리눅스 기반 모바일 기기에서 사용자 응답성 향상을 위한 프레임워크 지원 페이지 보호 기법

김승준[1)O], 김정호[1)], 홍성수[1),2)]
[1)]서울대학교 융합과학기술대학원 융합과학부
[2)]서울대학교 전기정보공학부
{sjkim, jhkim, sshong}@redwood.snu.ac.kr

Framework-assisted Page Protection for Improving Interactivity of Linux Based Mobile Devices

Seungjune Kim[1)O], Jeongho Kim[1)], Seongsoo Hong[1),2)]
[1)]Department of Transdisciplinary Studies, GSCST, SNU
[2)]Department of Electrical and Computer Engineering, SNU

요 약

스마트폰과 같은 모바일 기기가 널리 보급됨에 따라 사용자들은 모바일 기기 응용들을 사용하면서 빠른 응답성을 제공받기를 바란다. 하지만 모바일 기기 응용들은 종종 사용자가 기대하는 수준의 응답성을 제공하지 못한다. 응답성을 저해하는 주 원인들 중 하나는 과도한 페이지 폴트 발생에 따른 대화형 태스크 수행의 지연이다. 이는 대화형 태스크의 상주 페이지(resident page)들이 비대화형 태스크와의 페이지 캐시 경쟁에 의해 더욱 빈번히 victim으로 선정되어 스토리지로 쫓겨나기 때문이다. 이 논문은 이러한 문제를 해결하기 위해 프레임워크 지원 페이지 보호 기법을 제시한다. 제안한 기법은 프레임워크 레벨에서 대화형 태스크를 식별하고 이를 커널에 전달하여 페이지 replacement 시에 대화형 태스크의 페이지를 보호하고, 사용자 입력 처리 중에 발생하는 페이지 폴트를 줄인다. 실험 결과 제안된 기법은 기존 시스템에 비해 페이지 폴트 횟수를 37% 감소시켰고, 응답시간을 9% 단축할 수 있었다.

1. 서 론

스마트폰 보급 됨에 따라 사용자들이 고사양, 고성능 응용들을 요구하고 있다[1]. 특히 이러한 응용들의 응답성은 사용자 경험에 있어서 중요한 요소들 중 하나이다. 스마트폰 제조사들은 모바일 기기의 사용자 응답성을 개선시키기 위해 많은 노력을 하고 있다. 하지만 이러한 노력에도 불구하고 여전히 양질의 응답성을 제공받지 못하는 경우가 보고되고 있다[2].

고사양 응용들은 대용량 컨텐츠들과 라이브러리를 사용하기 때문에 빈번하게 스토리지를 사용한다. Linux 커널에서 스토리지에 대한 접근은 요구 페이징(demand paging)에 의해 처리 된다. 응용이 요청한 페이지가 페이지 캐시에 부재한 경우, 페이지 폴트 핸들러가 페이지 폴트를 발생시켜 해당 페이지가 페이지 캐시에 적재된다. 특히, 메모리 제약이 심한 모바일 기기에서는 페이지 캐시의 크기가 데스크탑 환경에 비해 작기 때문에 페이지 폴트 발생 빈도가 높아진다. 실험을 통해 확인한 결과 사진 갤러리 응용은 background I/O가 있을 때, 최악의 경우 약 10000번 이상의 페이지 폴트가 발생한다. 따라서 사용자 응답성 개선에 있어서 페이지 캐시 관리를 최적화하여 페이지 폴트 발생 횟수를 줄이는 것은 중요하다.

페이지 캐시 적중률을 높이기 위한 페이지 관리 정책은 prefetching 정책과 replacement 정책으로 나뉜다. 리눅스에서는 preferching정책으로 readahead 매커니즘을 제공한다. 파일 페이지를 read할 경우 순차적으로 read하는 패턴이 있음을 가정하고 read한 파일 페이지 이후의 일부 파일 페이지를 순차적으로 미리 메모리에 적재함으로써 페이지 캐시 적중률을 높인다[3]. Replacement 정책은 기본적으로 LRU/2에 기반한다. 이를 위해 커널은 모든 프로세스의 working set을 포함하는 `active_list`와 reclaim candidates를 포함하는 `inactive_list`를 관리한다. Theodor etl.에 따르면 이 정책은 다른 LRU/k에 비해 다양한 응용에 대하여 5~10% 정도의 높은 페이지 캐시 적중률을 갖는다[4].

그러나 이러한 리눅스의 페이지 캐시 정책은 사용자 응답성 측면에 있어서 한계가 있다. 입력 처리 과정에서 사용자와 상호작용하는 대화형 태스크의 페이지들이 다른 페이지들과 식별되지 않기 때문에 페이지 캐시 자원을 유리하게 할당해줄 수 없기 때문이다. 결과적으로 사용자 입력을 처리하는 과정 도중에 대화형 프로세스의 페이지들이 다른 페이지들에 의해 페이지 아웃 당하고 페이지 폴트를 발생시키며 다시 적재 되는 긴 응답시간을 초래한다.

본 논문에서는 이 문제를 해결하기 위해 프레임워크 기반 응용 페이지 보호 기법을 제시한다. 제안된 기법은 대화형 태스크를 프레임워크 레벨에서 식별하고, 커널로 해당 정보를 전달한다. 그리고 커널은 프레임워크로부터 전달받은 대화형 태스크 정보를 이용하여 대화형

태스크의 페이지가 victim으로 선정되는 것을 방지한다. 이에 따라 대화형 태스크의 페이지 폴트 발생이 줄어들게 된다. 우리는 제안된 기법을 안드로이드 Kitkat 4.0 기반과 리눅스 커널 3.4가 탑재된 넥서스5S 스마트폰에 구현되었다. 실험결과 기존 시스템 대비 페이지 폴트 횟수를 37%, 응답 시간 9%를 단축시킴으로써 제안된 기법의 효용성을 입증하였다.

이 논문의 나머지 부분은 다음과 같이 구성된다. 2장은 사용자 응답성 및 문제를 정의한다. 3장에서는 제안된 해결책에 대한 설명을 한다. 그리고 4장에서 제안된 기법의 실험결과를 보이고 5장은 관련연구, 6장에서는 본 논문의 결론을 맺는다.

2. 사용자 응답성 및 문제 정의

본 논문에서는 사용자 응답 처리 과정에서 발생하는 페이지 폴트로 인한 응답 시간 지연을 해결하고자 한다. 사용자 응답성은 사용자 입력 이벤트가 모바일 기기에 발생한 시점부터 화면에 결과가 표시되기까지 걸리는 총 소요시간으로 정의한다[5]. 사용자 입력을 처리하기 위해 이벤트가 응용 태스크에 전달되고 응용 태스크가 해당 이벤트에 따라 작업을 수행한다. 그리고 그 결과를 화면을 출력하는 태스크에 전달하여 사용자에게 표시한다.

페이지 폴트가 발생하면 사용자 응답 시간이 지연된다. 그림 1은 사용자 입력 이벤트가 모바일 기기에 발생한 시점부터 결과가 출력되기까지의 태스크 체인과 그 과정에서 발생하는 페이지 폴트를 나타낸다. 사용자 응답성 관련 태스크들은 각자의 작업을 수행하면서 페이지를 사용을 위한 메모리공간을 요구하게 된다. 이 때 요청한 페이지가 메모리에 부재한 경우 페이지 폴트가 발생하면서 해당 페이지를 메모리에 적재하기 위한 디스크 I/O가 발생하면서 응답 시간이 지연된다.

본 논문에서는 페이지 폴트 횟수 감소를 통해 모바일

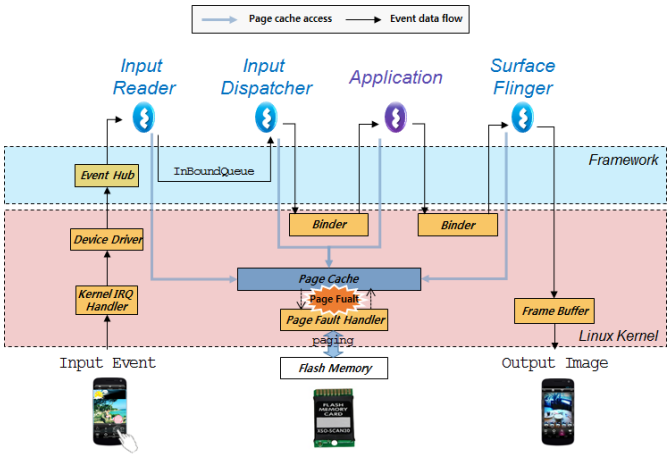

그림 1. 이벤트 전달 태스크 체인과 페이지 폴트 발생.

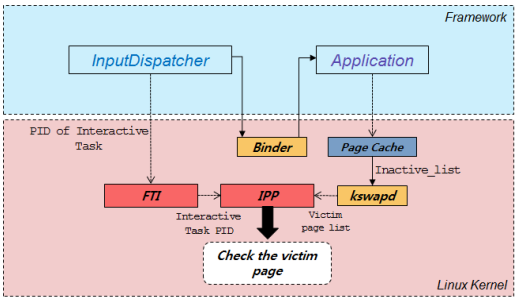

그림 2. 프레임워크 지원 페이지 보호 기법.

기기의 사용자 응답성 향상을 달성하고자 한다. 페이지 폴트 발생의 주된 원인은 기존 페이지 교체 기법이 대화형 태스크의 페이지와 다른 태스크의 페이지를 식별하지 않고 LRU/2 정책에 의해 페이지 reclaim victim을 선정하기 때문이다. 따라서 사용자 응답 관련 태스크들의 사용자 입력 처리 중 다른 태스크의 페이지에 의해 페이지 폴트가 발생하며 사용자 응답 시간을 지연시킨다.

3. 프레임워크 지원 페이지 보호 기법

본 장에서는 앞서 언급했던 대화형 태스크의 사용자 입력 처리 중 페이지 폴트 발생 횟수를 줄이기 위한 프레임워크 지원 페이지 보호 메커니즘을 설명한다. 이 메커니즘은 프레임워크의 지원을 받아 커널 레벨에서 대화형 태스크들을 식별하고 그들의 페이지들을 replacement 정책에서 보호한다. 이를 위해 이 메커니즘은 framework-assisted task identifier (FTI)와 interactive page protector (IPP)를 모듈로 구성된다.

FTI는 사용자 입력을 받은 대화형 응용 태스크의 정보를 프레임워크로부터 전달받아 커널에서 해당 응용 태스크를 식별한다. 구체적으로, 프레임워크 레벨에서는 InputDispatcher에 시스템 콜을 추가하여 사용자 입력을 처리하는 응용 태스크의 ID를 커널에 전달한다. 해당 시스템 콜은 프레임워크의 호출에 따라 인자로 넘겨받은 정보를 FTI에 기록한다. FTI는 이를 통해 커널 레벨에서 대화형 태스크를 식별한다.

IPP는 페이지 victim 선정 시 해당 페이지를 요청한 태스크를 식별하여 대화형 태스크의 페이지를 보호한다. 먼저, IPP는 메모리에 적재되는 페이지마다 해당 페이지를 요청한 태스크 ID를 페이지 구조체에 기록한다. 그리고 페이지 reclaim이 발생하여 victim 페이지를 선정하는 경우 프레임워크로부터 전달받은 대화형 태스크의 ID와 victim이 될 페이지에 기록된 태스크의 ID를 비교하여, 해당 페이지가 대화형 태스크인 경우 victim 선정에서 제외한다. 따라서 비대화형 태스크의 페이지들이 우선적으로 victim으로 선정이 되고 대화형 태스크의 페이지들은 다시 페이지 캐시의 리스트로 복구시킨다. 해당 페이지 캐시 리스트에서는 대화형 태스크의 페이지를 페이지 캐시 리스트의 head로 정렬시켜 기존 LRU정책에서 비대화형

태스크의 페이지가 대화형 태스크의 페이지보다 우선적으로 victim으로 선정되도록 한다.

4. 실험 평가

본 장에서는 제안된 기법을 검증하기 위한 실험과 그 결과를 보인다. 우리는 제안된 기법을 안드로이드 4.4와 리눅스 커널 3.4가 탑재된 구글 넥서스 5 스마트폰 상에서 구현하였다. 그리고 안드로이드에서 기본적으로 제공하는 사진 갤러리 응용의 응답시간과 사용자 응답 처리 중 발생하는 페이지 폴트 횟수를 측정하였다. 구체적으로, 갤러리 리스트에서 사진을 선택하여 화면에 출력되기까지의 페이지 폴트 횟수와 응답시간을 측정하였다. 응답시간 측정은 커널 레벨 프로파일링 도구인 kernel shark[6]를 사용하고, 페이지 폴트 횟수 측정은 `proc` filesystem[7]을 이용하여 측정하였다.

갤러리 응용 수행 시 I/O 집약적인 태스크를 동시에 실행시켰다. 페이지 reclaim이 빈번하게 발생하는 상황에서 응용의 페이지 폴트 횟수와 응답시간을 측정하였다. 결과는 그림 3과 같다. 실험 결과에서 알 수 있듯이 응용 페이지 보호에 따라 기존 시스템 대비 페이지 폴트 횟수는 약 37% 감소하였다. 그리고 페이지 폴트 횟수 감소에 따른 페이지 캐시 적중률의 증가로 응답 시간은 9% 감소하였다.

5. 관련 연구

모바일 환경에서의 사용자 응답성을 높이기 위한 기존 접근방법들은 커널 레벨과 프레임워크 레벨로 나눌 수 있다. 커널 레벨에서는 I/O로 인한 지연 시간을 줄이기 위해 많은 노력이 있었다. 대표적으로 모바일 환경에서의 I/O 특성을 파악하여 I/O scheduling를 최적화하는 기법이 있다. 응용 태스크와 asynchronous I/O의 dependency를 분석하고 응용 태스크와 dependable한 asynchronous I/O의 우선순위를 향상시켜 I/O scheduler에서 해당 I/O를 먼저 처리시킴으로써 I/O로 인한 응용 태스크의 수행이 지연되는 것을 방지한다[8].

프레임워크 레벨에서도 빠른 사용자 응답성을 제공하기 위해 다양한 기법을 사용하였다. 대표적으로 안드로이드는 응용태스크를 빠르게 실행시켜 사용자 응답성을 개선시켰다[9]. Zygote 프로세스라는 프로세스를 생성하여 응용이 자주 사용하는 라이브러리 클래스들을 미리 초기화 한다. 그리고 각 응용들이 생성될 때 미리 초기화된 Zygote 프로세스를 통해 생성됨으로써 응용이 생성되는 시간을 단축한다.

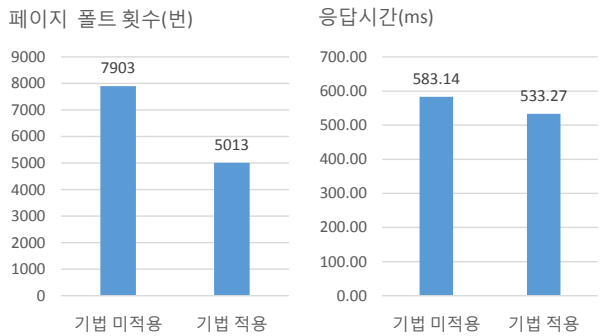

그림 3. 제안된 기법에 의한 페이지 폴트 횟수와 응답 시간의 감소.

6. 결론

이 논문은 모바일 환경에서의 사용자 응답성이 저조해지는 상황을 파악하고 이를 해결하기 위해 프레임워크 지원 페이지 보호 기법을 제안하였다. 이를 위해 먼저 사용자 응답성과 태스크 체인을 정의하고, 이 과정에서 페이지 폴트가 응답시간을 지연시킨다는 것을 밝혔다. 제안된 기법은 프레임워크에서 사용자 입력 처리 태스크를 식별하고 이를 커널로 전달하여 해당 대화형 페이지를 보호함으로써 사용자 입력 처리 중 페이지 폴트 발생을 최소화한다. 실험 결과 제안된 기법이 기존 시스템 대비 페이지 폴트 횟수는 약 37%, 응답시간은 9% 감소하였다.

Acknowledgement

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학 ICT연구센터육성 지원사업의 연구결과로 수행되었음(IITP-2015-(H8501-15-1015).

참고 문헌

[1] https://www.klick.com/health/news/blog/mhealth/mobile-apps-what-consumers-really-want/.

[2] http://bgr.com/2013/09/20/iphone-android-touch-screen-responsiveness/.

[3] W. Mauerer, “Professional Linux Kernel Architecture”.

[4] Johnson, Theodore, and Dennis Shasha. "X3: A Low Overhead High Performance Buffer Management Replacement Algorithm." (1994).

[5] S. Huh, J. Yoo, and S. Hong. "Cross-layer resource control and scheduling for improving interactivity in Android." Software: Practice and Experience (2014).

[6] http://rostedt.homelinux.com/kernelshark/.

[7] http://en.wikipedia.org/wiki/Procfs.

[8] D. Jeong, Y. Lee, and J. Kim. "Boosting quasi-asynchronous I/O for better responsiveness in mobile devices." Proceedings of the 13th USENIX Conference on File and Storage Technologies. USENIX Association, 2015.

[9] Bornstein, D. 'Google i/o 2008-dalvik virtual machine internals', 2008.