




연구논문/작품 제안서

2020 년도 제 1 학기

논문/작품	○논문(0) ○작품() ※ 해당란에 체크
제목	리눅스 커널에서의 I/O 완료 기법 연구
GitHub URL	https://github.com/Jeongcc/paper
팀원명단	한채정  (학번: 2016310249)

2020 년 3 월 25 일

지도교수 : 엄 영 익

서명



1. 과제의 필요성

1. 1 Abstract

최근 고성능 스토리지가 개발되어 등장하고 있는데, 이와 같은 고성능 스토리지는 HDD, SSD와 같은 기존 스토리지와 비교해보았을 때 더 짧은 응답시간을 제공한다. 하지만 기존의 운영체제는 고성능 스토리지가 아닌 기존 스토리지에 맞추어 설계되어 있다. 따라서 기존의 운영체제가 고성능 스토리지의 높은 I/O 성능을 충분히 활용하지 못 하는 문제점이 있다. 이로 인해 운영체제는 성능을 활용할 수 있는 I/O 완료 처리 기법 중 하나인 폴링 방식의 효용성에 대해 주목하고 있다. 본 논문에서는 리눅스 커널에서 고성능 스토리지의 높은 I/O 성능을 충분히 사용할 수 있는 방법을 과제로 연구하고자 한다.

1. 2 서론

I/O 완료 처리 방식은 응답 시간에 가장 크게 영향을 미치는 요소 중 하나로, 인터럽트 방식과 폴링 방식 등이 있다. 리눅스 커널에서는 I/O 요청을 처리하기 위하여 일반적으로 인터럽트 방식과 폴링 방식을 사용한다. 인터럽트 방식은 CPU가 한 프로세스에서 작업을 수행하는 도중에 인터럽트가 발생하면 하던 일을 중단하고 이미 정해진 코드를 실행해서 인터럽트의 요청을 처리하는 방식으로 동작한다. 그래서 프로세스에서 디바이스 드라이버로 I/O 요청을 보내고 Sleep 상태로 인터럽트 핸들러의 신호를 기다린다. 스토리지에서 I/O 처리가 끝난 후 인터럽트 핸들러가 사용자 프로세스로 신호를 보내고 I/O 요청이 끝난다. 인터럽트 방식은 스토리지에서 I/O를 처리하는 동안 다른 프로세스가 CPU를 사용할 수 있지만 I/O가 끝날 때에 Context Switch가 발생한다. 또한 인터럽트 방식의 CPU 사용량과 응답 시간을 살펴보면, CPU 사용량은 적지만 응답 시간이 느리다.

폴링 방식은 I/O 처리를 완료하였는지를 계속해서 확인한다. CPU가 각각의 주변 장치들을 연속적으로 순환하며 인터럽트 요구의 유무를 확인하므로 주변 장치의 상태를 보존할 필요가 없다. 그렇기 때문에 응답시간은 빠르지만, I/O 완료 여부를 확인하기 위해 프로세스가 계속 수행되므로 CPU 자원을 많이 소모한다는 단점이 있다. 또한, I/O 처리 완료 여부를 계속 확인하기 때문에 스토리지가 I/O 처리를 빨리할수록 폴링 기법은 더 높은 성능을 보이게 된다. 그래서 기존 운영체제가 고성능 스토리지의 높은 I/O 성능을 충분히 활용하지 못하고 있는 문제점의 해결방안으로 폴링 기법이 대두되고 있다.

그래서 인터럽트 방식과 폴링 방식을 대체할 수 있는 하이브리드 폴링이 제안되어 리눅스 커널에 이미 구현되어 있다. 하이브리드 폴링은 프로세스가 타이머를 사용해 일정 시간 동안 Sleep 한 후에 폴링한다. 폴링 방식과 비슷한 수준의 응답 시간을 보이면서 폴링 방식의 절반 정도의 CPU 이용률을 보인다. 하지만 모든 I/O 요청에 대해서 작동하지 않고, RWF_HIPRI 플래그가 설정되어 우선순위가 높은 경우에만 작동한다. 그래서 일반 동기 I/O 요청과 비동기 I/O 요청에 대해서는 처리

할 수 없다.

또한 리눅스 커널에서 일반적으로 사용되는 QEMU-KVM 하이퍼바이저는 2.9 버전부터 I/O 완료 처리 방식으로 인터럽트 방식을 기본으로, 적응형 폴링 방식을 옵션으로 제공하고 있다. 적응형 폴링 방식은 I/O 요청이 완료된 후에 해당 I/O 요청의 응답 시간을 기반으로 다음 폴링 시간을 조정한다. CPU 이용률이 폴링 방식과 비슷한 수준이기 때문에 비효율적이지만, 동기 I/O 요청뿐 아니라 비동기 I/O 요청에 대해서도 작동하기 때문에 일반적인 I/O 환경에도 적용할 수 있다는 점에서 하이브리드 방식과 다르다. 본 논문에서는 향상된 스토리지에 따른 리눅스 커널에서의 I/O 완료 처리 기법을 연구하고자 한다.

2. 선행연구 및 기술현황

2. 1 Y. Song and Y. I. Eom, "HyPI: Reducing CPU Comsumption of the I/O Completion Method in High-performance Storage Systems," Proc. of the Springer International Conference on Ubiquitous Information Management and Communication, pp.1-4, 2019

CPU 자원을 작게 소모하고, 좋은 퍼포먼스를 보이는 HyPI 방법을 제안한다. 폴링 기반의 I/O 완료 방법과 비교했을 때 퍼포먼스는 많은 차이가 나지 않지만 CPU 자원을 덜 소모하는 결과를 보인다.

2. 2 M. Chun and J. Kim, "An Adaptive Polling Selection Technique for Ultra-Low Latency Storage Systems", Korea Computer Congress, pp.1690-1692, 2018

전체 시스템의 CPU 이용률이 높은 상황에서 폴링 방식과 하이브리드 폴링 방식의 잦은 CPU 점유가 읽기 요청에 대한 꼬리 응답시간을 지연시키는 문제가 있을 밝히고, 전체 시스템 CPU 이용률을 관찰하여 개선한 적응형 폴링 선택 기법을 소개한다.

2. 3 H. LEE and Y. I. Eom, "I/O Completion Technique of Virtualized System Considering CPU Usage with High-Performance Storage Devices", Journal of KIISE, VOL 46 NO.07 pp. 0612-0619, 2019

기존의 I/O 완료 처리 기법에 대해 설명하고 새로운 I/O 완료 처리 기법의 필요성에 대해 이야기한다. QEMU-KVM 하이퍼바이저에서 제공하는 적응형 폴링 방식의 문제점을 설명한 뒤, 고성능 저장장치의 응답 시간을 최대한으로 활용하면서 CPU 사용량을 감소시키는 새로운 I/O 완료 처리 기법, HIPIV를 제안한다.

3. 작품/논문 전체 진행계획 및 구성

3. 1 진행계획

[표 1] 진행 계획

	3월	4월	5월	6월	7월
제출일정	지도교수님 선정 및 제안서, 서약 서 제출				
연구일정		리눅스 터널 공부 및 배경 지식 및 관련 연구 공부	배경지식 정리	실험환경 구축, 검증	실험 진행

	8월	9월	10월	11월	12월
제출일정		중간보고서 제출 (예상)		최종보고서 제출(예상),발표	
연구일정	실험 후 결과 정리	중간보고서 작성	논문작성 및 검토	논문제출, 발표준비	

3. 2 연구일정

3. 2. 1 리눅스 커널 및 I/O 시스템 공부

- 결과에 대한 정확한 분석을 한다.
- 제안한 개선방향이 실질적인 효과가 있도록 목표를 가지고 공부한다.
- 리눅스 커널에 구현된 I/O 시스템의 작동방식을 이해한다.
- 폴링과 하이브리드 폴링 방식을 분석한다.

3. 2. 2 선행 연구 탐색 및 분석

- 주제 설정을 위해 여러 논문들을 탐색하여 최신 동향과 기술을 분석한다.
- 실험에 반영할 수 있도록 정리한다.

3. 2. 3 연구 논문 주제 선정

- 기간 내에 수행 가능하도록 연구 논문 주제의 방향성을 설정한다.
- 기존에 연구된 부분에서 성능을 향상시킬 수 있는 부분에 대해 조사한다.

3. 2. 4 실험환경 구축 및 구현

- 리눅스를 기반으로 실험한다.
- 실험의 목적은 리눅스 커널에서 스토리지의 I/O 성능을 최대한 활용할 수 있도록 적절한 I/O 완료 기법을 연구한다.

3. 2. 5 결과 분석

- 기존의 I/O 완료 기법과 실험한 I/O 기법을 비교 분석한다.
- 보고서 형태로 정리한다.

3. 3 깃허브

깃허브에 매주 연구 수행을 기록한다. 4, 5, 6월에는 공부한 배경지식과 관련 논

문을 정리하고 교수님의 피드백을 받아 기록한다. 7, 8월에는 실험환경 구축과 실험 과정을 진행과정별로 기록한다. 또 실험결과를 포함한 실험 내용을 정리하여 개선방향, 향후 계획과 함께 기록한다. 9월에는 중간보고서를 올린다. 중간보고서에는 논문의 초안과 수정할 내용을 기록하고 관련 자료를 포함하도록 한다. 10, 11월에는 그 동안의 기록을 바탕으로 논문을 작성하고 발표 자료를 준비하여 올린다.

4. 기대효과 및 개선방향

학부 수준에서 현재 리눅스 커널 코드의 I/O 완료 기법을 완전히 대신할 만한 기법을 제시하는 일에는 어려움이 있을 것이다. 하지만 CPU 사용량을 줄이고, 고성능 스토리지의 빠른 응답 속도를 충분히 활용할 수 있는 방법을 연구하는 것은 잠재적인 가능성이 있다. 따라서 이 연구는 고성능 스토리지의 I/O 성능을 충분히 활용할 수 있는 I/O 완료 기법을 연구하고, 기존의 방식들과 비교하는 것에 의의를 가진다.

5. 참고문헌

- [1] Y. Song and Y. I. Eom, "HyPI: Reducing CPU Consumption of the I/O Completion Method in High-performance Storage Systems," Proc. of the Springer International Conference on Ubiquitous Information Management and Communication, pp.1-4, 2019
- [2] M. Chun and J. Kim, "An Adaptive Polling Selection Technique for Ultra-Low Latency Storage Systems", Korea Computer Congress, pp.1690-1692, 2018
- [3] D. L. Moal, "I/O Latency Optimization with Polling," Proc of the USENIX Conference of Linux Storage and Filesystems Conference, pp.1-25, 2017
- [4] D. I. Shin, Y. J. Yu, H. S. Kim, J. W. Choi, Y. Jung, and H. Y. Yoem, "Dynamic Interval Polling and Pipelined Post I/O Processing for Low-Latency Storage Class Memory," Proc. of the USENIX Workshop on Hot Topics in Storage and File Systems, pp. 1-5, 2018.
- [5] H. LEE and Y. I. Eom, "I/O Completion Technique of Virtualized System Considering CPU Usage with High-Performance Storage Devices", Journal of KIISE, VOL 46 NO.07 pp. 0612-0619, 2019
- [6] J. Yang, D. B. Minton, and F. Hady, "When Poll is Better than Interrupt," Proc. of the USENIX Conference of File and Storage Technologies, pp.1-7, 2012.