한국컴퓨터정보학회 동계학술대회 논문집 제21권 제1호 (2013. 1)

연쇄적 우선순위 상승 기법에 의한 안드로이드 스마트폰의 사용자 반응성 향상

이중현[○*], 허승주[**], 홍성수[* **]
[○*]서울대학교 전기컴퓨터공학부
[**]서울대학교 융합기술대학원 융합과학부 지능형융합시스템전공
e-mail:{jhlee85[○*], sjhuh[**], sshong[* **]}@filewood.snu.ac.kr

Improving Interactivity via Chained Priority Boosting for Android Smartphone

Joonghyun Lee[○*], Sungju Huh[*], Seongsoo Hong[* **]
[○*]Dept. of Electrical Engineering and Computer Science, Seoul National University
[**]Dept. of Intelligent Convergence Systems, Graduate School of Convergence Science and Technology, Seoul National University

● 요 약 ●

본 논문에서는 안드로이드의 고질적인 문제점인 사용자 반응성 문제 해결을 위한 연구를 소개한다. 특히 여러 응용들이 동시에 수행되는 경우 대화형 응용이 다른 응용들에 밀려 원하는 만큼 CPU를 얻지 못하는 상황에서 발생하는 반응지연 문제에 초점을 맞추고 이를 극복하기 위한 연쇄적 우선순위 상승 기법을 제시한다. 이 기법은 대화형 응용뿐만 아니라 기존 연구에서 고려하지 않은 터치 관련 이벤트 처리 스레드들과 대화형 응용의 자식 스레드들의 우선순위를 연쇄적으로 상향시킴으로써 터치에 대한 응답시간을 줄인다. 본 논문에서는 제안한 기법을 상용 스마트폰에 적용하여 유용성을 검증하였다. 실험 결과에 따르면 기존 안드로이드에 제안한 기법을 적용한 경우 평균반응시간이 기존의 31.91%로 감소하였다.

키워드: 안드로이드(Android), 사용자 반응성(User interactivity), 우선순위 상승(Priority Boosting)

I. 서 론

안드로이드는 멀티태스킹을 지원하는 리눅스 커널을 기반으로 설계되었다. 멀티태스킹은 사용자 편의성에 큰 기여를 하지만 상대적으로 제한된 컴퓨팅 자원을 가진 모바일 기기에서 큰 부담으로 작용한다. 특히 다수의 응용이 동시에 실행되는 경우에서 대화형 응용이 CPU를 원하는 만큼 할당받지 못하면 사용자가 느낄만한 반응지연이 발생할 수 있다.

이러한 반응지연 문제는 오래전부터 연구된 주제이며, 그 해결책으로 O(1) 스케줄러와 Multi-Level Feedback Queue 스케줄러가 있다[1]. 이들은 경험적인 기법을 통해 대화형 응용을 찾아내어 우선순위를 조절하여 반응지연 문제를 해소하였다. 하지만 이러한 방식은 일부 경우의 반응지연을 줄일 수 있는 반면, 경험적인 기법의 한계 상 반드시 대화형 응용을 찾는다는 보장이 없어 많은 부작용이 있었다.

안드로이드 스마트폰의 경우, 하나의 응용만이 화면을 점유하는 특징으로 인해 대화형 응용을 찾는 것이 용이해졌다. 이를 다룬 연구로는 [2, 3]이 있다. 이 논문들은 대화형 응용을 찾는 방식은 상이하지만 우선순위 상승을 통해 반응 지연을 해결하려는 것은 동일하다. 하지만 기존 방법들은 오직 단일 스레드로 구성된 대화형 응용만을 대상으로 삼고 있기 때문에 성능 향상에 한계가 존재한다. 특정 안드로이드 응용들은 복수의 스레드들로 구성되며 메인 스레드는 입출력만 담당하고 자식 스레드들이 전체적인 동작을 담당한다. 이러한 경우에, 기존 연구들은 효과적인 성능향상을 달성하기 어렵다.

본 논문에서는 멀티태스킹 환경에서 사용자 반응성 개선을 위해 응용의 메인 스레드뿐만 아니라 터치 관련 이벤트 처리를 담당하는 스레드들 그리고 응용이 추가적으로 생성하는 자식 스레드들의 우선순위를 연쇄적으로 상승시키는 기법을 소개한다. 그리고 제안한 기법을 상용 안드로이드 스마트폰에 적용하고 성능 향상 정도를 정량적으로 측정한다.

II. 연쇄적 우선순위 상승 기법

앞서 설명했듯, 연쇄적 우선순위 상승 기법은 사용자의 터치가 발생할 때마다 사용자 반응성과 관련 있는 모든 스레드들의 우선순위를 연쇄적으로 상승한다. 구체적으로 제안하는 기법은 아래의 그림과 같이 크게 (1) 관련 스레드 식별과 (2) 검출된 스레드의 우선순위 상향의 두 가지 단계로 동작한다.

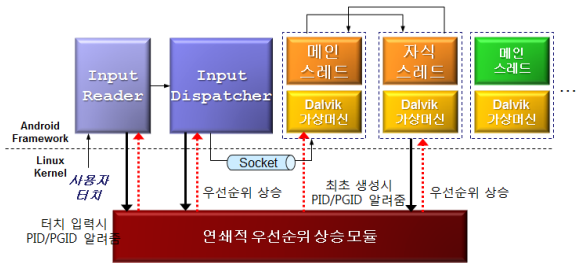

그림 1. 제안한 기법의 개괄

제안한 기법은 프레임워크 단에서 터치 관련 스레드를 식별하여 그 식별자를 커널 단으로 전달한다. 이를 위해, 터치 이벤트가 발생할 때 Input Reader가 자신과 Input Dispatcher의 식별자를 커널 내부의 변수에 등록한다. 뒤이어 Input Dispatcher는 대화형 응용과 통신할 때, 해당 메인 스레드의 식별자를 등록한다. 이후 대화형 응용의 메인 스레드가 자식 스레드를 생성하게 되면 시스템콜을 통해 커널에게 그 식별자를 알려준다.

터치 관련 스레드의 식별이 완료되면, 제안한 기법은 이들의 우선순위를 연쇄적으로 상승시킨다. 상승되는 우선순위의 정도는 하드웨어의 성능과 해당 응용의 특성에 따라 다르다. 따라서 본 기법에서는 응용이 CPU를 점유하는 시간에 비례하여 우선순위를 적응적으로 조절한다. 즉 스레드가 할당받은 타임 슬라이스 내에 작업을 마치지 못하면 추가적으로 우선순위를 상승시킨다. 또한 우선순위가 상승된 스레드가 다른 스레드에 과도한 영향을 끼치는 것을 막기 위해 최대 상승 가능한 우선순위를 제한한다.

마지막으로 제안한 기법은 사용자의 터치가 종료되었을 때 상승된 스레드들의 우선순위를 복원 시킨다. 이 때 자식 스레드의 우선순위는 복원 시키지 않는다. 자식 스레드가 남아있다는 것은 사용자 터치에 따른 처리가 더 필요하다는 것을 의미하기 때문이다. 대신, 자식 스레드들은 대화형 응용의 메인 스레드가 백그라운드로 전환될 때 우선순위를 하강시킨다.

III. 평가 및 결론

연쇄적 우선순위 부스트 기법을 평가하기 위해 이를 상용 스마트폰에 적용한 뒤 기존 안드로이드와 비교 실험을 수행하였다. 대상 시스템은 리눅스 커널 3.0.3 버전과 안드로이드 4.1 버전을 탑재하고 있는 Galaxy Nexus이다.

여러 응용이 동시에 실행되는 상황을 재현하기 위해 안드로이드 응용 중 이미지 편집기인 Avairy, 동영상 변환기인 FFmpeg media encoder, AVG antivirus 그리고 PI benchmark를 사용하였다.

본 실험에서는 위의 응용들이 멀티태스킹 되는 환경에서 Avairy이 특정 이미지에 대해 광원 조절 필터를 적용할 때의 응답시간을 25회 반복 측정하여 평균값을 얻었다. 결과는 아래의 표와 같다.

표 1. 제안한 기법과 기존 안드로이드의 성능 비교

실험결과	기존	제안한 기법 적용 후
평균(s)	12.872	4.108
표준편차	1.2026	0.6714
최대 지연시간(s)	16.2	5.1

결과에서 알 수 있듯, 제안한 기법은 기존에 비해 평균 반응시간이 31.91%로 감소하였다. 실험 결과 제안한 기법이 성공적으로 사용자 반응성을 개선함을 알 수 있다.

참고문헌

[1] L. A. Torrey, J. Coleman, and B. Miller, “A comparison of interactivity in the Linux 2.6 scheduler and an MLFQ scheduler” Software: Practice and Experience, Vol. 37, No. 4, pp. 347~364, 2007

[2] S.W. Bae, J.H. Kim, and Y.I. Eom, "Enhancing Interactivity in Mobile Operating Systems" Journal of KIISE: Computing Practices and Letters, Vol. 18, No. 7, pp. 533-537, July. 2012.

[3] S.J. Huh, J.H. Yoo, and S.S. Hong, “Improving Interactivity via VT-CFS and Framework-Assisted Task Characterization for Linux/Android Smartphones”, In Embedded and Real-Time Compting Systems and Applications(RTCSA), IEEE 18th International Conference on, pp. 31-40, August. 2012.