NVMeVirt 다중 인스턴스 지원 및 성능 모델 개선 연구

성민석 O1 이승윤 1 김상훈 1
1 아주대학교 소프트웨어학과
sms224523@ajou.ac.kr, soccer1356@ajou.ac.kr, sanghoonkim@ajou.ac.kr

Extending NVMeVirt for the Multiple Instance Feature and Improved Performance Models

Minsoek Sung O1 Seungyoon Lee 1 Sang Hoon Kim 1
1 Department of Software and Computer Engineering, Ajou University

요 약

이 논문은 NVMeVirt 소프트웨어를 개선하여 다중 인스턴스 기능을 추가하고, Zoned Namespace SSD의 모델링 오류를 수정하는데 초점을 맞추고 있다. NVMeVirt는 현재 단일 인스턴스만을 지원하고 있어 복잡한 환경을 모델링하는데 제약이 있다. 논문은 이러한 제한을 극복하기 위해 다중 인스턴스 지원을 통해 더 다양한 환경을 에뮬레이션할 수 있도록 기능을 확장하고자 한다. 또한, NVMeVirt의 ZNS(SSD) 모델링에서 발생하는 오류를 수정하여 실제 디바이스와 더 유사한 시뮬레이션 환경을 제공하고자 한다.

1. 서 론

NVMeVirt[1]는 NVMe 디바이스 가상화를 지원하는 소프트웨어로 설정에 따라 유저 스페이스에서 다양한 종류의 NVMe 디바이스를 정의할 수 있다.

현재 NVMeVirt는 단일 인스턴스에 대해서만 가상화를 지원하고 있다 이는 실제 복잡한 환경을 설계하는데 어려움을 준다. 이에 따라 NVMeVirt의 기능을 확장하여 다중 인스턴스를 가상화하고 관리할 수 있는 기능이 필요하다. 또한, NVMeVirt는 SSD의 read, write 등의 연산을 time latency를 통해 실제 디바이스처럼 모델링한다. 현재 ZNS SSD의 latency에 대한 모델링 오류가 있어 실제와 다른 현상이 나타난다.[2] 이에 따라 ZNS의 모델에 대한 수정이 필요하다.

따라서 본 논문에서는 NVMeVirt에 다중 인스턴스 기능을 추가하여 다양한 환경을 설정하고 에뮬레이션할 수 있도록 기능을 확장하고, ZNS의 모델에 대한 개선 방안을 제시할 것이다.

2. 관련 연구

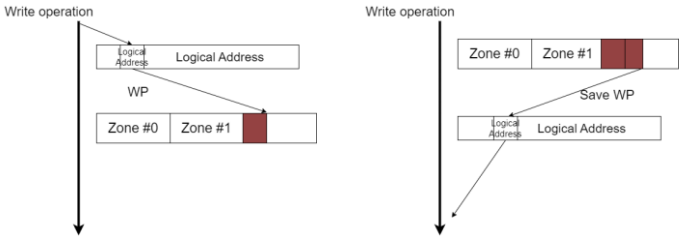
2.1 NVMeVirt

NVMeVirt는 가상 NVMe 디바이스와 호스트의 I/O 스택 사이의 다리 역할을 수행하며 일반적인 Conventional SSD, NVM SSD, Zoned namespace(ZNS) SSD및 Key-Value(KV) SSD와 같은 네 가지 타입의 NVMe 디바이스를 지원한다. 또한 PCI peer-to-peer DMA, NVMe-oF 등의 기능을 제공한다.

NVMeVirt는 PCI bus 라는 가상 PCI 장치를 통해 중앙 시스템에 연결된다. 가상 NVMe 디바이스(이하 NVMeV)는 할당된 메모리 공간에 PCI configuration header를 설정하고 이와 함께 가상 PCIe bus를 생성한다. PCI 서브시스템을 통해 해당 가상 PCI 장치를 PCIe 시스템에 연결한다.

실제 NVMe 디바이스는 control block에 직접 접근하여 요청을 전달한다. 그러나 NVMeV는 메모리에 control block이 존재하여 특별한 이벤트 없이 처리된다. Control block에 전달된 요청을 처리하기 위해 NVMeVirt는 커널 쓰레드인 dispatcher를 사용한다. Dispatcher는 control block의 값 변화를 감지하고 control request를 처리하며, 실제 I/O request 처리는 커널 쓰레드인 I/O worker를 통해 수행한다.

2.2 Zoned namespace(ZNS) SSD



[그림 1]ZNS 설명도

ZNS[3] 디바이스의 내부는 zone으로 나눠지고, 각 zone은 I/O 연산의 기본 단위인 block으로 나누어진다. 각 zone은 sequential write 명령만을 지원하며, zone 당 하나의 sequential write 명령만 수행하도록 제한을 두어 zone에 저장된 데이터의 연속성을 보장한다. 그러나 write pointer(WP)의 싱크가 맞지 않는 오류가 발생할 수 있다. 이를 해결하기 위해 ZNS SSD는 append 명령을 제안하였다. append는 WP를 명시적으로 알려야 하는 write와 달리 zone의 시작 LBA(ZSLBA)를 WP로 하여 데이터를 처리한다.

각 zone은 상태에 따라 할 수 있는 행동이 제한된다. implicit open, explicit open, close, empty, full이 있다. zone

의 상태는 open, close, finish 그리고 reset 명령어를 이용하여 명시적으로 관리할 수 있다. open 명령어는 zone의 상태를 explicit open 상태로 전환하고, close 명령어는 zone이 open 상태일 때 close 상태로 전환한다. finish 명령어는 해당 zone을 full 상태로 전환하며, reset 명령어는 zone을 empty 상태로 전환한다.

3. NVMeVirt의 개선 요구사항

3.1 다중 인스턴스 기능 확장

현재 NVMeVirt는 단일 인스턴스에 대해서만 가상화를 지원하고 있어, 다수의 디바이스를 동시에 가상화 하거나 관리하는데 제한이 있다. 이로 인해 다수의 NVMe 디바이스를 동시에 테스트하거나 시뮬레이션 하는 환경을 제공하지 못하며, 대규모 클라우드 컴퓨팅 환경이나 데이터 센터와 같은 다수의 디바이스가 연결된 복잡한 상황을 모델링 하는데 어려움을 준다. 이에 따라 NVMeVirt의 기능을 확장하여 다중 인스턴스를 가상화하고 관리할 수 있는 기능이 필요하다.

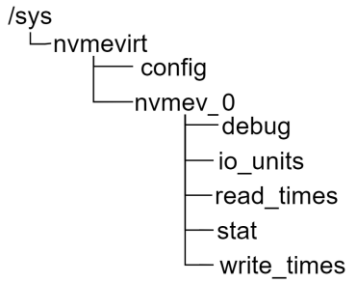
3.2 ZNS 모델 수정

현재 NVMeVirt는 write와 append가 같은 지연 모델을 사용하고 있다. 이는 zone 내부, zone과 zone 간의 확장성에 문제를 발생시킨다. 실제 ZNS 디바이스는 append의 latency가 write보다 길다. 또한 reset 연산의 latency가 NAND flash를 erase하는 latency와 동일하다. 이는 실제 디바이스에서 자주 발생하지 않는 상황으로 reset 연산에 대한 수정이 필요하다.

4. 다중 인스턴스 구현

4.1 유저 인터페이스 변경

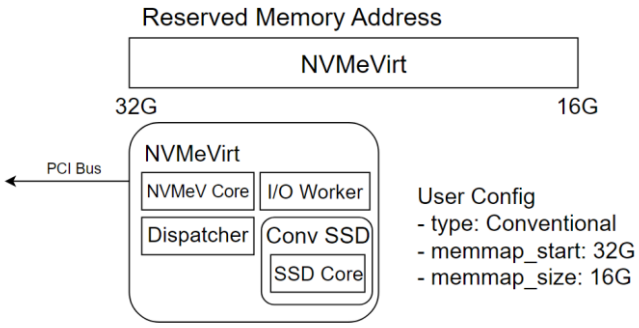
기존 프로그램은 procfs을 이용하여 사용자와 파라미터 전달 및 데이터를 유저에게 노출하였다. 그러나 프로그램이 다수의 가상 디바이스를 다루게 되면서 사용자가 각 디바이스를 체계적으로 관리할 수 있는 인터페이스를 제공하고자 sysfs으로 변경하였다. 사용자와 프로그램 간 커뮤니케이션은 NVMeVirt 모듈 로드 후 생성된 /sys/nvmevirt 디렉토리 아래의 config 파일로 진행된다. 사용자가 create 혹은 delete 명령어와 몇 가지 파라미터를 입력하면 사용자의 입력 값에 맞춰 가상 디바이스를 생성, 삭제한다.



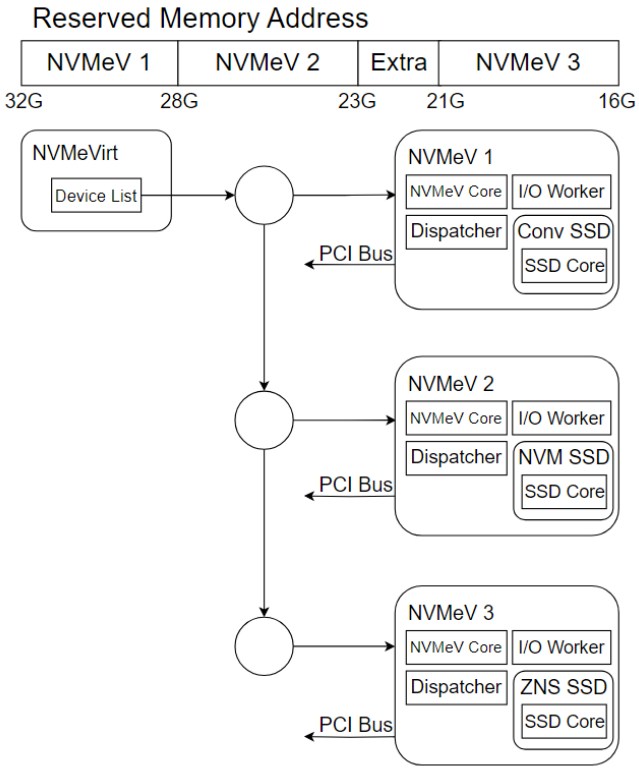
[그림 2] NVMeVirt의 트리 구조

4.2 static 리소스 분리

NVMeVirt는 단일 디바이스를 고려하여 개발된 프로그램으로 대부분의 리소스가 단일 디바이스에 최적화되어 있다. 모듈이 로드 후, 여러 가상 디바이스를 생성하기 위해서는 전역 변수로 선언된 변수들을 로컬 변수로 변경하고, static하게 할당하는 리소스를 각 디바이스별로 관리하고 모듈화하는 등 프로그램 구조를 전반적으로 수정하였다. 또한 NVMeVirt는 컴파일 전 사용자의 설정한 디바이스 타입에 따라 로드되는 코드가 다르다. 그러나 프로그램이 실행된 후에 사용자가 어떤 디바이스 타입을 선택할 지 알 수 없으므로 모든 코드를 로드 하여야 한다. 컴파일 당시 사용자의 설정을 처리 과정은 모듈 로드 후 가상 디바이스를 생성하는 과정에 추가하였다.



[그림 3] 기존 NVMeVirt 구조



[그림 4] 다중 인스턴스 구조

5. ZNS 모델링

5.1 Append latency 수정

현재 NVMeVirt의 append latency는 write latency와 동일하

다. 하지만 write operation은 append operation보다 낮은 latency를 가져야 한다. 따라서 append latency를 증가시켜주었다.

5.2 Reset latency 설계

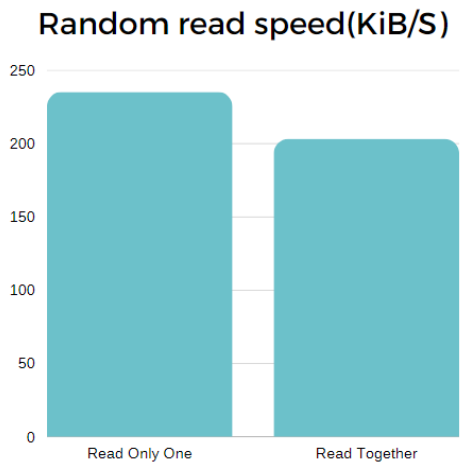
NVMeVirt의 reset latency는 현재 delay를 주지 않고 있다. 그로 인하여, Zone occupancy가 reset의 performance에 영향을 주지 못하고 있으며, read, write 그리고 append operations도 reset latency에 유의미한 영향을 주지 못하고 있다.

기존 NVMeVirt의 block erase에서, 한번의 erase operation latency와 무수히 많은 program latency를 통해 erase time이 정해졌기 때문에 erase operation latency가 erase time에 유의미한 영향을 주지 않았다. 따라서 erase operation latency가 설정되어 있지 않았다. 하지만, reset operation을 진행할 경우 erase연산을 많이 하게 된다. 이전과 달리 erase latency가 중요하게 되므로 samsung 980 pro erase latency를 참고하여 추가해주었다. 또한, reset의 실행 시간을 측정하여 이를 latency에 더해 줌으로 써 reset latency를 구현하였다.

6. 실행 결과

NVMeVirt의 개선 사항에 대한 결과 분석은 리눅스 5.15.37, x86-64 아키텍처에서 진행하였다. 테스트를 위해 2GB 메모리 영역을 reserve하였고 테스트 도구로는 fio를 사용하였다.

실행 테스트는 NVM, conventional 두 가지 타입의 가상 디바이스에 대해 fio를 이용하여 동시 random read 수행하여 두 디바이스가 독립적으로 I/O 연산을 처리하는지 확인하고자 한다.



[그림 5] fio 실행 결과

[그림 5]에서 Read Only one은 하나의 가상 disk에서 random read를 진행한 것이고, Read Together은 두개의 가상 disk에서 동시에 random read를 진행한 것이다. 위의 그림과 같이 하나의 가상 disk에서 진행하였을 때의 성능이 더욱 좋은 것을 알 수 있으나, 동시에 여러 개 random read를 실험하였을 때에도 성능 하락폭이 크지 않음을 알 수 있다.

	IOPS	Latency	BW
latency증가	2293	13924	287KiB
기존	2038	15651	255KiB

[표 1] latency 모델 변화에 따른 성능

위 표와 같이 reset latency 및 erase latency에 변화를 주었을 시 IOPS와 BW면에서 소폭 증가하였다.

7. 결론

본 연구는 NVMeVirt의 기능을 개선하여 다중 인스턴스 기능을 지원하고 ZNS SSD의 모델링 오류를 수정하여 실제 ZNS 디바이스와 유사한 에뮬레이션 환경을 제공하고자 하였다.

다중 인스턴스 기능을 구현하기 위해 유저 인터페이스를 변경하고 static하게 할당된 리소스를 모듈화 하였다. 또한 사용자의 설정에 따라 달라지는 커널 모듈로 로드되는 코드를 통합하였다. 그리고 NVMeVirt의 ZNS 모델 중 append와 reset 연산이 문제가 되었다. append 연산의 latency를 write 연산보다 크게 수정하고 reset 연산의 latency를 재설계함으로써 문제를 해결하였다.

이러한 개선 사항들을 통해 NVMeVirt를 이용하여 다양한 환경을 에뮬레이션하고 실제 환경과 더 가깝게 모사할 것으로 기대한다. 마지막으로, 256개의 instance를 생성하는데 성공하였으나 이론상으로 더 많은 instance를 생성할 수 있을 것으로 보인다.

사사의 글

"본 연구는 2023년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음"(2022-0-01077)

참고문헌

[1] S.-H Kim, J. Shim, E. Lee, S. Jeong, I. Kang, & J.-S Kim (2023). NVMeVirt: A Versatile Software-defined Virtual NVMe Device. In *21st USENIX Conference on File and Storage Technologies (FAST 23)* (pp. 379-394). USENIX Association.

[2] A. Krijn Doekemeijer, "Performance Characterization of NVMe Flash Devices with Zoned Namespaces (ZNS)," in *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, 2023.

[3] Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R. Ganger, & George Amvrosiadis (2021). ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)* (pp. 689-703). USENIX Association.