Journal of The Korea Society of Computer and Information
Vol. 24 No. 9, pp. 1-8, September 2019

www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.09.001

Performance Isolation of Shared Space for Virtualized SSD based Storage Systems

Sungho Kim*, Jong Wook Kwak**

Abstract

In this paper, we propose a performance isolation of shared space for virtualized SSD based storage systems, to solve the weakness in a VSSD framework. The proposed scheme adopts a CFQ scheduler and a shared space-based FTL for the fairness and the performance isolation for multiple users on virtualized SSD based storage systems. Using the CFQ scheduler, we ensure SLOs for the storage systems such as a service time, a allocated space, and a IO latency for users on the virtualized storage systems. In addition, to improve a throughput and reduce a computational latency for garbage collection, a shared space-based FTL is adopted to maintain the information of SLOs for users and it manages shared spaces among the users. In our experiments, the proposal improved the throughput of garbage collection by 7.11%, on average, and reduced the computational latency for garbage collection by 9.63% on average, compared to the previous work.

▸Keyword: NAND Flash Memory, Virtualized Storage Systems, Solid State Drive, Garbage Collection, Performance Isolation, SLOs

I. Introduction

가상화 저장소(virtualized storage)는 사용자들이 추가적인 하드웨어 구축비용과 별도의 관리비용 없이 대량의 저장 공간을 요구조건에 따라 제공해주는 시스템이다[1]. 이러한 가상화 저장소 시스템은 시스템 내 할당 가능한 자원인 IO 지연시간, 처리량(throughput), 서비스 시간, 할당 공간과 같은 SLOs(Service Level Objectives) 기반 사용자들의 요구조건에 따라 저장 공간을 제공한다. 과거 수십 년 동안 가상화 저장소 시스템은 저렴한 비용으로 많은 데이터를 저장할 수 있는 자기 디스크(magnetic disk)를 주로 사용하였으며, 이러한 특성을 반영하여 기존 가상화 저장소 시스템은 자기 디스크의 고유 특성인 디스크 헤더 기반 순차 접근 속도 향상을 초점으로 많은 연구가 진행되었다[2-4]

최근에는 낸드 플래시를 사용한 가상화 저장소 시스템에 대한 연구가 활발히 진행 중이다[5-9]. 낸드 플래시 메모리는 기존의 자기 디스크와 달리 낮은 전력, 빠른 동작 속도, 높은 집적도와 같은 장점들을 가지고 있어 가상화 저장소 시스템뿐만 아니라 임베디드 장치, 소형 저장 매체, SSD(Solid State Drive)와 같은 다양한 분야에서 사용되고 있다. 그러나 낸드 플래시 메모리는 기존의 자기 디스크와 다른 고유의 특징들을 가지고 있다[10-11]. 이는 기존의 자기 디스크 기반 기법들이 낸드 플래시 메모리의 고유 특성인 제자리 갱신 불가, 느린 소거 연산 속도, 유사한 랜덤/순차 접근 시간 등의 고려를 하지 않기 때문에, 낸드 플래시 메모리 기반 가상화 저장소 시스템에 그대로 적용하기 어렵다[3-4]. 기존의 연구에서는 낸드 플래시 메모리의 고유 특성을 플래시 사상 계층(FTL: Flash Translation Layer)을 이용하여 완화하였으나, 낸드 플래시 메모리 기반 가상화 저장소 시스템은 기존의 플래시 사상 계층만으로 모든 근본적인 문제점들을 해결하기 어렵다[10-11].

이러한 문제점들을 해결하기 위해, 기존의 연구자들은 OPS 고립(Over-Provisioning Space Isolation), VSSD 프레임워크

• First Author: Sungho Kim, Corresponding Author: Jong Wook Kwak
*Sungho Kim (shk@gitc.or.kr), Software Research Team, Gyeongbuk Institute of IT Convergence Industry Technology (GITC)
**Jong Wook Kwak (kwak@yu.ac.kr), Dept. of Computer Engineering, Yeungnam University
• Received: 2019. 07. 01, Revised: 2019. 08. 14, Accepted: 2019. 08. 14.
• This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1A09000654)

(framework)와 같은 낸드 플래시 메모리 기반의 가상화 저장소 시스템 기법들을 제안했다[8-9]. OPS 고립과 VSSD 프레임워크는 가비지 컬렉션에 의해 의도치 않게 해당 사용자의 데이터가 아닌 다른 사용자의 데이터를 이주하게 되는 문제점을 해결하는 공통점이 있으며, 특히 VSSD 프레임워크는 낸드 플래시 메모리의 고유 특징들을 반영한 새로운 스케줄링 기법을 채택하여 OPS 고립보다 향상된 SLOs를 제공했다.

그러나 기존의 기법들은 동시에 다양한 어플리케이션이 동작하면서 발생할 수 있는 작업량의 특성을 유연하게 관리하고, 사용 패턴의 변화에 대한 추적이 부족하다. 예를 들어 현재 가상화 저장소 시스템은 쓰기 연산을 위한 사용 가능 블록들이 충분한 상황이고, 특정한 사용자가 많은 쓰기 연산을 요청하였다고 가정하자. 이 때 OPS 고립과 VSSD 프레임워크에서는 가상화 저장소 시스템 내 쓰기 연산이 가능한 블록들이 많음에도 불구하고, 사용자 SLOs로 인해 이런 유휴 공간을 활용하지 못하는 문제점이 존재한다. 이는 낸드 플래시 메모리 기반의 가상화 저장소 시스템 전체적인 측면에서 처리량, IO 지연시간, 서비스 시간과 같은 성능 저하의 원인이 된다.

본 논문에서는 OPS 고립과 VSSD 프레임워크에서 발생할 수 있는 어플리케이션 작업량 변화로 인해 시스템 전체의 성능이 저하되는 문제점을 해결하기 위한 새로운 기법을 제안한다. 제안하는 기법은 가상화 저장소 시스템의 성능 향상과 사용자 SLOs를 보장하기 위해, 가상화 저장소 시스템 내 일부 블록들을 공유하여 어플리케이션 작업량 변화에 따라 유연하게 대처한다. 이를 통해 가상화 저장소 시스템은 어플리케이션 작업량에서 많은 변화가 발생하면 가상화 저장소 시스템의 상대적 성능 향상에 유리한 환경을 제공하고, 만약 그렇지 못한 경우라 하더라도 기존의 기법들과 동일하게 사용자들의 SLOs를 보장한다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 낸드 플래시 메모리의 배경 지식을 서술하며 낸드 플래시 메모리 기반의 가상화 저장소 시스템에 대한 관련 연구들을 소개한다. 3장에서는 기존의 기법이 가지는 문제점들을 언급하고, 본 논문의 동기를 서술한다. 제안하는 기법은 4장에서 상세하게 기술한다. 5장에서는 기존의 기법들과 비교하여 제안하는 기법의 성능 평가를 진행하며 마지막 6장에서는 본 논문의 결론을 맺는다.

II. Background and Related Works

1. Background

플래시 메모리는 플래시 셀의 배열에 따라 노어(NOR)와 낸드(NAND)로 구분한다. 노어 플래시 메모리는 플래시 셀을 병렬로 구성하는 방식으로, 낮은 지연시간의 특성으로 BIOS와 XIP(Execute In Place)와 같은 빠른 동작 속도를 요구하는 장치들에 주로 사용한다. 반면 낸드 플래시 메모리는 직렬 단위로 셀들을 구성하여 플래시 셀의 높은 집적도를 가져올 수 있어, 플래시 메모리를 대용량화하기에 유리하다. 따라서 낸드 플래시 메모리는 기존의 자기 디스크를 대체하는 용도로 주로 활용된다. 이하 본 논문에서는 대용량 플래시 메모리인 낸드 플래시 메모리를 기준으로 서술하며, 낸드 플래시 메모리의 주요 특성은 아래 표 1과 같다[12-14].

Table 1. Characteristics of flash cells

Description	SLC	MLC	TLC
read latency	25us	60us	100us
program latency	200us	800us	2.4ms
erase latency	700us	1.5ms	3.0ms
bit per flash cell	1bit	2bit	3bit
endurance	10^5	10^4	10^3
price per bit	high	low	very low

이러한 장점들과 달리, 낸드 플래시 메모리는 기존의 자기 디스크와 다른 고유의 특징들을 가지고 있으며, 이러한 고유의 특징들은 다음과 같다.

특징 1. 이미 존재하는 데이터에 갱신이 발생하였을 때, 낸드 플래시 메모리는 제자리 갱신이 불가능하다. 이는 외부 자리 갱신(out-place update)을 요구한다.

특징 2. 낸드 플래시 메모리의 연산은 자기 디스크와 다른 소거 연산이 있다. 이는 이미 기록된 공간을 초기화하여 새로운 공간을 확보하기 위한 연산이다.

특징 3. 낸드 플래시 메모리의 연산 단위는 페이지와 블록으로 구분한다. 페이지는 연산의 최소 단위로서, 쓰기와 읽기 연산에 해당한다. 블록은 페이지의 묶음이며, 소거 연산의 단위이다.

특징 4. 낸드 플래시 메모리의 수명은 셀 단위로 측정하고, 셀의 수명은 소거 연산에 종속적이다. 또한 낸드 플래시 메모리는 자기 디스크와 달리 상대적으로 수명이 짧다.

이러한 문제점을 해결하기 위해서, 낸드 플래시 메모리는 플래시 사상 계층(FTL: Flash Translation Layer)을 주로 사용하고 있다[10-11]. 플래시 사상 계층은 낸드 플래시 메모리 위에 새로운 소프트웨어 계층을 두어, 기존의 자기 디스크와 동일한 수행을 가능하게 한다. 플래시 사상 계층은 낸드 플래시 메모리의 고유한 특징들을 해결하기 위해 플래시 사상 테이블, 가비지 컬렉션, 마모도 평준화와 같은 주요 요소들로 구성한다.

플래시 사상 테이블은 특징 1을 해결하기 위한 기법이다[15]. 이것은 낸드 플래시 메모리의 제자리 갱신을 수행하기 위해 플래시 사상 테이블의 논리 주소와 낸드 플래시 메모리의 물리 주소를 상호 묶음으로 관리한다. 논리 주소는 호스트에서 연산을 명령한 주소로서, 낸드 플래시 메모리에서는 가상의 주소를 의미한다. 물리 주소는 논리 주소에 사상한 낸드 플래시 메모리의 주소로서, 실제 데이터 주소를 의미한다. 이러한 일련의 동작들을 외부 자리 갱신이라 하며, 이를 통해 낸드 플래시 메모리의 갱신을 가능하게 한다. 이러한 플래시 사상 테이블은

사상 테이블의 구성에 따라 페이지, 블록, 하이브리드로의 구성이 가능하다.

가비지 컬렉션은 특징 2와 특징 3을 해결하기 위한 기법이다[16]. 가비지 컬렉션은 플래시 사상 테이블과 밀접한 연관성이 있다. 플래시 사상 테이블은 항상 최신의 데이터를 가지고 있고, 갱신 이전의 데이터는 관리하지 않는다. 이 경우, 플래시 사상 테이블은 데이터 갱신이 발생한 페이지를 유효 페이지(valid page), 갱신 이전의 페이지를 무효 페이지(invalid page)로 기록한다. 따라서 가비지 컬렉션은 갱신에 의해서 관리가 불필요한 무효 페이지를 초기화하여 새로운 공간을 확보하는 작업을 수행한다. 이 작업은 낸드 플래시 메모리의 수명을 감소시킬 수 있기 때문에, 기존의 연구자들은 이 과정에서 낸드 플래시 메모리의 수명도 함께 고려하였다.

특징 4는 낸드 플래시 메모리의 수명과 직접적인 연관성 있으며, 이를 해결하는 기법이 마모도 평준화이다[16]. 마모도 평준화는 거의 사용하지 않거나 드물게 사용 하는 블록들의 공간을 소거 연산을 활용하여 낸드 플래시 메모리 전체 블록의 수명을 고르게 줄일 수 있도록 유도한다. 하지만 이 작업은 가비지 컬렉션과 또 다른 수행이기 때문에, 추가적인 읽기/쓰기/소거 연산에 대한 지연시간이 발생하며, 이로 인해 마모도 평준화에서는 낸드 플래시 메모리 수명과 연산 수행 오버헤드를 모두 고려해야 한다.

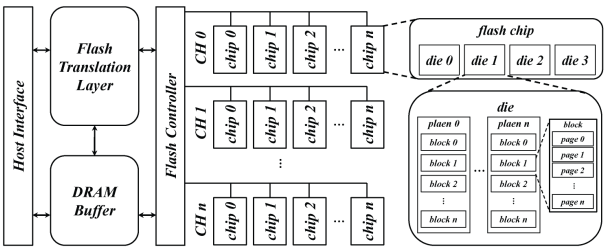


Fig. 1. Overall block diagram of Solid State Drives

이와 더불어 SSD는 다수의 낸드 플래시 메모리를 관리하기에 이외에도 추가적인 많은 기법들을 적용하고 있다. 그림 1은 SSD의 주요 구조를 보여주고 있다. 그림 1에서 SSD의 성능 향상에 가장 중요한 요소들은 병렬화(parallelism)와 DRAM 버퍼이다.

병렬화는 SSD 내에 하나 이상의 연산을 동시에 수행할 수 있도록 하며, 병렬화의 단위에 따라 채널(channel), 칩(chip), 다이(die), 플래인(plane)으로 구분 및 수행이 가능하다. SSD 병렬화 수행의 예시로는 SSD가 채널, 칩, 다이, 플래인을 각각 2개씩 가지고 있다고 가정할 경우, 한 번에 16개의 읽기/쓰기/소거 연산을 동시에 수행하는 것이 가능하다. 이처럼 SSD 병렬화는 동시에 1개 이상의 연산을 수행하여 SSD 내 성능을 크게 향상시킬 수 있다. DRAM 버퍼는 SSD와 운영체제 간 지연시간의 차이를 줄이기 위해 자주 사용하는 페이지들, 블록 단위 페이지들에 대해 읽기/쓰기 연산 지연시간 감소나 가비지 컬렉션 지연시간 감소와 같은 목적으로 활용되어 채택된다.

2. Related Works

기존의 연구자들은 가상화 저장소 시스템의 연구 분야 중 스케줄러/플래시 사상 계층에 관한 연구를 활발히 진행하였다. CFFQ(Completely Fair FIFO Queueing)는 사용자들의 공평성(fairness)과 고성능(high performance)을 보장하기 위한 스케줄러로 제안되었다[5]. CFFQ는 낸드 플래시 메모리에서 가장 좋은 성능을 보이는 스케줄러인 FIFO와 사용자들의 공평성을 보장하는 CFQ(Completely Fair Queueing)와의 결합을 통해 우수한 성능을 향상을 보였다. 그러나 이 기법은 리눅스 시스템에 기본적으로 내장된 스케줄러들만으로 설계하였기 때문에 기존 CFFQ 혹은 CFQ와 거의 유사한 성능 결과를 보여 실제 효율성이 떨어지는 단점이 있다.

FIOS는 읽기와 쓰기 연산이 비대칭적으로 수행하는 환경에서 사용자들의 공평성 문제를 해결하기 위해 제안되었다[6]. FIOS는 낸드 플래시 메모리에서 긴 지연시간을 요구하는 쓰기 연산보다는 읽기 연산을 중심으로 스케줄링하며, 이를 통해 FIOS는 낸드 플래시 메모리에 접근하는 전제 연산 지연시간을 감소시켰다. 그러나 이 기법은 읽기 연산을 중점으로 수행하는 작업에서 유리한 측면이 있지만, 사용자들의 공평성에는 치명적인 영향을 초래할 수 있다. 또한 FIOS는 이전의 접근 패턴을 기반으로 수행하기 때문에, 새로운 패턴 변화에 민감하지 못한 문제점이 있다.

FlashFQ는 FIOS에 기반을 둔 기법으로 접근 패턴에 민감한 문제를 보완하기 위해 제안되었다[7]. 이 기법은 기존의 FIOS와 마찬가지로 읽기 연산을 기본적으로 고려하되, IO들을 병렬화에 최적화되도록 설계하였다. IO 연산의 병렬화는 낸드 플래시 메모리 측면에서 동시에 읽기/쓰기/소거 연산을 수행할 수 있기 때문에 연산의 효율성이 높다. 그러나 FlashFQ 또한 읽기 연산의 측면에서 효율성을 향상시킬 수 있지만, 사용자들의 공평성을 보장하기 힘든 문제점이 있다.

OPS 고립은 가비지 컬렉션 정책을 수행하면서 사용자들 간의 성능 고립(performance isolation)이 침해되는 문제점을 해결하기 위해 제안되었다[8]. 성능 고립 침해 문제는 가상화 저장소를 사용하는 특정한 사용자에 가비지 컬렉션이 발생하였을 경우, 자신이 가지는 데이터뿐만 아니라 다른 사용자들의 데이터들도 함께 이주하는 현상을 의미한다. 이는 의도치 않게 특정한 사용자가 부당하게 다른 사용자의 데이터 이주와 같은 오버헤드를 전가시키는 것으로 해당 사용자의 SLOs를 침해하는 것을 의미한다. OPS 고립은 이러한 문제점을 해결하기 위해 사용자 단위의 전용 블록을 할당했고, 이를 통해 성능 고립이 침해되는 문제점을 해결하였다. 그러나 이 기법은 가상화 저장소 시스템에서 동시에 다수의 사용자들이 읽기/쓰기/소건 연산을 수행하였을 때 발생할 수 있는 사용자들 간 공평한 연산 수행, 즉 공평성에 대한 고려가 부족했다.

VSSD 프레임워크는 OPS 고립의 장점을 그대로 가져가되 공평성 문제를 해결한 기법이다[9]. 이 기법은 사용자들의 SLOs를 보장하기 위해 OPS 고립과 유사한 사용자 단위의 블

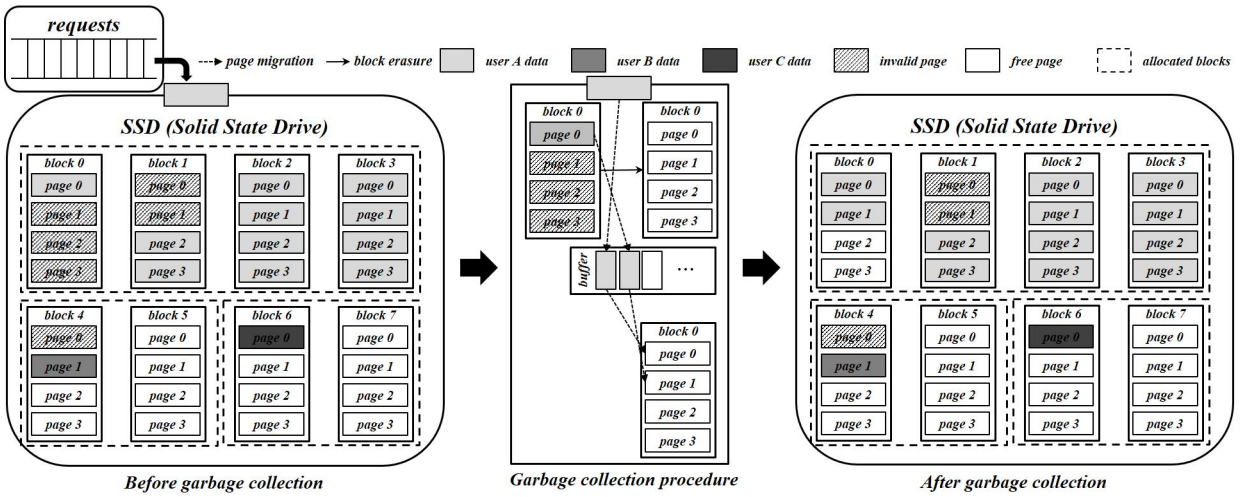


Fig. 2. A weakness of VSSD framework

록 할당을 수행한다. 또한 VSSD 프레임워크는 사용자들 간 공평성을 해결하기 위해 채널(channel) 큐를 이용한 신용도(credit) 기반 스케줄러를 사용한다. 신용도 기반 스케줄러는 사용자들이 수행하는 순서에 따라 신용도를 부여하고, 신용도가 없을 때까지 작업을 수행한다. 이를 통해 이 기법은 다수의 사용자들에 대한 공평성 문제를 해결하였다. 하지만 VSSD 프레임워크는 동시에 다수의 사용자가 다양한 어플리케이션들을 동작시키면서 발생할 수 있는 작업량 변화에 대한 대처가 유연하지 못한 측면이 있다. 예를 들어 가상화 저장소 시스템의 특정 사용자가 특정 시점에 많은 쓰기 연산을 요구하는 경우, VSSD 프레임워크는 해당 사용자에게 할당된 공간만을 위주로 사용하기 때문에 내부적으로 많은 가비지 컬렉션 연산을 발생시킨다. 이는 가상화 저장소 시스템 측면에서 쓰기 연산을 수행할 수 있는 공간이 많은 경우, 저장소 내 전체 블록에 대한 공간 활용도가 감소하게 된다. 이 결과로 VSSD 프레임워크에서는 이와 같은 작업량 변화가 발생할 경우, 가상화 저장소 시스템의 일부 공간만을 집중적으로 사용하여 전반적인 성능 감소가 발생할 수 있는 주요 요인이 될 수 있다.

그 이외에도 최근에는 NVMe(Non-Volatile Memory Express) SSDs에 SLOs를 보장하기 위한 다양한 연구도 진행중에 있다[17].

III. Motivation

이 절에서는 VSSD 프레임워크가 가질 수 있는 문제점에 대해서 자세히 분석하고, 이러한 문제점에 대한 해결 방향을 제시한다.

VSSD 프레임워크에서는 OPS 고립에서 발생할 수 있는 사용자 간 SLOs 침해와 채널 큐에 대한 문제점을 내부 가비지 컬렉션 정책을 통해 해결하였나, 이로 인해 가상화 저장소 시스템 전체의 성능 저하를 가져 올 수 있다. 그림 2에서는 VSSD 프레임워크가 내부 가비지 컬렉션으로 인해 발생할 수 있는 문제점을 보여주고 있다. 그림 2에서 사용자 A는 새로운 데이터로 인해 쓰기 연산이 발생했고, 사용자 A에 대한 자유 블록(free block)들은 존재하지 않는 상황이다. 이 때 사용자 A는 새로운 자유 블록을 확보하기 위해 하나의 블록(블록 0)을 선정한 후 새로운 공간을 확보하게 된다. 그러나 가상화 저장소 시스템 전체 성능 측면에서는 이와 같은 수행이 결코 옳은 선택이 아닐 수 있다. 예를 들어 그림 2와 같이 사용자 B와 사용자 C는 한 개의 자유 블록을 보유하고 있고, 나머지 블록들은 하나의 페이지만 사용하고 있다. 이 경우 가상화 저장소 시스템에서는 사용자 B와 사용자 C의 자유 블록을 사용자 A에게 할당하여 이를 유연하게 활용함으로써 유리한 측면을 가져올 수 있으며, 아래는 이러한 측면 분석의 요약이다.

1. 컴퓨팅 시스템은 인접하는 공간을 참조하는 공간 지역성과 특정한 영역을 주어진 특정 시간에 집중적으로 참조하는 시간 지역성의 특성을 가지고 있다. 가상화 저장소 시스템도 예외는 아니다.

2. 낸드 플래시 메모리는 오랜 기간 동안 공간을 사용할 경우 무효 페이지의 개수가 점차 증가하게 된다. 이는 가비지 컬렉션 수행 기간을 최대한 연장할 경우 많은 무효 페이지들을 통해 가비지 컬렉션의 상대적 작업 효율성을 향상할 수 있음을 의미한다.

위에서 제시한 분석은 컴퓨팅 시스템의 기본적인 특성이며, 특히 낸드 플래시 메모리에서 두드러지는 특성이라고 할 수 있다. 또한 이러한 특성들은 어플리케이션의 작업량에 큰 변화가 발생하였을 경우에서 보다 더 큰 의미를 가진다. 즉 가상화 저장소 시스템에서는 사용자 SLOs를 침해하지 않는 범위 내에서 컴퓨팅 시스템의 블록들을 상호 공유할 수 있으며, 이렇게 공유된 블록들을 효과적으로 활용한다면 궁극적으로 전체 시스템의 성능을 보다 더 향상시킬 수 있게 된다. 이러한 동기를 기반으로 본 논문에서는 VSSD 프레임워크의 문제점을 해결하기 위한 방안을 제시한다.

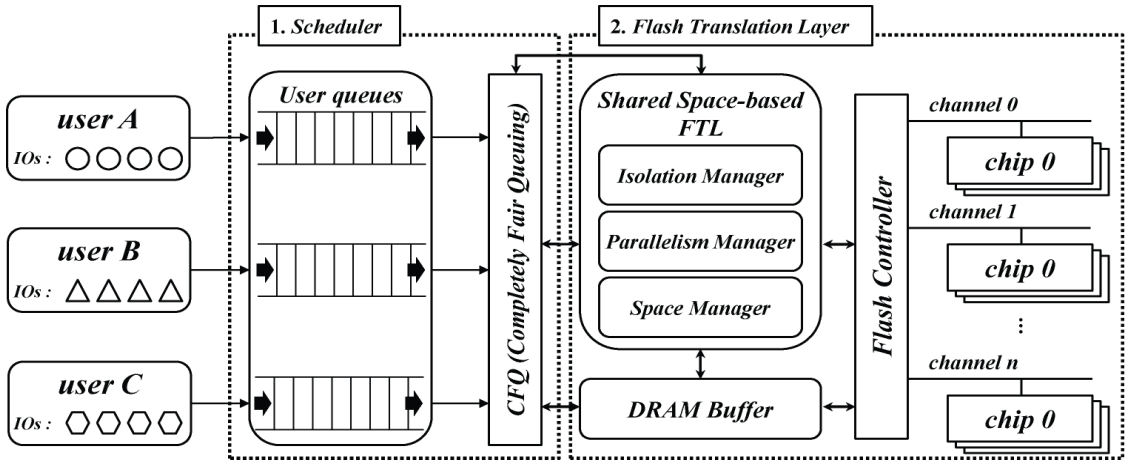


Fig. 3. The proposed scheme

IV. The Proposed Scheme

본 절에서는 논문에서 제안하는 기법을 서술한다. 제안하는 기법의 구조는 그림 3과 같다. 제안 기법은 VSSD의 문제점을 해결하기 위해, 다음 두 가지 기법들을 사용한다.

1. 사용자들의 IO를 구분하고 자원 사용의 공평한 기회를 부여하기 위해, 사용자 큐와 CFQ(Completely Fair Queuing) 기반 스케줄러를 사용한다.
2. 사용자들 간 SLOs를 보장하기 위해 플래시 사상 계층에 추가적인 기능인 고립 관리자, 공간 관리자, 병렬화 관리자를 추가하여 사용한다.

제안 스케줄러는 호스트로부터 입력받은 사용자들의 IO를 스케줄링하기에 앞서, 개별 사용자 큐에 이를 미리 보관한다. 이는 두 가지 목적을 달성하기 위해 사용한다. 첫 번째, 개별 사용자 큐는 개별 사용자의 SLOs를 보장하기 위해 사용하고, 이를 통해 개별 사용자의 공평한 IO 처리 시간을 보장한다. 두 번째, 개별 사용자 큐의 사용은 공평하게 정해진 IO 처리 시간을 보장하여, 사용자 연산 처리 간 SSD 연산 수행 병렬화를 가속화시킨다. 이를 통해 가상화 저장소 시스템과 개별 사용자 측면에서는 연산 수행의 효율성과 체감 지연시간 감소의 효과를 동시에 가져올 수 있다.

제안 스케줄러에서는 SLOs 기반 개별 사용자 큐에 IO 처리 시간을 부여하고, 사용자들의 IO 처리 시간 보장은 CFQ 스케줄러를 기반으로 아래의 순서로 수행한다.

1. 해당 사용자에게 SLOs 기반 IO 처리 시간을 부여한다.
2. IO 처리 시간 동안에 개별 사용자 큐에 해당하는 IO를 처리한다.
3. IO 처리 시간이 만료되면 다음 사용자에게 IO 수행 권한을 이양한다. 이때 제안 스케줄러는 라운드 로빈(round robin) 방식으로 다음 사용자를 선정하게 된다.

제안 스케줄러에서는 기존 CFQ 스케줄러와 수행 방식은 동일하나 개별 사용자의 SLOs 기반으로 동작하는 것이 큰 차이점에 해당하고, 이를 통해 제안 기법은 개별 사용자 간 공평성을 유지하게 된다.

제안 스케줄러 과정을 통해 처리 요청된 IO들은 실질적인 연산을 수행하기 위해 플래시 사상 계층으로 전달되며, 제안 기법은 사용자들 간 SLOs를 보장하기 위해 플래시 사상 계층에 추가적인 공간 관리자, 병렬화 관리자, 그리고 고립 관리자를 추가하여 사용한다.

공간 관리자는 개별 사용자들의 SLOs를 관리한다. 이는 스케줄러의 서비스 시간, 가상화 저장소 시스템의 할당 공간, 공유 공간 정보에 해당하며, 스케줄러 관련 정보뿐만 아니라 공유 공간 기반의 플래시 사상 계층의 전반적인 부분에 대한 정보를 관리한다. 병렬화 관리자는 가상화 저장소 시스템 내 하나 이상의 읽기/쓰기/소거 연산을 동시에 수행 할 수 있는 단위를 결정하며, 제안 기법은 최대 병렬화 단위인 채널, 칩, 다이, 플래인을 모두 동시에 수행하는 방식을 사용한다. 고립 관리자는 어플리케이션 작업량의 변화에 따라 가비지 컬렉션 수행의 유연성을 보장하는 역할을 수행하며, 이를 위해 가상화 저장소 시스템 내의 공유 공간을 활용하며 이에 대한 관리를 수행한다. 공유 공간은 모든 사용자들이 가상화 저장소 시스템으로부터 할당받은 공간 중 일부를 공유 공간으로 활용하며, 수식 (1)을 통해 실제 할당할 공유 공간의 크기를 결정한다.

$$s = \sum_{0}^{n-1} u_n \times \beta \qquad (1)$$
$$0 < \beta < 1$$

수식 (1)에서 n은 가상화 저장소 시스템을 이용 중인 모든 사용자들, u는 각 사용자가 할당 받은 공간, β는 가상화 저장소 시스템에서 설정한 공유 공간의 비율, s는 전체 사용자에 대한 공유 공간을 의미한다. 고립 관리자는 사용자들에게 할당된 영역을 기준으로 공유 공간을 확보한다.

고립 관리자가 공유 공간을 확보하는 예시로 가상화 저장소 시스템으로부터 세 명의 개별 사용자가 각각 40GB, 20GB, 20GB의

할당 공간을 부여받았고, 전체 용량은 100GB라고 가정하자. 이 때 β가 10%인 경우, 고립 관리자에서는 개별 사용자가 4GB, 2GB, 2GB인 총 8GB의 크기를 공유 공간으로 확보하게 된다. 이와 같이 고립 관리자가 공유 공간을 확보하는 이유는 향후 20GB를 할당받을 예정인 개별 사용자들에게 기존 사용자들의 SLOs 피해를 전가시키지 않기 위해서이다. 만약 이러한 나머지 공간(20GB)을 다른 사용자들의 성능 향상 목적으로 할당하게 되면, 새롭게 할당받을 사용자들이 기존의 사용자들이 성능 향상 목적으로 사용한 공간에 대해 읽기/쓰기/소거 연산에 대한 피해가 전가 될 수밖에 없다. 따라서 고립 관리자는 이러한 문제점을 방지하기 위해서 사용자들이 할당한 영역에 국한해서 공유 공간을 할당한다.

V. Performance Evaluation

1. Experimental Setup

본 논문에서 제안하는 기법의 성능 평가를 진행하기 위해, DiskSim 4.0 시뮬레이터를 사용하였다[18]. DiskSim 4.0은 저장 공간의 다양한 요소들이 잘 반영되어 있고 확장성이 뛰어나기 때문에 보조 기억 장치의 성능 평가 용도로 많은 연구에서 활용되고 있다. 이러한 특성 때문에, 마이크로소프트社는 DiskSim 4.0을 SSD의 특징들을 가지는 모델로 추가 확장하여 배포하였다[19]. 표 2는 실험에서 사용한 SSD의 기본적인 매개변수를 보여주고 있다.

Table 2. Parameters of simulation environment

Parameter	Description
Page read latency	60us
Page program latency	800us
Block erase latency	1.5ms
Over-provision blocks	15%
Garbage Collection Trigger	5%
Number of channels	10
Chips per channel	1
Dies per chip	2
Planes per die	2
Blocks per plane	variable
Pages per block	128
Page size	4KB

성능 평가의 신뢰성을 보장하기 위해, 본 논문에서는 UMass Trace Repository와 IOTTA Repository 워크로드를 사용하였다[20-21]. UMass Trace Repository는 OLTP에서 동작하는 어플리케이션의 특성을 가지고 있고, IOTTA Repository는 서버 측면에서 단말 사용자, 웹 서비스 등의 특정을 가지는 워크로드이다. 표 3은 실험 환경에서 사용한 워크로드의 특성을 보여주고 있다.

Table 3. Workload characteristics

Workload	Write ratio	Avg, request size (pages)	workload size (GB)
financial1	76	1.44	5
financial2	17	1.09	5
webmail	73	4.35	40
web-vm	72	4.19	40
webusers	70	3.51	10
online	64	3.80	10

한편, 시스템에서 각 사용자별로 제공되는 SLOs를 평가하기 위해 표 4와 같은 환경을 가정하여 실험을 진행하였다. 가상화 저장소 시스템에서 할당받은 SLOs에 대한 상세는 표 4와 같다.

Table 4. SLOs assigned by virtualized storage system

Workloads		Allocated blocks		SSD capacity for users (GB)
User 1	User 2	User 1	User 2	
financial1	financial2	256	256	10
webmail	web-vm	2048	2048	80
webusers	online	512	512	20

2. Experimental Results

본 논문에서는 제안하는 기법을 기존의 OPS 고립을 향상시킨 VSSD 프레임워크 기법과 비교한다. 성능 평가에 대한 비교 지표는 SSD에서 가장 많은 오버헤드를 초래하는 가비지 컬렉션의 평균 처리량, 평균 지연시간을 기준으로 한다.

그림 4는 가비지 컬렉션의 평균 처리량을 VSSD 프레임워크를 기준으로 정규화하여 보여주고 있다. 제안하는 기법은 기존의 VSSD 프레임워크와 비교하여 평균적으로 7.11%의 가비지

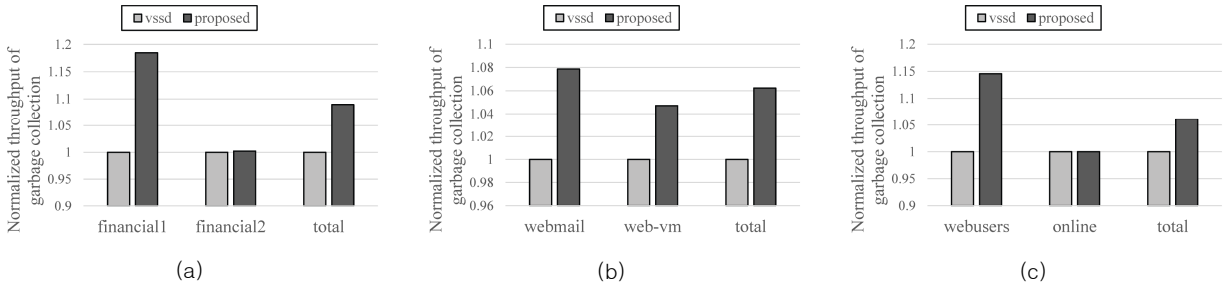


Fig. 4. The normalized throughput of garbage collection

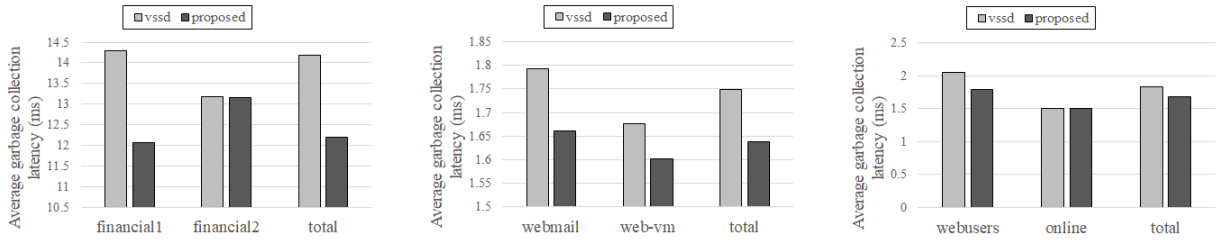


Fig. 5. Average garbage collection latency

컬렉션의 처리량 향상을 보였다. 가상화 저장소 시스템에서 가비지 컬렉션은 많은 오버헤드를 차지하기 때문에, 결과적으로 사용자들은 보다 더 향상된 성능을 기대할 수 있다. 특히, 제안하는 기법은 공유 영역을 사용함에도 다른 사용자들의 SLOs를 침해하지 않았다는 것에 주목할 필요가 있다. 오히려 제안하는 기법은 사용자들에게 보다 나은 SLOs를 제공한다.

그림 5는 가비지 컬렉션의 평균 지연시간을 VSSD 프레임워크를 기준으로 정규화하여 보여주고 있다. 이는 사용자가 체감하는 실제 SLOs의 결과라 할 수 있다. 제안하는 기법은 VSSD 프레임워크와 비교하였을 때 평균적으로 9.63%의 가비지 컬렉션의 평균 지연시간을 감소시켰다. 특히, 그림 5(a)는 VSSD 프레임워크와 비교하여 측정된 실제 시간으로 최대 1.98ms까지 지연시간을 감소시켰다. 이는 한 번의 가비지 컬렉션 수행에서 최소 하나의 소거 연산을 줄였음을 의미한다. 결과적으로 보았을 때, 제안하는 기법은 가상화 저장소 시스템의 전체 성능을 향상시키고, 이를 통해 사용자들에게 보다 나은 SLOs를 제공할 수 있다.

VI. Conclusions

본 논문에서는 기존 VSSD 프레임워크에서 발생하는 내부 가비지 컬렉션 문제를 해결하기 위한 기법을 제안하였다. 제안 기법은 내부 가비지 컬렉션 문제와 사용자 SLOs를 보장하기 위해 공유 블록을 사용하였으며, 이는 지연 가비지 컬렉션 동작을 통해 내부 가비지 컬렉션의 효율성을 높였다. 실험 결과에서 제안하는 기법은 기존 VSSD 프레임워크와 비교하여 가비지 컬렉션 처리량을 7.11% 향상시켰고, 가비지 컬렉션 평균 지연시간을 9.63% 감소시켰다. 이를 통해, 제안 기법은 가상화 저장소 시스템의 전체 성능을 향상시켰으며, 이는 가상화 저장소 시스템을 활용하는 전체 사용자들에게 보다 향상된 SLOs를 제공하는 역할을 한다.

향후 연구에서는 제안 기법의 공유 공간을 사용할 시 발생할 수 있는 다양한 문제점에 대해서 분석을 진행할 예정이다. 또한 가상화 시스템에서 자주 사용하는 블록에 대한 추적 및 관리를 통해 사용자들 간의 공유 공간을 효율적으로 판단할 수 있는 기법에 대한 연구도 진행할 것이다.

REFERENCES

[1] C. A. Waldspurger, "Memory resource management in VMware ESX server," ACM SIGOPS Operating Systems Review, Vol. 36, No. SI, pp. 181, Dec. 2002.

[2] T. Garfinkel, and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," Ndss, Vol. 3, No. 2003, pp. 191-206, Feb. 2003.

[3] S. Iyer, and P. Druschel, "Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O," ACM SIGOPS Operating Systems Review, Vol. 35, No. 5, pp. 117-130, Oct. 2001.

[4] Y. Zhang, and B. Bhargava, "Self-Learning Disk Scheduling," IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 1, pp. 50-65, Jan. 2009.

[5] M. Yi, M. Lee, and Y. I. Eom, "Cffq: I/o scheduler for providing fairness and high performance in ssd devices," Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, pp. 87, Jan. 2017.

[6] S. Park, and K. Shen, "FIOS: a fair, efficient flash I/O scheduler," FAST, Vol. 12, pp. 13-13, Feb. 2012.

[7] K. Shen, and S. Park, "Flashfq: A fair queueing i/o scheduler for flash-based ssds," Presented as part of the 2013 USENIX Annual Technical Conference, pp. 67-78, 2013.

[8] J. Kim, D. Lee, and S. H. Noh, "Towards SLO Complying SSDs Through OPS Isolation," 13th USENIX Conference on File and Storage Technologies (FAST 15), pp. 183-189, 2015.

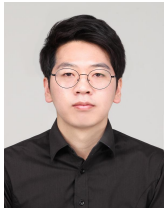
[9] D. W. Chang, H. H. Chen, and W. J. Su, "VSSD: performance isolation in a solid-state drive," ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 20, No. 4, pp. 51, 2015.

[10] T. S. Chung, D. J. Park, S. Park, D. H. Lee, S. W. Lee, and H. J. Song, "A survey of flash translation layer," Journal of Systems Architecture, Vol. 55, No. 5-6, pp. 332-343, 2009.

[11] S. Kim, and J. W. Kwak, "Analysis of Potential Risks for Garbage Collection and Wear Leveling Interference

in FTL-based NAND Flash Memory," Journal of The Korea Society of Computer and Information, Vol. 24, No. 3, pp. 1-9, Mar. 2019.
[12] "MT29F4G08ABADAWP 8Gbit SLC NAND Flash Memory Data Sheet," Micro Technology, 2009.
[13] "K9GAG08U0M 2G x 8bit MLC NAND Flash Memory Data Sheet," Samsung Electronics, https://www.samsung.com, 2007
[14] S. Hachiya, K. Johguchi, K. Miyaji, and K. Takeuchi, "TLC/MLC NAND flash mix-and-match design with exchangeable storage array," International Conference on Solid State Devices and Materials, pp. 894-895, Sep. 2013.
[15] D. Ma, J. Feng, and G. Li, "A survey of address translation technologies for flash memories," ACM Computing Surveys (CSUR), Vol. 46, No. 3, pp. 36, 2014.
[16] M. C. Yang, Y. M. Chang, C. W. Tsao, P. C. Huang, Y. H. Chang, and T. W. Kuo, "Garbage collection and wear leveling for flash memory: Past and future," 2014 IEEE International Conference on Smart Computing, pp. 66-73, Nov. 2014.
[17] S. M. Huang, and L. P. Chang, "Providing SLO compliance on NVMe SSDs through parallelism reservation," ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 23, No. 3, pp. 28, 2018
[18] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger, "The DiskSim simulation environment version 4.0 reference manual reference manual (CMUPDL-08-101)," Technical report Parallel Data Laboratory, 2008.
[19] V. Prabhakaran, and T. Wobber, "SD extension for DiskSim simulation environment," Microsoft Reseach, 2009.
[20] "I/O and search engine I/O. umass trace repository," Application. OLTP, 2007.
[21] "SNIA IOTTA Repository," Trace. Exchange, 2010.

Authors



Sungho Kim received a B.S. degree in the Department of Computer Engineering from Yeungnam University College, Daegu, Korea in 2012 and a Ph.D. degree in Department of Computer Engineering from Yeungnam University, Gyeongsan, Gyeongsangbuk, Korea in 2019. He is currently as a senior researcher in the software research team at Gyeongbuk Institute of IT Convergence Industry Technology (GITC). His current research interests include a big data, a deep learning, embedded systems and non-volatile memory systems.

Jong Wook Kwak received a B.S. degree in Computer Engineering from Kyungpook National University, Daegu, Korea in 1998, a M.S. degree in Computer Engineering from Seoul National University, Seoul, Korea in 2001, and a Ph.D. degree in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea in 2006. From 2006 to 2007, he worked as a senior engineer in the SoC R&D Center, at Samsung Electronics Co., Ltd. He is currently a professor in the Department of Computer Engineering, Yeungnam University. His research interests include advanced processor architecture, low-power mobile embedded system, and high performance parallel computing.