NVMs 이주 후보자에 해당할 경우 페이지 이주를 통해 공간을 확보한다. 만약 이주 후보자가 아닌 경우 표 4의 조건에 따라 상태 전이를 수행한다. 이러한 절차를 반복적으로 수행하여 CTP-SSD는 NVMs 이주 페이지를 찾는다(그림 3 ③, 알고리즘 2 1-9).



[Figure 4] The number of write operations in hybrid buffer



[Figure 5] The number of read operations in hybrid buffer

<Table 4> The state diagram in DRAM page types

# of write operations	Description	State diagram
3	frequently write operations	Increase
2		
1	frequently write operations in the past	Conditional decrease (reference bit unset)
0	infrequently write operations	Decrease

절차 4. 절차 3에서 이주할 페이지를 선택하고 NVMs로 이주할 때 NVMs에서 공간이 부족할 경우 reference bit가 참조되지 않은 비트에 해당하는 블록을 SSD로 내보낸다. 이는 인접한 페이지들이 더 이상 읽기 연산이 발생할 가능성이 늦기 때문이다. 이후 NVMs는 DRAM에서 이주한 페이지를 할당한다(그림 3 ④, 알고리즘 3 1-10).

절차 5. 마지막으로 CTP-SSD는 NVMs에서 dirty bit가 1이고, 다시 쓰기 연산이 발생한 경우 DRAM으로 이주한다. 이때 해당 페이지의 reference bit가 0인 경우 candidate 상태로, 반대로 1인 경우 wait 상태로 이주를 수행한다.

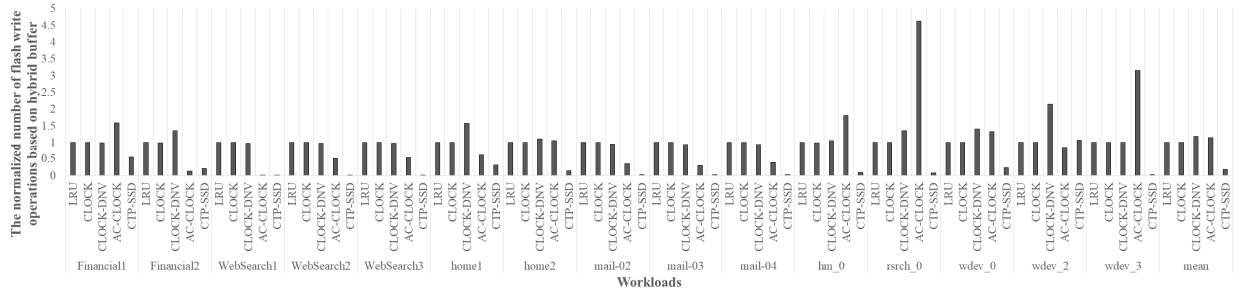
위의 절차를 통해 CTP-SSD는 DRAM에서 쓰기 연산이 자주 발생하는 페이지들을 유지하고, NVMs에서는 읽기 연산이 주로 발생하는 페이지들을 유지한다. 마지막으로 SSD로 내보내는 페이지들은 블록 단위로 구성함으로써 SSD에서 발생하는 소거 연산에 대한 비용을 최소화할 수 있다.

4. 성능 평가

이 장에서는 CTP-SSD의 성능 평가를 진행하기 위해 DiskSim 4.0 기반에 하이브리드 버퍼 모델을 적용한 시뮬레이션을 구성하였다[14-15]. 시뮬레이션에서 사용한 매개변수는 표 5와 같다.

<Table 5> Parameters for the simulator environment

description		value
buffer size	DRAM	6.4MB (20%)
	NVMs	25.6MB (80%)
SSD capacity		20GB
page size		4KB
pages per block		64
garbage collector trigger		6# of free blocks < 20%



[Figure 6] The normalized number of flash write operations based on hybrid buffer

시뮬레이션에 사용한 벤치마크들은 UMass Trace Repository와 FIU(Florida International University), MSRC(Microsoft Research Cambridge)를 사용하였다[16-18]. 시뮬레이션에서 사용한 벤치마크들은 SSD 성능 평가를 측정하고 신뢰성을 확보하기 위해 많은 연구자들이 활용하고 있다. 벤치마크들의 주요특성으로 UMass Trace Repository는 온라인 처리에서 수행하는 어플리케이션 특성을 가지며, FIU와 MSRC는 서버 시스템에서 운영하는 웹, 사용자, 터미널 등 특성을 가지고 있다. 실험 환경에서는 쓰기 연산 비율이 15% 이상이 되는 작업부하를 선정하였으며, 사용한 작업부하 특성은 표 6과 같다.

<Table 6> Characteristics of workloads

workload		ratio of request (%)		avg. request
		read	write	(KB)
UMass	Financial1	23	76	3.28
	Financial2	83	17	2.39
	WebSearch1	0	100	15.14
	WebSearch2	0	100	15.07
	WebSearch3	0	100	15.4
FIU	home1	2	98	6.3
	home2	13	87	12.77
	mail-02	17	83	39.8
	mail-03	13	87	41.53
	mail-04	20	80	41.2
MSRC	hm_0	36	64	7.99
	rsrch_0	10	90	8.92
	wdev_0	21	79	9.07
	web_0	30	70	14.99
	prn_0	11	89	11.09

CTP-SSD의 성능 평가를 진행하기 위해, 본 논문에서는 LRU, CLOCK, CLOCK-DNV, AC-CLOCK과 성능 비교 분석하였다. SSD에서 하이브리드 버퍼의 성능 비교 지표는 하이브리드 버퍼 읽기와 쓰기 연산 횟수와 SSD에서의 쓰기 연산 횟수를 비교 분석한다.

그림 4와 그림 5는 하이브리드 버퍼에의 읽기와 쓰기 연산 횟수를 보여주고 있다. 하이브리드 버퍼에서 읽기와 쓰기 연산 횟수는 하이브리드 버퍼에서 얼마나 많은 연산이 이루어지고 이러한 연산으로 인해 SSD 쓰기 연산 횟수를 감소시킬 수 있는 지표라고 할 수 있다.

제안한 CTP-SSD는 LRU, CLOCK, CLOCK-DNV, AC-CLOCK과 비교하여 하이브리드 버퍼에서의 읽기 연산 횟수를 28.9%, 28.8%, 58.27%, -17.9% 증가시켰고. 쓰기 연산 횟수를 45.6%, 45.7%, 15.6%, -5.8% 증가시켰다. 해당 지표에서는 제안하는 CTP-SSD가 AC-CLOCK에 연산 횟수 성능이 떨어진 것을 확인할 수 있다. 이는 AC-CLOCK 정책이 메인 메모리에서 최대한 유지하는데 효율성을 가졌기 때문에 이러한 결과를 보인 것으로 판단된다.

그림 6은 SSD에서의 쓰기 연산 횟수를 보여주고 있다. 해당 지표는 SSD의 수명 향상 및 지연시간 감소에 중요한 요소라고 할 수 있다. CTP-SSD는 LRU, CLOCK, CLOCK-DNV, AC-CLOCK과 비교하여 SSD에서의 쓰기 연산 횟수를 19.23배, 19.21배, 19.09배, 9.34배 감소시켰다. 이러한 결과는 SSD에서의 고유 특성인 소거 연산을 고려하여 설계하였기에 이러한 결과를 보인 것을 확인할 수 있다.

종합적으로 제안하는 CTP-SSD는 AC-CLOCK 대비 하이브리드 버퍼 메모리에서 읽기와 쓰기 연산의 효율이 낮은 이유가 SSD 소거 연산 비용 감소에 초점을 잡았다고 볼 수 있다. 이는 SSD 전체 시스템 측면에서 SSD 소거 연산에 대한 비용이 크기 때문에 최소한의 trade-off를 통해 성능을 향상한 것이다.

따라서 CTP-SSD는 하이브리드 버퍼에 장점을 가지면서 SSD에서의 주요 성능 지표인 SSD에서의 쓰기 연산 횟수를 줄임으로써 SSD에서의 수명 향상 및 지연시간 감소 기여할 수 있음을 실험 결과를 통해 입증하였다.

5. 결 론

본 논문에서는 클라우드 기반 서버 시스템에서 대용량의 정보 데이터를 관리하기 위한 SSD 기반 하이브리드 버퍼 정책인 CTP-SSD를 제안하였다. CTP-SSD는 하이브리드 버퍼에 대한 특성과 SSD에 대한 특성을 적용하기 위해 DRAM 특성, 버퍼 특성, 연산 패턴의 세 가지의 정책 방향성으로 설계하였다. 이러한 설계 방향을 기반으로 CTP-SSD는 할당, 패턴 추적, 희생 페이지 선정, DRAM과 NVMs 이주 전략을 다섯 가지 절차로 구분하여 정책을 수행한다. 성능 평가를 통해 CTP-SSD는 LRU, CLOCK, CLOCK-DNV, AC-CLOCK과 비교하여 SSD에서의 쓰기 연산 횟수를 19.23배, 19.21배, 19.09배, 9.34배 감소시켰다. 이러한 결과를 통해 CTP-SSD는 타 정책과 비교하여 SSD 성능 향상에 대한 결과를 실험을 통해 입증하였다.

제안하는 정책을 통해 클라우드 기반 서버 구축환경에서는 시스템에 적합한 정책 및 운영체제를 사용하고 이러한 정책들을 통해 클라우드 시스템에서 사용자들에게 서비스 할당, 관리, 활용에 성능적인 효율성을 부여할 수 있을 것으로 기대한다.

Acknowledgement

This work was supported by the Promotion of Innovative Businesses for Regulation-Free Special Zones(P0020333) funded by the Ministry of SMEs and Startups(MSS, Korea)

References

[1] H. S. P. Wong, K. Akarvardar, D. Antoniadis, J, Bokor, C. Hu, T. J. King-Liu, S. Salahuddin, “A Density Metric for Semiconductor Technology,” Proceedings of the IEEE, Vol. 108, No. 4, pp. 478-482, 2020.

[2] G. Yu, J. W. Choi, J. Park, S. Nam, U. Baek, M. S. Kim, “Design and Implementation of AI-based Indoor Autonomous Guide Robot,” Journal of The Korean Institute of Plant Engineering, Vol. 28, No. 2, pp. 43-49, 2023.

[3] K. Zhou, Y. Zhang, P. Huang, H. Wang, Y. Ji, B. Cheng, Y. Liu, "Efficient SSD cache for cloud block storage via leveraging block reuse distances," IEEE Transactions on Parallel and distributed systems, Vol. 31, No. 11, pp. 2496-2509, 2020.

[4] A. Goda, “Recent Progress on 3D NAND Flash Technologies,” Electronics, Vol. 10, No. 24, pp. 3156-3172. 2021.

[5] J. H. Choi, K. M. Kim, J. W. Kwak, “WPA: Write Pattern Aware Hybrid Disk Buffer Management for Improving Lifespan of NAND Flash Memory.” IEEE Transactions on Consumer Electronics, Vol. 66, No. 2, pp. 193-202, 2020.

[6] Y. Luo, M. Lin, "Flash translation layer: a review and bibliometric analysis," International Journal of Intelligent Computing and Cybernetics, Vol. 14,

No. 3, pp. 480-508, 2021.

[7] X. Wang, P. Jin, Adaptive multi-grained buffer management for database systems. Future Internet, Vol. 13, No. 12. pp. 303, 2021.

[8] A. Olgun, H. Hassan, A. G. Yağlıkçı, Y. C. Tuğrul, L. Orosa, H. Luo, O. Mutlu, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023.

[9] S. Kargar, F. Nawab, "Extending the lifetime of NVM: challenges and opportunities," Proceedings of the VLDB Endowment, Vo. 14, No. 12, pp. 3194-3197, 2021.

[10] S. Kim, S. H. Hwang, and J. W. Kwak, "Adaptive-Classification CLOCK: Page replacement policy based on read/write access pattern for hybrid DRAM and PCM main memory," Microprocessors and Microsystems, Vol. 57, pp. 65-75. 2018.

[11] D. H. Kang, S. J. Han, Y. C. Kim, Y. I. Eom, "CLOCK-DNV: a write buffer algorithm for flash storage devices of consumer electronics," IEEE Transactions on Consumer Electronics, Vol. 63, No. 1, pp. 85-91, 2017.

[12] V. Samsung, N. SSD, "860 EVO," 2018.

[13] H. Kim, S. Ahn, "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage," In FAST, Vol. 8, No. 16, pp. 1-14, 2008.

[14] G. Ganger, B. Worthington, Y. Patt, "The DiskSim simulation environment (v4.0) [Online]," Available: http://www.pdl.cmu.edu/DiskSim/Online-document, Parallel Data Lab, 2009.

[15] V. Prabhakaran, T. Wobber, "SSD extension for DiskSim simulation environment," Microsoft Research, 2009.

[16] "OLTP and Websearch Traces form UMass Trace Repository," http://traces.cs.umass.edu.

[17] R. Koller, and R. Rangaswami, "I/O deduplication: Utilizing content similarity to improve I/O performance," ACM Transactions on Storage (TOS), Vol. 6, No. 3, pp. 13-39. 2010.

[18] Storage Networking Industry Association. "MSR Cambridge Traces," http://iotta.snia.org/traces, 2010.