사물인터넷융복합논문지 Vol. 10, No. 3, pp. 19-24, 2024 https://doi.org/10.20465/KIOTS.2024.10.3.019

저장시스템의 응답 시간 개선을 위한 효율적인 이중 큐 전략

이현섭*
백석대학교 컴퓨터공학부 교수

An Efficient Dual Queue Strategy for Improving Storage System Response Times

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

요 약 최근 빅데이터, 클라우드 컴퓨팅, 인공지능 등 대용량 데이터 처리 기술의 발전에 따라 데이터센터와 엔터프라이즈 환경에서 고성능 저장장치에 대한 요구가 증가하고 있다. 특히 저장장치의 빠른 데이터 응답 속도는 전체 시스템 성능을 좌우하는 핵심 요소이다. 이에 NVMe(Non-Volatile Memory Express) 인터페이스 기반 SSD(Solid State Drive)가 주목받고 있으나, 다수 호스트의 대량 데이터 입출력 요청을 동시에 처리하는 과정에서 새로운 병목 현상이 발생하고 있다. SSD는 일반적으로 호스트 요청을 내부 큐에 순차적으로 쌓아 처리하는 방식을 취한다. 이때 긴 전송 길이 요청이 먼저 처리되면 짧은 요청들이 장기간 대기하여 평균 응답 시간이 증가한다. 이 문제를 해결하기 위해 데이터 전송 시간제한과 데이터 분할 전송 방법이 제안되었으나 근본적인 해결책이 되지 못했다. 본 논문에서는 저장장치 내부 데이터 처리 스케줄링 전략인 DQBS(Dual Queue Based Scheduling Scheme)를 제안한다. 이 방식은 이중 큐 기반의 스케줄링 전략으로 하나의 큐에서는 요청 순서를, 다른 큐에서는 전송 길이를 기준으로 데이터 전송 순서를 관리한다. 그리고 요청 시간과 전송 길이를 종합적으로 고려하여 효율적인 데이터 전송 순서를 결정한다. 이를 통해 대기 시간이 긴 요청과 짧은 요청을 균형있게 처리할 수 있어 전체 평균 응답 시간을 단축시킬 수 있다. 실제 시뮬레이션 결과, 제안 기법은 기존 순차 처리 방식 대비 월등히 향상된 성능을 보였다. 본 연구는 고성능 SSD 환경에서 데이터 전송 효율을 극대화하는 스케줄링 기법을 제시하여, 차세대 고성능 저장 시스템의 발전에 기여할 수 있을 것으로 기대된다.

주제어 : 낸드 플래시 메모리, 저장장치, 응답 시간, 스케줄링, 우선순위 큐

Abstract Recent advances in large-scale data processing technologies such as big data, cloud computing, and artificial intelligence have increased the demand for high-performance storage devices in data centers and enterprise environments. In particular, the fast data response speed of storage devices is a key factor that determines the overall system performance. Solid state drives (SSDs) based on the Non-Volatile Memory Express (NVMe) interface are gaining traction, but new bottlenecks are emerging in the process of handling large data input and output requests from multiple hosts simultaneously. SSDs typically process host requests by sequentially stacking them in an internal queue. When long transfer length requests are processed first, shorter requests wait longer, increasing the average response time. To solve this problem, data transfer timeout and data partitioning methods have been proposed, but they do not provide a fundamental solution. In this paper, we propose a dual queue based scheduling scheme (DQBS), which manages the data transfer order based on the request order in one queue and the transfer length in the other queue. Then, the request time and transmission length are comprehensively considered to determine the efficient data transmission order. This enables the balanced processing of long and short requests, thus reducing the overall average response time. The simulation results show that the proposed method outperforms the existing sequential processing method. This study presents a scheduling technique that maximizes data transfer efficiency in a high-performance SSD environment, which is expected to contribute to the development of next-generation high-performance storage systems

Key Words : Nand Flash Memory, Storage, Response Time, Scheduling, Priority Queue

*This paper was supported by 2024 Baekseok University Research Fund
*교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2024년 04월 06일 수정일 2024년 05월 29일 심사완료일 2024년 06월 10일

1. 서론

최근 빅데이터, 클라우드 컴퓨팅, 인공지능 등 대용량 데이터 처리를 요구하는 기술이 급속히 발전함에 따라[1], 데이터센터와 엔터프라이즈 환경에서 저장장치의 높은 데이터 처리 성능이 필수적으로 요구되고 있다[2]. 특히 저장장치의 빠른 데이터 응답 속도는 전체 시스템의 성능을 좌우하는 중요한 요소로 작용한다[3]. 이에 NVMe (Non-Volatile Memory Express) 인터페이스 기반의 SSD(Solid State Drive)가 주목받고 있다. NVMe SSD는 기존 SATA 방식 대비 뛰어난 대역폭과 응답 지연 시간을 제공하는 장점이 있다[4]. 그러나 NVMe SSD는 다수 호스트로부터 대량의 데이터 입출력 요청을 동시에 받게 되면서, 내부적인 처리 과정에서 새로운 병목 현상이 발생하게 되었다[5]. SSD는 일반적으로 호스트의 요청을 내부 큐에 차례로 쌓아 놓고, 데이터 전송이 준비되는 대로 하나씩 처리하는 방식을 취한다[6]. 이때 긴 전송 길이를 가진 요청이 먼저 처리되면, 그 뒤를 이어 전송 길이가 짧은 다수의 요청이 장기간 대기하게 되어 평균 응답 시간이 매우 증가하는 문제가 발생한다[7]. 이를 해결하기 위해 각 요청의 데이터 전송 시간에 일정 제한을 두고, 전송 중 일부 요청을 나누어 중간에 끼워 넣어 전송하는 방식이 제안되었다[8]. 그러나 이 방법 역시 근본적인 해결책이 되지 못했다. 응답 시간 개선 효과가 요청 특성에 따라 제한적이었기 때문이다[9].

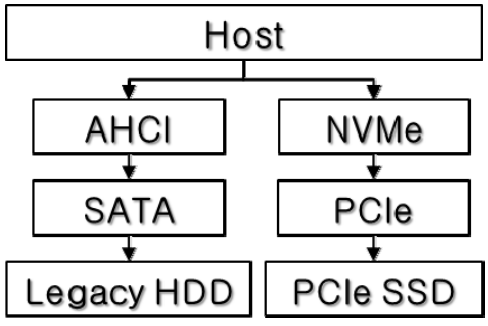
이러한 응답 지연 문제를 해결하기 위해 본 논문에서는 NVMe SSD 내부의 데이터 처리 스케줄링 전략을 개선하여, 호스트 입출력 요청에 대한 전송 응답 시간을 향상하는 DQBS(Dual Queue Based Scheduling Scheme)를 제안한다. 이 방법은 이중 큐 기반의 스케줄링 기법을 적용하였다, 하나의 큐에서는 요청 순서를 기준으로 전송 요청을 관리하며, 다른 하나의 큐에서는 전송 길이를 기준으로 관리한다. 그리고 요청 시간과 전송 길이를 종합적으로 고려하여 효율적인 데이터 전송 순서를 결정한다. 이 방법은 대기 시간이 긴 요청과 짧은 요청을 균형있게 처리함으로써 전체 평균 응답 시간을 단축시킬 수 있다. 실제 시뮬레이션 결과, 제안 기법이 기존의 단순 순차 처리 방식 대비 월등히 향상된 성능을 보이는 것으로 확인되었다.

논문의 구성은 다음과 같다. 2장에서는 SSD의 인터페이스 구조에 대한 배경과 대량의 데이터 처리중 발생할 수 있는 병목현상 문제를 설명한다. 3장에서는 논문에서 제안하는 이중 큐 기반의 스케줄링 기법을 설명하고 예제를 통해 효과를 분석한다. 4장에서는 시뮬레이션을 통해 제안하는 DQBS 기법을 적용했을 때 향상되는 응답 시간을 측정한다.

2. 배경

2.1 SSD의 인터페이스

SSD는 작동 인터페이스에 따라 다양한 성능 특성이 있다. 기존에 많이 사용되던 SATA(Serial Advanced Technology Attachment) 인터페이스는 HDD를 위해 설계된 것으로, SSD의 고성능을 충분히 발휘하기에는 한계가 있었다. 이에 비해 NVMe(Non-Volatile Memory Express) 인터페이스는 SSD 특성에 최적화되어 설계되었다.

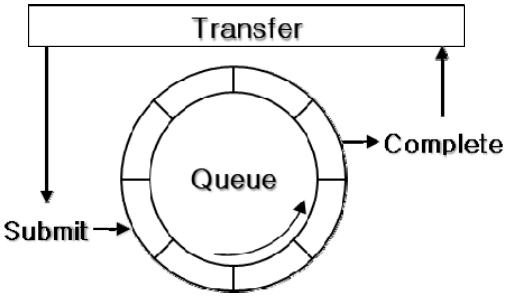


[Fig. 1] SATA and SATA Interface

Fig. 1은 저장장치의 SATA와 NVMe 인터페이스를 통한 기본 데이터 전송 작동 원리를 보여준다. SATA 방식에서 호스트는 AHCI(Advanced Host Controller Interface)를 통해 SSD와 통신한다. AHCI는 HDD 환경을 위해 만들어져 SSD의 빠른 속도를 충분히 활용하지 못한다는 한계가 있다. 반면 NVMe는 PCIe(Peripheral Component Interconnect Express) 버스를 사용하여 호스트와 SSD가 직접 통신하는 구조를 가진다. PCIe는 고속 직렬 인터페이스로 SATA 대비 매우 높은 대역폭과 낮은 지연시간을 제공한다. NVMe는 SSD 내부 구조에 대한 병렬 액세스와 큐잉 메커니즘을 지원해 데이터 처리 효율을 극대화한다[10]. 이렇듯 NVMe는 SSD 특성에 최적화되어 설계되어 높은 IOPS, 낮은 지연시간, 우수한 데이터 전송 성능 등의 이점을 가진다[11].

2.2 전송 스케줄링 최적화의 필요성

SSD는 호스트에게서 들어오는 데이터 입출력 요청을 내부 큐에 일렬로 쌓아두고, 순차적으로 처리하는 방식을 취한다. 호스트 요청은 요청된 순서대로 큐에 차례로 들어가고, SSD 내부에서 준비가 완료된 요청은 전송 큐로 이동하여 최종적으로 호스트에 응답이 전달된다. 과거 SATA 인터페이스 환경에서는 SSD의 데이터 처리 속도보다 SATA 인터페이스의 대역폭이 더 낮아 전송 병목 현상이 발생했다. 하지만 NVMe와 같은 고성능 인터페이스가 등장하면서 상황이 바뀌었다. NVMe는 SSD를 위해 특화된 설계로 높은 대역폭을 제공하지만 SSD 내부의 요청 스케줄링이 최적화되지 않으면 전체 시스템 성능이 병목에 걸리게 된다.



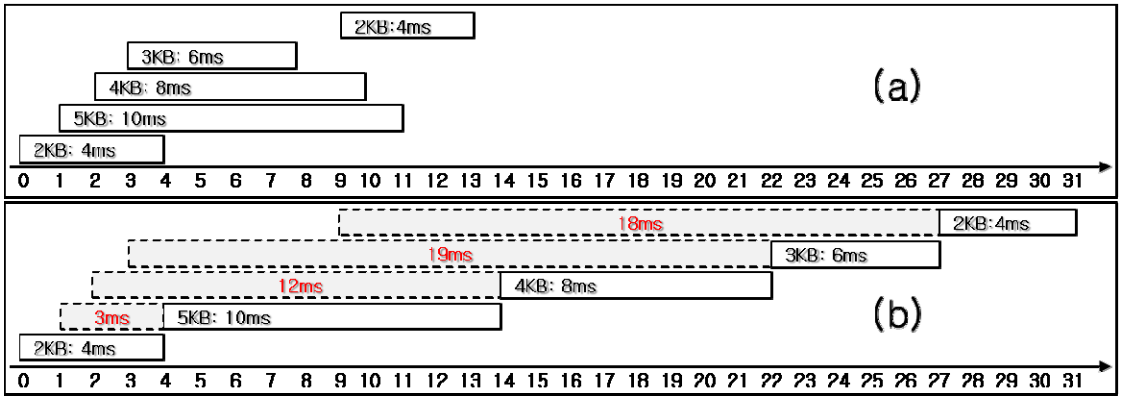
[Fig. 2] Structure of Internal Request Queue

Fig. 2는 SSD 내부의 요청 큐 구조를 보여준다. 그림과 같이 데이터 처리 요청은 들어온 순서대로 큐에 쌓이고, 순서대로 작업이 진행된다. 즉 요청된 작업의 순서대로 데이터 처리 및 전송이 진행된다. 그러나 대용량 데이터 집약 어플리케이션에서 먼저 도착한 긴 전송 길이 요청이 진행될 경우, 뒤이어 도착한 다수의 짧은 요청들이 오랜 시간 대기해야 한다. 이 때문에 평균 응답 시간이 크게 증가하여 전체 시스템 성능이 저하된다. 이러한 문제는 고성능 SSD[12-14]와 고대역폭의 NVMe 인터페이스 환경에서도 동일하게 발생할 수 있다. 따라서 SSD의 성능을 극대화하고, 평균 응답 시간을 단축하기 위해서는, 호스트 요청에 대해 요청의 특성에 따라 전송 순서와 타이밍을 조절하고 내부 스케줄링 정책을 최적화할 필요가 있다.

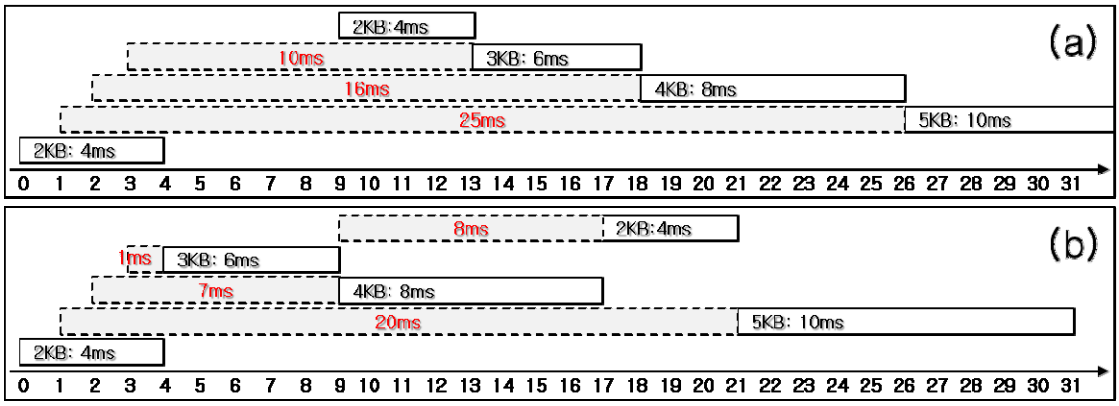
3. 전송 큐 스케줄링의 문제점 분석

3.1 요청 순서에 의한 큐잉 전략의 문제점 분석

Fig. 3은 임의의 요청 시간에 각각 서로 다른 데이터 전송 요청이 처리되는 과정을 보여주고 있다. Fig. 3 (a)에서 x 좌표의 선은 0ms부터 각각 31ms까지 1ms씩 증가하는 시간의 흐름을 보여준다. 그리고 각 데이터 전송 시간은 512B당 1ms의 시간이 필요하다고 가정하였다. 예를 들어 그림의 2KB는 512B를 4번 전송하는 용량이기 때문에 총 4ms의 전송시간이 필요하다. Fig. 3 (a)의 예제에서는 각각 2KB, 5KB, 4KB, 3KB, 2KB 크기의 데이터 전송 요청을 0ms, 1ms, 2ms, 3ms, 9ms의 시간에 받았다. 이 전송 요청을 들어온 입력의 순서대로 FIFO(First In First Out)로 처리하면 Fig. 3 (b)의 예제와 같다. 그림의 예제에서는 요청된 순서대로 데이터 전송을 처리하였다. 따라서 총 데이터 전송 시간은 각각의 데이터 처리 및 전송시간을 합산한 31ms의 시간이 소모되었다. 그러나 각각의 데이터 전송이 요청받은 시간에서 전송 시작까지 지연된 시간은 순서대로 0, 3, 12, 19,



[Fig. 3] Problem with Queue Sending in the Requested Order



[Fig. 4] Problem with Transfer Ording Based on Data Length

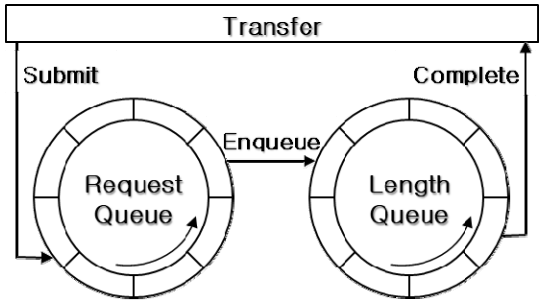
18ms이다. 결과적으로 각각의 데이터 처리가 완료하는데 소모된 시간은 4ms, 13ms, 20ms, 25ms, 22ms이다. 이것은 데이터 처리를 시작하기까지 평균적으로 약 10.4ms의 시간이 지연되었고, 총 31ms의 데이터를 전송시간이 필요한 데이터를 전송 완료하는데 소요된 시간은 82ms의 시간이 소모되어 사실상 1ms의 시간이 필요한 512B를 전송하는 데 필요한 시간은 평균 2.64ms를 의미한다. 이 결과는 요청의 순서대로 데이터를 처리했을 때 데이터를 전송 시작하기 전까지 지연되는 시간이 데이터를 전송하는 시간보다 약 1.6배 이상 높다는 것을 의미한다.

3.2 전송 길이에 의한 큐잉 전략의 문제점 분석

Fig. 4는 전송 길이를 기반으로 데이터 전송을 처리하는 과정과 문제점을 보여주고 있다. Fig. 4 (a)는 순수하게 전체 데이터의 전송 길이를 순서로 스케줄링하고 저장장치의 내부적으로 데이터가 올라오는 것을 기다렸다가 처리하였다. 이 경우 총 지연시간과 평균 지연시간은 각각 51ms와 10.2ms이다. Fig. 4 (b)의 경우 조금 더 지연시간을 줄이기 위해 입력된 순서를 고려하였고, 총 지연시간과 평균 지연시간은 각각 36ms와 7.2ms로 가장 좋은 성능을 보였다. 그러나 (b)의 5KB는 두 번째로 요청이 되었으나 다른 데이터 전송이 완료될 때까지 장기 대기하는 문제가 있다.

4. 전송 길이를 고려한 이중 큐 전략

4.1 핵심 아이디어



[Fig. 5] Key Idea of DQBS

Fig. 5는 제안하는 아이디어인 DQBS의 핵심 아이디어를 보여주고 있다. DQBS는 그림과 같이 먼저 들어온 요청 순서를 우선순위로 하는 요청 큐와 짧은 데이터의 길이를 우선순위로 하는 길이 큐로 구성되어 있다. 요청 큐는 먼저 요청된 데이터 순서대로 데이터 전송을 처리하기 위한 1차 큐이다. 길이 큐는 전송 길이를 기반으로 우선순위를 정하기 위한 2차 큐이다. DQBS는 평균 전송 지연시간을 줄이고 동시에 장기 지연되는 작업을 회피하기 위해 1, 2차큐와 장기지연 회피 전략을 사용한다. 이를 위해 DQBS는 먼저 요청된 데이터는 요청 순서에 따라 요청 큐에 삽입된다. 그다음 요청된 순서대로 요청 큐에서 길이 큐로 삽입된다. 이때 삽입되는 작업은 장기 대기 시간을 고려하여 우선순위 가중치를 적용한다. 마지막으로 길이 큐에서는 길이가 요청 이후 대기 시간과 요청된 데이터 길이를 고려하여 짧은 순서대로 전송을 진행한다.

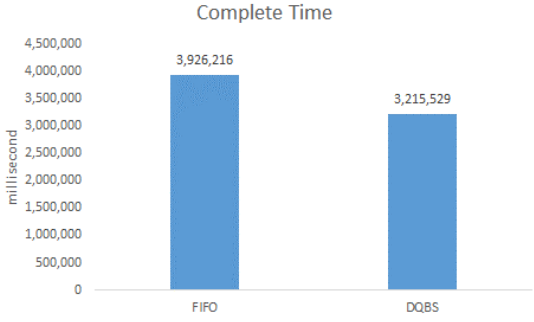
4.2 에이징 회피 정책

평균 지연시간을 줄이기 위해 DQBS에서는 두 종류의

이중 큐를 사용하였다. 1차 큐는 요청된 순서대로 응답을 처리한다. 2차 큐는 1차 큐에 대기 중인 데이터의 순서를 변경하여 응답 시간을 개선한다. 2차 큐는 응답 시간을 개선하기 위해 전송 데이터 길이를 기준으로 전송 순서를 재정렬하는 우선순위 큐 기법을 적용하였다. 그러나 단순히 짧은 데이터 길이 순서로 데이터를 처리할 경우, 길이가 긴 요청은 처리되지 않고 장기대기 문제가 발생할 수 있다. 이러한 에이징 문제를 회피하기 위해 다음과 같은 에이징 회피 전략을 적용하였다. 먼저, 1차 큐에서 요청 순서대로 쌓여있는 큐들을 2차 큐에 관리하여 우선순위를 재조정한다. 이때, 2차 큐로 삽입된 요청 중 우선순위가 가장 낮은 요청의 시작과 예상 완료 시간을 합산하여 예상 완료 시간을 계산한다. 그다음 이 1차 큐에서 2차 큐로 삽입 대기 중인 요청의 예상 완료 시간보다 응답 짧은 경우 이 요청이 완료될 때까지 1차 큐에서 신규 삽입되는 요청을 지연한다. DQBS는 이러한 방법을 통해 데이터가 긴 데이터의 요청이 우선순위에 밀려 장기 대기하는 문제를 해결하였다.

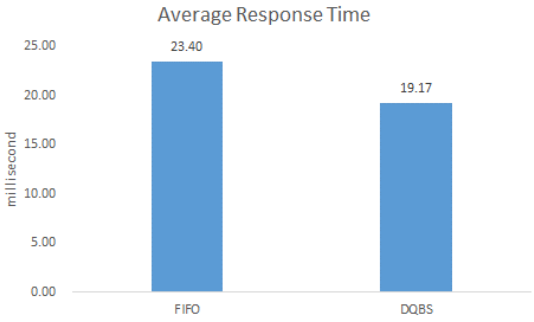
5. 실험 및 평가

제안하는 아이디어의 우수성을 평가하기 위해 시뮬레이션을 진행했다. 실험에서 사용한 데이터는 엔터프라이즈에서 실제 요청된 데이터의 읽기 트레이스[15]를 이용하였다. 플래시 메모리의 페이지 크기는 512B로 가정하였고, 한 페이지에 대한 요청을 처리하는 비용은 1ms로 계산하였다.



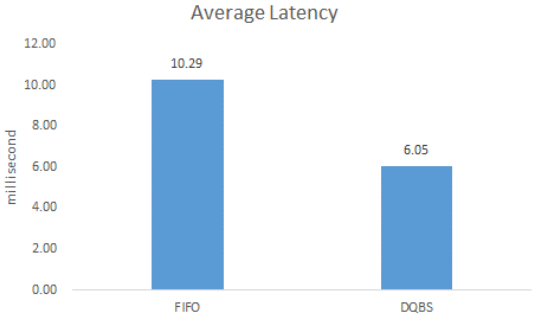
[Fig. 6] Complete Time

Fig. 6은 167,764개의 명령을 처리했을 때 완료된 시간의 합이다. 처리된 총 작업의 크기는 약 537.09GB다. 모든 명령을 FIFO를 이용하여 스케줄링한 경우 3,926,216ms의 시간이 소모되었다. 반면 논문에서 제안하는 DQBS는 3,215,216ms의 시간이 소모되었다. 즉, 제안하는 기법이 단순히 순차적으로 처리하는 FIFO의 성능과 비교하여 약 18.1% 성능이 향상되는 것을 확인하였다.



[Fig. 7] Average Response Time

Fig. 7은 명령들을 처리하는 동안 평균 응답 시간의 결과를 보여주고 있다. FIFO는 하나의 명령 당 23.40ms의 응답 속도를 보여주었다. 반면 DQBS는 19.17ms의 응답 속도를 보여주었다. 따라서 제안하는 DQBS 기법이 우수한 성능을 보여주는 것을 확인할 수 있었다.



[Fig. 8] Average Latency

Fig. 8은 평균 지연시간의 결과를 보여주고 있다. 그림과 같이 각 명령이 시작되기 전까지 FIFO는 10.29ms가 지연되었고, DQBS는 6.05ms가 지연되었다. 결과적으로 DQBS가 FIFO보다 약 41.16%의 우수한 지연 성능을 보여주었다. 시뮬레이션을 통해 DQBS가 단순히 순차적으로 요청된 데이터를 처리하는 것보다 우수한 성능을 보여주었다.

6. 결론

본 논문에서는 저장장치의 응답 및 지연 시간 문제를 향상하기 위해 이중 큐를 사용하는 DQBS 기법을 제안하였다. 첫 번째 큐는 요청의 순서대로 동작 스케줄을 관리하였고, 두 번째 큐는 데이터의 길이를 기반으로 스케줄링하는 우선순위 큐 기법을 적용하였다. 또한, 에이징 문제를 회피하기 위한 회피 정책을 적용하였다. 마지막으로 실험을 통해 제안하는 기법이 순차적으로 데이터 요청을 처리하는 방법과 비교하여 우수한 응답 시간 및 지연시간의 성능을 보여주는 것을 확인하였다. 향후에는 스케줄링을 위해 소모되는 자원을 최적화하기 위한 연구를 진행할 예정이다.

REFERENCES

[1] M.Chen, S.Mao, and Y.Liu, "Big data: A survey," *Mobile Networks and Applications* Vol.19, No.2, pp.171-209, 2014.

[2] M.Marmol, Y.Fernandez, and P.Cuenca, "Enterprise Storage Cloud: A Hybrid Cloud Storage System," *Sensors*, Vol.19, No.18, 2019.

[3] C.Luo, E.Shedulund, J.Nelson, L.Frederick, E.Canino, and P.Zhang, "Data Analytics Applications and Performance in Large Data Center Environments," *IEEE Access*, Vol.7, pp.71176-71189, 2019.

[4] NVM Express Specification Revision 1.4, *NVM Express, Inc.*, 2019.

[5] J.Lee, Y.Kim, G.M.Shipman, S.Oral, and F.Wang, "Constructing energy efficient manycore parallel data movement on modern SSD arrays," *IEEE International Symposium on High-Performance Parallel and Distributed Computing*, pp.61-72, 2018.

[6] R.Salkhordeh and H.R.Tavanir, "Analytical Performance Models for MQ-based Multi-stream SSD Architecture," *IEEE Transactions on Computers*, Vol.64, No.5, pp.1323-1336, 2015.

[7] Y.Park, J.Cha, H.Lee, and S.Park, "Understanding NVMe Performance at Enterprise Scale," *ACM Transactions on Storage*, Vol.18, No.4, pp.1-29, 2022.

[8] D.Jo, Y.Kwon, H.Kim, and, C. Kyung, "Hit-Ratio Aware Delay-Capped Request Scheduling for NVMe SSD," *IEEE Access*, Vol.8, pp.135525-135538, 2020.

[9] S.Yun, J.Shin, and M.Erez, "Reducing the Tail Latency in NVMe SSDs," *IEEE Computer Architecture Letters*, Vol.19, No.2, pp.129-132, 2020.

[10] Y.Kim, S.Oral, G.M.Shipman, J.Lee, D.Dillow and F.Wang, "Orchestrating an Ensemble of Datacenter SSDs for High-Performance Sequential Data Accesses," *ACM Transactions on Storage,* Vol.10, No.4, pp.1-28, 2014.

[11] M.Bjørling, P.Bonnet, L.Bouganim and N.Dayan, "TheQuery-Friendly NVM Express Key-Value SSD," *IEEE Data Engineering Bulletin*, Vol.40, No.1, pp.36-48, 2017.

[12] 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.

[13] 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.

[14] 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.

[15] 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