논문 2019-14-08

초저지연 저장장치를 위한 적응형 폴링 선택 기법

(An Adaptive Polling Selection Technique for Ultra-Low Latency Storage Systems)

천 명 준, 김 윤 아, 김 지 홍*
(Myoungjun Chun, Yoona Kim, Jihong Kim)

Abstract : Recently, ultra-low latency flash storage devices such as Z-SSD and Optane SSD were introduced with the significant technological improvement in the storage devices which provide much faster response time than today's other NVMe SSDs. With such ultra-low latency, 10μs, storage devices the cost of context switch could be an overhead during interrupt-driven I/O completion process. As an interrupt-driven I/O completion process could bring an interrupt handling overhead, polling or hybrid-polling for the I/O completion is known to perform better. In this paper, we analyze tail latency problem in a polling process caused by process scheduling in data center environment where multiple applications run simultaneously under one system and we introduce our adaptive polling selection technique which dynamically selects efficient processing method between two techniques according to the system's conditions.

Keywords : Flash memory, Storage system, Polling, I/O stack optimization

Ⅰ. 서 론

전통적으로 운영체제는 I/O 요청을 처리하기 위한 완료 통지방식으로 비 동기적으로 동작하는 인터럽트를 사용해 왔다. 인터럽트를 사용한 I/O 완료 통지에서는 사용자 프로세스 (User Process)는 저장장치에서 I/O가 처리되는 동안 Sleep 상태로 대기하며, 저장장치에서 I/O 처리가 끝나면 인터럽트 핸들러가 호출되어 해당하는 사용자 프로세스를 실행하도록 하며 I/O가 완료되는 방식이다. 인터럽트 방식은 저장장치가 I/O를 처리하는 중 CPU를 양보하기 때문에 다른 프로세스가 그사이 CPU를 점유할 수 있다는 장점이 있지만, I/O가 끝날 때 인터럽트 핸들러가 처리되는 시간과 문맥 교환 (Context Switch) 비용이 I/O 응답시간에 포함되게 된다.

한편, 최근 저장장치 시장에 소개된 Z-NAND 기반 Samsung Z-SSD [1]나 3D XPoint 기반 Intel Optane SSD [2]들은 각각 12μs, 10μs의 초저지연 읽기 응답시간을 제공하며, 차세대 저장장치의 응답시간은 이보다 더욱 감소할 것으로 기대된다.

이러한 빠른 응답시간을 제공하는 저장장치에서는 인터럽트 방식의 인터럽트 핸들러 처리 시간과 문맥 교환 비용이 새로운 병목 지점이 될 수 있어, 동기적으로 I/O가 완료되었는지 저장장치의 상태를 계속하여 검사하는 폴링 (Polling) 방식을 사용하는 연구가 기 제안되었다 [3]. I/O가 저장장치에서 처리되는 동안 CPU를 양보하지 않으므로, 인터럽트 방식의 단점이었던 인터럽트 핸들러 처리 시간과 문맥 교환 비용이 I/O 응답시간에 포함되지 않아 빠른 응답시간을 제공하는 차세대 저장장치에 적합하다. 하지만 폴링 방식은 I/O 완료 여부를 검사하는 과정에서 프로세스가 필연적으로 CPU를 계속 점유하여야 하는 한계가 있다. 하이브리드 폴링 (Hybrid Polling)은 프로세스가 타이머를 사용해 예측한 저장장치의 I/O 처리 시간만큼을 Sleep 한 뒤 깨어나 폴링을 시작함으로써 폴링의 빠른 I/O 응답시간과 인

*Corresponding Author (jihong@davinci.snu.ac.kr)
Received: Feb. 24, 2019, Revised: Mar. 19, 2019, Accepted: Apr. 01, 2019.
M. Chun, Y. Kim, J. Kim: Seoul National University.
※ 본 논문은 정부(미래창조과학부)의 재원으로 한국연구재단의 (NRF-2018R1A2B6006878) 지원을 받아 연구하였음.

 2019 Apr. 14(2) 63-69
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2019.14.2.63

터럽트의 낮은 CPU 이용률을 모두 취하기 위해 제안된 방식 [4] 이다. 가장 이상적인 방식이지만, 저장장치의 I/O 처리 시간에 대한 정확한 예측이 이루어지지 않을 경우, I/O 응답시간의 지연이 발생한다.

본 연구에서는 데이터센터와 같이 여러 응용이 하나의 시스템에서 동작하는 환경에서 폴링과 하이브리드 폴링의 부작용을 분석하였다. 이러한 환경에서는, 여러 응용이 동시에 CPU와 저장장치를 공유하기 때문에 프로세스 스케줄링에서의 응용 간 CPU 경쟁과 저장장치에서의 I/O 요청 간의 간섭이 발생하게 된다. 먼저, 폴링 방식은 여러 응용이 동시에 CPU 사용을 위해 경쟁할 때 I/O 프로세스가 폴링 시 사용한 과도한 CPU 타임 퀀텀 (Time Quantum)으로 인해 자주 선점되어 I/O를 추가로 요청하거나 완료하지 못하고, 꼬리 응답시간이 크게 지연되는 문제가 관찰되었다. 반면에, 하이브리드 폴링 방식은 여러 응용이 동시에 초저지연 저장장치를 공유하여 I/O 요청 간의 간섭 정도가 높은 경우, I/O 처리 시간에 대한 예측이 잘못되는 경우가 잦아져 평균 응답시간이 지연되는 것을 관찰하였다.

본 논문에서는 폴링과 하이브리드 폴링 방식이 상호 보완적일 수 있다는 고찰을 통해, 현재 시스템 CPU 이용률과 I/O 요청이 집중되는 정도를 관찰하여 현재 상황에서 가장 알맞은 I/O 완료 통지방식을 선택하는 적응형 폴링 선택 기법인 APS (Adaptive Polling Selection)를 제안한다.

논문의 구성은 다음과 같다. 먼저 2장에서는 리눅스 커널에 구현된 폴링, 하이브리드 폴링 방식을 분석한다. 3장에서는 여러 시스템 CPU이용률 상황에서 인터럽트, 폴링, 하이브리드 폴링 I/O 통지방식을 평가한 결과와 부작용에 대해 분석한다. 4장에서는 제안하는 적응형 폴링 선택 기법의 동작 원리에 대해 설명하고 성능 개선에 대한 평가한 결과를 보인 후, 5장에서 결론을 맺는다.

Ⅱ. 폴링, 하이브리드 폴링

리눅스 커널에서 폴링과 하이브리드 폴링은 블록 계층에 구현되어 있으며 블록 I/O이고 IOCB_HIPRI 플래그가 설정된 Direct I/O에 한해 시도된다 [4-6]. 표 1은 본 장에서 살펴볼 주요 구성 함수의 이름과 역할을 보여준다.

먼저 blk_mq_poll 함수는 폴링이 수행될 때 가장 처음으로 호출되며 NVMe 드라이버 계층의 nvme_poll 함수를 통해 저장장치의 I/O 처리가 끝

표 1. 폴링 구현 구성 함수
Table 1. Set of polling functions

Function Name	Operation
blk_mq_poll	Check I/O completion of the storage device by busy-spinning
blk_mq_poll_hybrid_sleep	Transit to sleep state for a specified time
blk_mq_poll_nsec	Determine the amount of time for a process to sleep

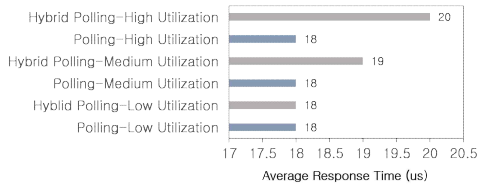
났는지 반복하며 확인한다. 이 과정에서, 프로세스는 인터럽트 방식을 사용했을 때보다 추가적인 타임 퀀텀을 소모하게 된다.

blk_mq_poll_hybrid_sleep 함수는 blk_mq_poll 함수의 시작 부분에서 하이브리드 폴링을 사용할 경우에 호출되며, Hrtimer를 사용하여 프로세스를 지정한 시간 동안 Sleep 상태로 바꾸는 역할을 한다. Sleep 상태가 된 프로세스는 지정한 시간이 끝나면 문맥 교환을 통해 blk_mq_poll 함수로 돌아가 폴링을 시작한다. blk_mq_poll_nsec 함수는 하이브리드 폴링을 사용할 때만 blk_mq_poll_hybrid_polling 함수의 시작 부분에서 호출되며, 프로세스가 Sleep 할 시간을 결정한다. 이는 과거의 I/O 완료 시간을 통해 현재 I/O 요청의 완료 시간을 예측함으로써 이루어진다. 기본적으로 설정된 Sleep 시간은 최근 16개 I/O 완료 시간의 평균을 2로 나눈 값이다. 이상적인 하이브리드 폴링의 경우, Sleep이 끝나고 폴링을 시작한 직후 저장장치에서 I/O 처리가 끝나 완료 통지가 바로 이루어진다. 하지만, Sleep 시간보다 I/O 처리가 빨리 끝날 경우는 인터럽트와 같이 I/O 응답시간에 인터럽트 핸들러와 문맥 교환 비용이 추가된다. 반대로 I/O 처리가 완료되는 시점을 실제보다 일찍 예측하여 Sleep 상태에서 깨어날 경우, 폴링 방식과 같이 추가적인 타임 퀀텀을 소모하게 된다.

Ⅲ. I/O 완료 통지방식별 응답시간 평가

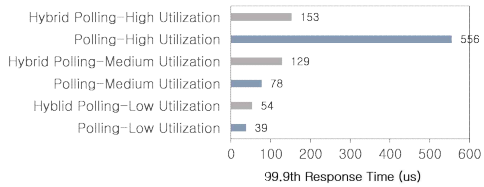
1. 실험 환경 및 각 통지방식의 부작용 분석

본 실험의 목표는 여러 응용이 하나의 시스템에서 동작할 때 폴링과 하이브리드 폴링 방식의 부작용이 읽기 요청에 대한 평균 응답시간과 꼬리 응답시간에 드러나는 것을 관찰하고, 원인을 분석하여



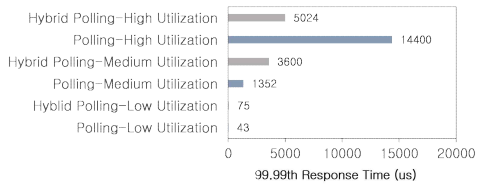
(a) 평균 읽기 응답시간

(a) Average response time



(b) 99.9 퍼센타일 읽기 응답시간

(b) 99.9th response time



(c) 99.99 퍼센타일 읽기 응답시간

(c) 99.99th response time

그림 1. 시스템 CPU 이용률에 따른 읽기 응답시간

Fig. 1 Read response time under varying system CPU utilization

연구의 동기를 얻기 위함이다. 모든 실험은 Intel i7-4790 3.6Ghz 8-스레드 CPU, PC3-10600 16GB 메인 메모리, 리눅스 커널 4.14.36-stable 환경에서 Samsung SZ985 Z-NAND SSD를 사용하여 진행하였다. 응답시간 분포 측정을 위해 저장장치 벤치마크로 FIO [7]를 사용하여 4개의 쓰레드, 4의 큐 깊이 설정으로 10GB 파일에 대한 4k 임의 읽기 요청에 대한 응답시간을 폴링, 하이브리드 폴링의 2가지 I/O 통지방식별로 측정하였다. 또한, 여러 응용이 하나의 시스템에서 동작하는 환경을 만들기 위해, CPU 계산 작업과 I/O 작업을 모두 수행하는 그래프 마이닝 응용인 Pegasus [8]로 그래프 마이닝을 응답시간 측정을 위한 FIO와 동시에 수행하여 CPU 경쟁과 저장장치 간섭이 동시에 일어나도록 하였다. 그래프 마이닝 작업을 하는 프로세스가 하나일 때를 Low Utilization, 4개일 때를 Medium Utilization, 8개일 때를 High Utilization으로 분리

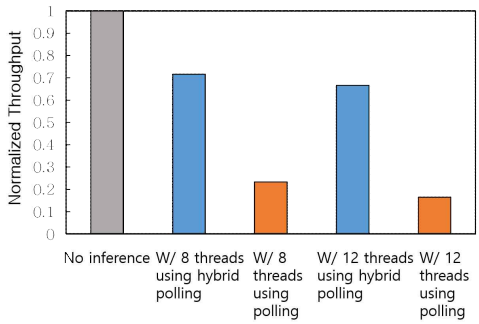


그림 2. I/O 프로세스의 통지방식별 그래프 마이닝 응용의 처리량 비교

Fig. 2 Comparison of graph mining application throughput of different I/O completion methods

하여 간섭에 정도에 따른 변화를 관찰하였다.

그림 1는 그래프 마이닝 응용의 수를 늘려 가며 측정한 폴링과 하이브리드 폴링 방식의 읽기 응답시간 분포를 보여준다. 그래프 마이닝 응용의 수가 적은 Low Utilization 상황에서는 폴링과 하이브리드 방식이 비슷한 평균 응답시간을 보여주며 99.9, 99.99 퍼센타일의 응답시간 또한 큰 차이가 없는 모습이다. 이러한 결과를 통해, CPU 자원에 여유가 있고 I/O 요청 간의 간섭도 심하지 않기 때문에 두 방식의 부작용이 드러나지 않는 환경이라는 것을 확인할 수 있다.

하지만 Medium, High Utilization 순서로 여러 프로세스가 CPU를 경쟁하는 정도가 심해짐에 따라 폴링 방식의 꼬리 응답시간은 점점 악화되는 양상을 보이며, High Utilization 상황처럼 CPU를 점유하기 힘든 상황에서는 하이브리드 폴링보다 3배가량 긴 14400us의 꼬리 응답시간이 관찰되었다. 폴링 방식은 저장장치에서 I/O 요청이 처리되는 과정에도 프로세스가 Sleep을 하지 않기 때문에 하이브리드 폴링이나 인터럽트 방식을 사용할 때보다 같은 양의 I/O 요청을 처리하더라도 CPU 타임 퀀텀을 더 많이 사용하게 된다. 따라서 같은 양의 I/O 요청 처리 과정에서도 프로세스가 선점되는 경우가 더 자주 발생하게 되고, 프로세스가 선점당한 동안은 I/O 요청에 대한 처리가 이루어지지 못하기 때문에, 선점된 사이의 I/O 요청들은 응답시간이 다시 프로세스가 스케줄링 되기까지로 지연되어 긴 꼬리 응답시간을 만든다. 이를 확인하기 위해 폴링을 사용할 때의 FIO 벤치마크의 CPU 이용률을 각 Utilization 상황별로 측정해 보았다. Low

Utilization 상황에선 96.29%, Medium Utilization 상황에선 92.98%로 100%에 근접한 이용률을 보여 프로세스 선점이 거의 일어나지 않은 것을 확인할 수 있다. 하지만 High Utilization 상황에서는 CPU 이용률이 68.27%에 불과하여, 폴링을 하던 프로세스가 CPU를 빼앗기는 선점이 자주 이루어졌으며 다시 스케줄링 되기까지 오랜 지연이 생기는 경우가 실제 잦았음을 알 수 있다.

하이브리드 폴링 방식은 I/O 처리 중 타임 퀀텀을 폴링 방식보다 적게 사용함으로, 꼬리 응답시간에서는 폴링 방식보다 좋은 결과를 보였다. 하지만, High Utilization 상황에서는 평균 응답시간에서 폴링 방식보다 2us 가량 지연이 더 발생하였는데, 이는 인터럽트 방식을 사용할 때와 비슷한 응답시간이다. 이러한 지연의 원인은 리눅스 커널에서 구현된 하이브리드 폴링이 저장장치의 I/O 처리 시간을 예측하는 루틴과 SSD의 특성에서 찾을 수 있다. 앞서 설명했듯, 리눅스 커널에 구현된 하이브리드 폴링의 프로세스가 Sleep 시간은 최근 완료된 16개 I/O 처리 시간의 평균에 기반을 둔 예측으로 결정된다. 하지만, SSD는 여러 I/O 요청 간의 간섭이 심할 때 가비지 컬렉션과 같은 내부 작업 등의 영향으로 수백 배에 달하는 꼬리 응답시간이 보이는 것이 이미 널리 알려진 바 있다 [9]. 따라서, 커널 계층의 소수의 과거 I/O 처리 시간을 기반으로 한 예측은 I/O 요청 간의 간섭으로 인해 하나의 불균일한 I/O 처리 시간으로 인하여 크게 왜곡되는 경우가 발생하고, 이로 인해 프로세스가 I/O 처리 완료 이후에 깨어남으로써 인터럽트 방식과 같이 동작하여 평균 응답시간이 지연되는 것으로 보인다. ZSSD와 같은 초저지연 저장장치가 사용되는 환경이, 스왑 공간을 위한 빠른 캐시나 웹 서버와 같은 읽기 응답시간이 매우 중요한 환경임을 고려해 본다면 디바이스 응답시간의 16.67% 정도에 해당하는 2us 가량의 지연도 충분히 사용자 경험에 악영향을 미칠 가능성이 있다.

2. 개선 방안 분석

앞서 폴링 방식의 꼬리 응답시간 지연은 잦은 프로세스 스케줄링에 의한 선점 때문임을 지적한 바 있다. 따라서, 프로세스의 스케줄링 우선순위를 높여 선점되지 않도록 하는 것은 직관적으로 생각할 수 있는 해결 방안이다. 하지만, 본 논문의 여러 응용 간의 CPU 경쟁이 발생하는 실험 환경에서는 프로세스 스케줄링 정책의 변경이 다른 응용에 크게 영향을 미쳐 성능 저하를 야기할 수 있다. 그림 2는 폴링을 사용하는 I/O 프로세스의 스케줄링 우선순위를 SCHED_RR로 설정하여 선점되지 않도록 하였을 때와 하이브리드 폴링 방식을 사용했을 때, 동시에 수행하는 그래프 마이닝 응용의 처리량의 차이를 간섭이 없을 때와 비교하여 나타낸 것이다. 간섭이 없을 때와 비교하여, I/O 프로세스 응용이 하이브리드 폴링을 사용할 때는 처리량이 최대 33% 저하되었지만, 스케줄링 우선순위를 높여 폴링을 사용할 때는 최대 84% 처리량이 저하되는 것을 확인할 수 있다. 결과를 통해, 폴링 I/O 프로세스의 스케줄링 우선순위를 올려 주는 기법은 여러 응용이 동시에 동작하는 환경에서는, 다른 응용의 처리량을 심각하게 감소시키는 부작용이 있어 적용하기 어려움을 알 수 있다.

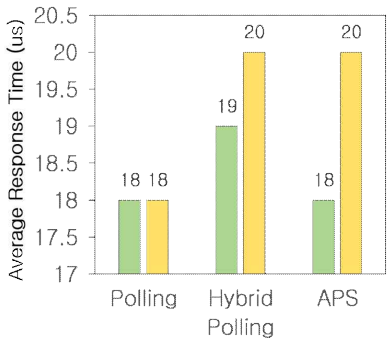
하이브리드 폴링의 평균 지연시간이 저하되는 원인은 커널 계층에서 저장장치의 I/O 처리 시간을 정확히 판단하지 못했기 때문이다. 이를 해결하기 위해, 커널의 I/O 처리 시간 예측 방식을 개선함으로써 예측의 정확도를 높이는 방법이 있다. 하지만, 저장장치의 상태를 알 수 없는 커널 계층의 한정된 정보만으로 저장장치에 I/O 요청을 보내기 전 해당 I/O 요청에 대한 처리 시간을 정확히 예측하는 것을 매우 어려운 과제이다. 또한, 한 저장장치에 맞춰 커널의 예측 기법을 수정하였더라도, 특성이 다른 저장장치를 사용할 때도 수정한 예측 기법이 정확하리라고 보장할 수 없는 한계가 여전히 존재한다.

본 논문에서는 초저지연 저장장치의 특성을 살리기 위해서는 가능한 폴링과 같이 동작하는 것을 기본으로 하되, 시스템 전체의 CPU 이용률이 높은 상황에서 폴링을 사용하는 것이 다른 응용에 끼치는 악영향을 최소화하기 위해, 시스템 전체의 CPU 이용률을 커널 계층에서 관찰하여 이를 기반으로 폴링과 하이브리드 폴링을 선택해서 사용하는 기법을 제안한다. 제안하는 기법은 폴링에 가까운 평균 응답시간과, 하이브리드 폴링에 가까운 꼬리 응답시간을 보이는 것을 목표로 한다.

Ⅳ. 적응형 폴링 선택 기법

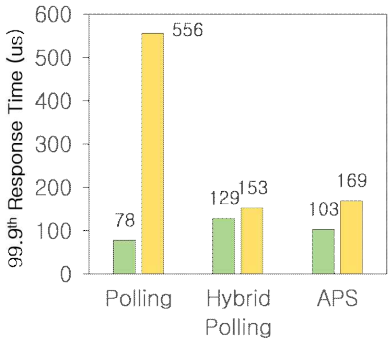
1. 동작 원리

적응형 폴링 선택 기법은 현재 시스템 CPU 이용률에 따라 폴링과 하이브리드 폴링 방식의 효율성이 변화하는 것에 착안하여, 시스템 CPU 이용률을 예측한 뒤 현재 상황에서 가장 적합하다고 판단된 I/O 통지방식을 사용하는 기법이다.



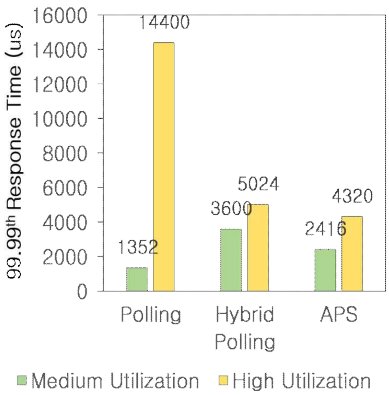
(a) 평균 읽기 응답시간

(a) Average read response time



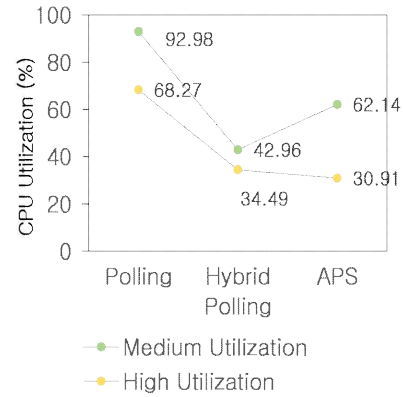
(b) 99.9 퍼센타일 읽기 응답시간

(b) 99.9th read response time



(c) 99.99 퍼센타일 읽기 응답시간

(c) 99.99th read response time



(d) CPU 이용률

(d) CPU utilization

그림 3. 적응형 폴링 선택 (APS) 기법 평가

Fig. 3 Evaluation of Adaptive Polling Selection

이를 위해, 새로운 I/O 요청이 들어오면 블록 계층에서 폴링과 하이브리드 폴링 중 어떠한 방식이 적합할지 올바르게 예측하여야 한다. 만약 새로운 I/O 요청이 들어온 시점의 시스템 CPU 이용률만을 가지고 판단할 경우 일시적으로 CPU 이용률이 변화하는 시점에는 예측의 정확도가 떨어질 수 있다. 따라서 최근 16개 I/O 요청이 완료된 시점의 CPU 이용률을 추가적으로 수집해 두고, 새로운 I/O 요청이 들어오면 현재 CPU 이용률과 과거 CPU 이용률을 종합하여 현재 사용할 수 있는 CPU 이용률의 마진(Margin)을 계산한다. 계산된 마진 수치가 높다면 폴링을 사용해도 무방하다고 판단하여 폴링을 사용하고, 마진 수치가 중간이라면 적당한 수치(기본 구현에서는 최근 16개의 I/O 완료 시간 평균을 2로 나눈 값)로 하이브리드 폴링을 사용한다. 마진 수치가 낮아 CPU 이용률에 여유가 없다면, 하이브리드 폴링을 사용하되 최대한 많은 시간 동안 프로세스를 Sleep 시켜 인터럽트에 가깝게 동작하도록 한다.

2. 기법 평가

본 실험에서는 앞 2장에서 폴링과 하이브리드 폴링 방식의 부작용이 발생했던 Medium Utilization, High Utilization 상황에서 적응형 폴링 선택 기법을 통해 평균, 99.9 퍼센타일, 99.99 퍼센타일의 읽기 응답시간이 단축된 정도를 기존 방식들과의 비교로 평가한다. 또한, 읽기 요청을 보내는 응용인 FIO 벤치마크의 CPU 이용률을 측정하여 기

법이 의도한 대로, CPU를 과도하게 사용하지 않음으로써 다른 응용의 성능에 간섭을 적게 일으키고 있는지 분석하였다.

그림 3의 (a)는 시스템 CPU 이용률에 따른 각 방식의 평균 읽기 응답시간을, (b)는 99.9 퍼센타일 읽기 응답시간을, (c)는 99.99 퍼센타일 읽기 응답시간을 나타낸다. 적응형 폴링 선택 (APS) 기법은 시스템 CPU 이용률이 낮아 마진이 클 때는 평균과 꼬리 응답시간 모두 폴링에 가까운 안정적인 응답시간을 달성했으며, 시스템 CPU 이용률이 높아 마진이 얼마 없는 상황에서는 하이브리드 폴링에 가깝거나 소폭 개선된 응답시간을 나타내는 것을 확인하였다. 구체적으로, 99.9 퍼센타일의 꼬리 응답시간이 Medium Utilization 상황에서 하이브리드 폴링에 비해 20.15% 감소하였으며 High Utilization 상황에서 폴링에 비해 69.23% 감소하였다. 99.99 퍼센타일의 꼬리 응답시간의 개선은 Medium Utilization 상황에서 32.89%, High Utilization 상황에서 70.00% 감소하여 더 큰 향상폭을 보였다.

그림 3의 (d)는 읽기 요청을 발생시키는 FIO 벤치마크의 CPU 이용률을 보이고 있다. 마진이 높은 Medium Utilization 상황에서 적응형 폴링 선택 기법은 CPU를 상대적으로 높은 마진만큼 더 사용하는 정책하에 동작하므로, 하이브리드 폴링 보다는 높고 폴링보다는 낮은 CPU 이용률을 보이면서도 폴링에 가까운 평균 응답시간의 개선이 있었다. 마진이 얼마 없는 High Utilization 상황에서는 CPU를 하이브리드 폴링보다도 적게 사용하는 인터럽트에 가까운 방식으로 동작하면서도, 폴링에 비해 99.99 퍼센타일 꼬리 응답시간을 소폭 개선할 수 있음을 확인하였다.

V. 결 론

본 논문에서는 초저지연 저장장치에서 인터럽트 처리의 비용을 줄이기 위해 사용되는 폴링과 하이브리드 폴링 방식이 시스템 CPU 이용률의 변화에 따라 긴 읽기 꼬리 응답시간을 만드는 문제가 있음을 지적하고 이를 개선한 적응형 폴링 선택 기법을 제안하였다. 제안한 기법은 시스템 CPU 이용률을 예측하여 I/O 요청이 들어온 시점에 가장 효율적일 것으로 기대되는 I/O 완료 통지방법을 선택적으로 적용함으로써, 폴링과 하이브리드 폴링의 장점을 모두 취할 수 있음을 실제 저장장치에서의 평가를 통해 확인하였다.

References

[1] Samsung Z-SSD SZ985, Avaliable on : https://www.samsung. com/us/labs/pdfs/-collateral/Samsung_Z-NAND_Technology_Brief_v5.pdf.

[2] Intel Optane SSD 900P, Avaliable on : https://www.intel.com/content/www/us/en/-products-memory-storage/solid-state-drives/gaming-enthusiast-ssds/optane-900p-series.html.

[3] J. Yang, D.B. Minturn, F. Hady, “When Poll is Better than Interrupt,” Proceedings of the USENIX Conference on File and Storage Technologies, Vol. 12, pp. 3-3, 2012.

[4] D.L. Moal., “I/O Latency Optimization with Polling,” Proceedings of Valut-Linux Storage and Filesystems Conference, 2012.

[5] A. Eisenman, D. Gardner, I.. AbdellRahman, J. Axboe, S. Dong, K. Hazelwood, C. Petersen, A. Cidon, S. Katti, “Reducing DRAM Footprint with NVM in Facebook”, Proceedings of the Europeam Conference on Computer Systems, 2018.

[6] J. Zhang, M. Kwon, D. Gouk, S. Koh, C. Lee, M. Alian, M. Chun, M.T. Kandemir, N. Kim, M. Jung, “FlashShare: Punching Through Server Storage Stack from Kernel to Firmware for Ultra-Low Lantecy SSDs”, Proceedings of the USENIX Symposium on Operating Systems Design and Implementation, pp. 477-492, 2018.

[7] Flexible I/O Tester, Avaliable on : https://github.com/axboe/fio.

[8] U. Kang, C.E. Tsourakakis, C. Faloutsos, “Pegasus: A Peta-Scale Graph Mining System”, Proceedings of IEEE International Conference on Data Mining, pp. 229-238, 2009.

[9] J. Dean, L. Barroso, “The Tail at Scale”, Journal of Communications of the ACM, Vol. 56, No. 2, pp. 74-80, 2013.

Myoungjun Chun (천 명 준)

He received the B.S. degree in Computer Science from Yonsei University in 2017. He is currently a Ph.D. Student in Computer Engineering at Seoul National University. His research interests include storage system and operating system.

Email: mjchun@davinci.snu.ac.kr

Yoona Kim (김 윤 아)

She received the B.S. degree in Computer Engineering from Handong University in 2018. She is currently a M.S. Student in Computer Engineering at Seoul National University. Her research interests include storage system and operating system.

Email: yoonakim@davinci.snu.ac.kr

Jihong Kim (김 지 홍)

He received the Ph.D. degree in Computer Science from University of Washington in 1995. He is currently a professor of Computer Engineering at Seoul National University, Seoul, Korea. His research interests include embedded system and NAND flash memory.

Email: jihong@davinci.snu.ac.kr