한국컴퓨터정보학회 동계학술대회 논문집 제21권 제1호 (2013. 1)

안드로이드 단말의 태스크 그룹에 따른 CPU 점유율 분석

김명선°, 임진택*, 박대동*
°*서울대학교 전기컴퓨터공학부
e-mail: {mskim°, jtlim*}@filewood.snu.ac.kr, ddpark@redwood.snu.ac.kr*

CPU Usage Analysis According to the Task Group in Android Mobile

Myungsun Kim°, Jintaek Lim*, Daedong Park*
°*School of Electrical Engineering and Computer Science, Seoul National University

● 요 약 ●

리눅스 기반 안드로이드 단말에서는 CFS(Completely Fair scheduler)가 사용되고 있다. 그리고 CFS는 태스크의 nice값 조절을 통해서 응용프로그램의 CPU 점유율을 제어할 수 있다. 하지만 안드로이드를 업그레이드할 때마다 수많은 태스크의 nice값을 적절하게 맞추는 일은 매우 어려운 일이다. 이러한 문제를 해결하기 위하여 안드로이드 단말은 리눅스의 cgroup(control group)을 사용하여 태스크들을 그룹으로 나눈다. 고성능과 빠른 응답 특성이 필요한 태스크들을 apps 그룹에 할당하여 높은 CPU 점유율을 보장하고, 그렇지 않은 태스크들을 background 그룹에 할당한다. 하지만 안드로이드의 버전이 업그레이드 되면서 각 그룹에 속한 태스크들에도 변화가 생긴다. 그 결과 동일하게 제작된 태스크들의 CPU 점유율이 달라지게 되고 예기치 못한 성능 저하가 발생할 수 있다. 본 연구에서는 안드로이드 버전 향상에 따라 동종 태스크들이 이전 버전에서보다 성능이 하락하는 현상의 원인을 파악하였다. 아울러 분석과 실험을 통하여 태스크의 nice 값보다 그룹 스케줄링 메커니즘이 어떻게 태스크의 CPU 점유율을 결정 짓는지 규명하였다.

키워드: 컨트롤그룹(cgroup), 안드로이드(android), 그룹 스케쥴링(group scheduling)

I. 서 론

최신의 안드로이드 단말은 다양한 자원을 가지고 있다. 멀티코어, GPU, 메모리, 네트워크 등이 이런 자원에 해당한다. 사용자를 위한 수많은 응용 프로그램들은 이러한 자원들을 동시에 활용하려고 서로 경합한다.[1][2] 따라서 안드로이드 프레임워크 또는 그 아래에 위치한 리눅스 커널에서 해당 자원을 적절히 태스크들에게 공유시키는 메커니즘이 필요하다. 이러한 자원들의 제어는 태스크별 자원 점유율을 조절함으로써 가능하다.[4]안드로이드 단말에서 자원 점유율 조절은 다음과 같이 수행된다. init.rc 파일을 사용하여 자원 점유율을 설정한다. 이 파일을 통하여 apps와 background 태스크 그룹이 자원을 사용할 수 있는 할당된 비율이 정의된다.여기에서 말하는 할당된 비율은 “share”라는 값으로 설정된다. 각 그룹의 share 값은 리눅스 커널에 전달된다. 그 후 리눅스 커널은 share값을 기반으로 cgroup과 subsystem을 통하여 태스크별 자원 점유율을 제어한다.[3] cgroup이란 어떤 자원에 대한 태스크들의 집합을 나타낸다. Subsystem이란 cgroup에 의하여 정의된 그룹 내부의 태스크들이 사용할 수 있는 자원들을 모듈 형태로 나타낸 것이다.이러한 자원 점유율 제어를 리눅스 커널이 제공하므로 안드로이드 프레임워크 개발자들은 손쉽게 응용 프로그램의 특성에 맞추어 단말의 성능을 예측하고 태스크들을 각 그룹에 할당할 수 있다. 하지만 안드로이드 버전이 Ginger Bread, Ice cream Sandwich, Jelly Bean과 같이 변화함에 따라 태스크를 cgorup에 할당하는 정책도 계속 변화하고 있다. 그리고 이러한 정책 변화로 태스크 그룹에 속한 태스크들이 이전 안드로이드 버전과 달라지게 된다. 결과적으로 어떤 특정 태스크의 자원 점유율이 이전 버전과 달라서 응용 프로그램의 성능이 예측과 다르거나 저하되는 현상이 나타날 수 있다.

우리는 그룹 스케쥴링에 필요한 자료구조와 동작방식을 분석하여 위와 같은 문제점의 원인을 찾아냈다. 또한, 안드로이드 버전이 업그레이드 되면서 속한 그룹이 달라진 태스크를 대상으로 실험하였다. 그 결과 동일한 nice값을 갖지만 속한 그룹의 변경으로 인해 태스크들의 스케쥴링 지연이 최대 9배까지 늘어 나 는 현상을 태스크 타이밍 도를 통하여 보였다. 스케쥴링 지연에 따라서 태스크의 CPU 점유율이 떨어짐을 확인하였고, 해당 태스크를 사용하는 응용 프로그램의 성능이 떨어지는 현상도 실험을 통하여 보였다.

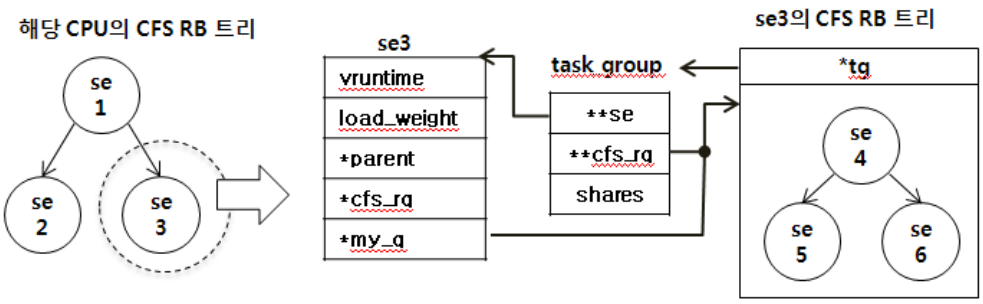

그림 9. se, task_group, CFS RB 트리의 그룹 스케줄링에서의 관계
Fig. 1. Relationship diagram for se, task_group and CFS RB tree in group scheduling

II. 안드로이드의 그룹 스케줄링 분석

1. 그룹 스케줄링에 필요한 자료구조 분석

본 절에서는 안드로이드 버전 상승에 따라 발생하는 태스크의 성능 저하 원인을 분석하기 위하여 안드로이드의 그룹 스케줄링을 설명한다. 안드로이드 프레임워크 하부에 위치한 리눅 스커널에서는 태스크 스케줄링을 위해 CFS(Completely Fair Scheduler)를 사용하고 있다. CFS는 nice 값으로 표현되는 태스크의 weight를 기반으로 CPU를 할당한다. 이 때, 대기 중인 태스크들을 위한 자료 구조로써 균형 이진 트리 중 하나인 RB(red black) 트리를 사용한다. RB 트리의 각 노드는 se(sched_entity)라는 자료구조로 구성된다. se는 한 태스크를 가리키는 task_struct나 여러 개의 태스크를 포함한 cgroup의 task_struct로 연결된다.그리고 각 se 가 가지고 있는 vruntime(virtual runtime)의 크기에 따라 RB 트리에서의 노드 위치가 결정된다. rb_lef most 포인터는 RB 트리에서 vruntime이 가장 작은 노드를 가리키는데, 이는 다음 스케쥴 대상이 되는 se이다. 그림 1은 그룹 스케줄링에 필요한 task_group, se, RB 트리의 관계를 보여준다. se3를 제외한 모든 se 들은 하나의 태스크를 가리키는 task_struct로 연결된다.[3] se3는 se4, se5, se6 로 이루어진 cgroup을 가리키는 task_group 포인터로 연결된다. 이때 task_group 포인터 중 cfs_rq는 se4, se5, se6가 형성하는 se3의 RB 트리를 가리킨다. 동시에 se3의 my_q는 se3 태스크 그룹이 소유한 런큐이므로 동일하게 se3의 CFS RB 트리를 가리킨다. se3의 parent 포인터는null값이다. 이는 se3가 태스크 그룹의 최상위 그룹임을 뜻한다. 그리고 se4, se5, se6의 parent 포인터는 모두 se3가 된다. 이러한 관계를 가지고 CFS는 vruntime을 기반으로 se1, se2, se3의 RB 트리에서의 위치를 결정한다. 얼마 후 시스템이 운용됨에 따라 vruntime이 변하고, se3가 해당 CPU CFS RB 트리의 rb_leftmost 포인터가 된다. 이때 se4, se5, se6 중에서 vruntime이 가장 작은 se가 스케줄링 된다.

2. vruntime기반 그룹 스케줄링 동작 분석

안드로이드는 버전에 따라서 태스크를 cgroup에 할당하는 정책이 다르다. Ice cream Sandwich에서는 ANDROID_PRIORITY_BACKGROUND(nice값 = 10)을 기준으로 하여 태스크들을 두 개의 cgroup으로 분류한다. 즉, 10보다 작은 nice값을 가진 태스크들은 foreground 그룹, 그 밖의 태스크들은 background 그룹으로 분류된다. Jelly Bean에서는 ANDROID_PRIORITY_BACKGROUND보다 nice값이 큰 태스크들은 background 그룹, -16보다 크고 ANDROID_ PRIORITY_BACKGROUND 보다 작은 태스크들인 경우 apps 그룹에 속한다. 그리고 -16보다 작거나 같은 태스크들은 부모 그룹을 상속받게 된다. 이렇게 정해진 cgroup들은 RB 트리에서 하나의 se로 표현되며, 매 스케줄링 이벤트 마다 각 se의 vruntime을 기반으로 RB 트리상의 위치가 결정된다. se의 vruntime은 다음과 같이 계산된다.

$$VR(t) = (W0 \times PR(t))/F(W) \quad \text{식 1.}$$

위 식에서 VR(t)은 vruntime을 나타내고 PR(t)는 se가 실제 수행된 시간을 나타낸다. W0는 nice값이 0일때의 weight이다. 그리고 F(W)는 해당 se가 태스크 그룹일 경우 그룹의 share 값의 함수이며, 그룹이 아니고 단일 태스크일 경우 해당 태스크의 weight 값이다. 그룹의 share 값은 init.rc에서 정해진다. Jelly Bean에서 예를 들면, apps 그룹의 share값은 1024이고 background그룹의 share값은 52이다. 식 1.에서 알 수 있듯이 vruntime은 share값에 반비례 한다. 따라서 동일한 시간만큼 CPU를 사용했을 때, background그룹은 apps그룹 보다 vruntime 값이 많이 증가하게 된다. 그 결과 background 그룹 vruntime이 클 가능성이 높고, 해당 CPU에서 CFS RB 트리의 rb_leftmost 노드에 위치할 확률이 작게 된다. 따라서 apps그룹의 태스크들이 더 큰 CPU 점유율을 나타내게 된다.[5][6]

III. 실험 및 검증

본 장에서는 같은 nice 값의 태스크가 안드로이드 버전 변화에 따라 다른 cgroup에 포함되었을 때 발생하는 CPU 점유율 변화를 실험을 통하여 보인다. 실험 환경을 표 1. 에 나타내었다. 실험 방법은 LCD가 꺼진 상태에서 음원 재생 프로그램의 AudioTrack 태스크가 동작 중일 때 하드웨어 key를 통해 LCD on 시키고, 이

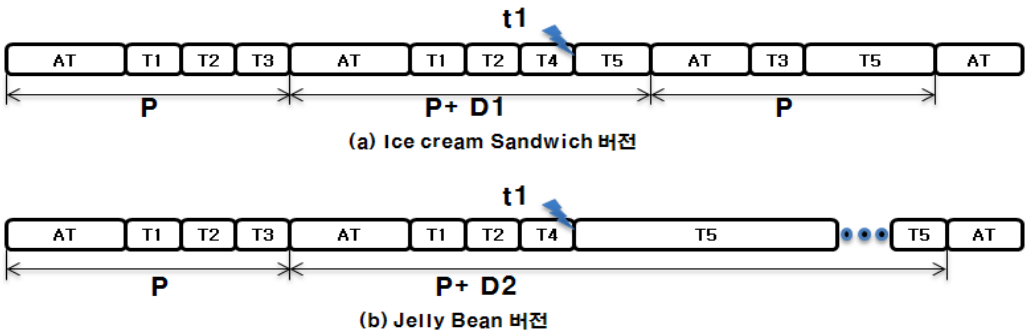

그림 10 AudioTrack 태스크의 스케줄링 주기 변화
Fig. 2. Scheduling period change for AudioTrack task

때 AudioTrack 태스크의 CPU 사용 상황을 측정하였다. AudioTrack 태스크는 안드로이드 네이티브 서비스인 Audio Flinger에 음원 데이터를 버퍼링 한 뒤에 전달하는 역할을 한다. 이 태스크는 Ice cream Sandwich 버전과 Jelly Bean 버전에서 동일한 nice 값 -16을 갖는다. 따라서 Ice cream Sandwich 버전에서는 foreground 태스크 그룹에 속하고, Jelly Bean 버전에서는 부모 프로세스인 media server로부터 그룹을 상속 받는다. 예를 들어, AudioTrack 태스크가 생성될 때, media server가 background 태스크 그룹에 속했다면, 이 태스크 역시 동일한 그룹에 속하게 된다. 그리고 apps 태스크 그룹일 경우도 마찬가지이다. AudioTrack 태스크가 apps 그룹을 상속받았을 경우, 이 태스크를 사용하는 응용 프로그램은 Ice cream Sandwich 버전과 동일한 성능을 나타낸다. 하지만 AudioTrack 태스크가 background 태스크를 상속 받았을 경우, 이 응용 프로그램의 성능은 저하된다.

표 1. 실험환경
Table 1. Test environment

	Jelly Bean	Ice cream Sandwich
안드로이드 버전	4.1.1	4.0.4
리눅스 커널 버전	3.0.31	3.0.15
응용프로그램	음원 재생 프로그램	
실험 대상 태스크	AudioTrack	

이런 관찰 결과는 그림2.에서 설명될 수 있다. 그림 2.는 AudioTrack 및 기타 태스크들이 안드로이드 Ice cream Sandwich 버전과 Jelly Bean 버전에서 각각 동작할 때 시간에 따른 CPU 점유 상황을 나타낸 것이다. AT는 AudioTrack 태스크를 나타내고, T1, T2, T3, T4는 background 그룹에 속한 태스크들이다. 음원 재생의 응용 프로그램이 수행되면 얼마 후 저전력을 위하여 LCD는 꺼지게 된다. 이때 AudioTrack 태스크는 P의 주기로 스케줄링 된다. t1 시점에서 사용자는 단말의 다른 응용 프로그램을 사용하기 위해서 키를 누르고 LCD는 켜진다. 이때 T5(론처 혹은 시스템 서버) 태스크가 다른 응용 프로그램의 사용 준비를 위하여 스케줄링 된다. Ice cream Sandwich 버전에서 T5는 foreground, Jelly Bean 버전에서는 apps 그룹에 속해있다. 그림 2.의 (a)는 AT가 foreground 태스크 그룹에 있고 share값이 큰 경우를 나타낸다. 따라서 식 1.에서 알 수 있듯이 해당 태스크 그룹은 vruntime이 작아서 CFS RB 트리의 rb_leftmost 노드로 갈 확률이 높아진다. 결과적으로 이 그룹에 속한 AT는 스케줄링 주기가 짧게 되고 이 태스크를 사용하는 응용 프로그램이 높은 성능을 나타내게 한다. 반대로 (b)의 경우 태스크 그룹이 background 태스크 그룹에 있고 share 값이 작다. 해당 태스크 그룹은 vruntime 증가분이 커서 스케줄링 주기가 길다. 스케줄링 주기가 길다는 의미는 그만큼 CPU 점유율이 낮음을 나타낸다. 따라서 AudioTrack 태스크를 사용하는 응용 프로그램의 성능은 떨어진다. 그림 2.의 D1과 D2는 AT 태스크의 스케줄링 지연을 나타낸다. 실험에서 D1 값은 최악의 경우 30ms 이고, D2 값은 최악의 경우 290ms였다. 최악의 D2 값을 나타낼 때 음원 재생의 응용 프로그램에서 끊김 현상이 발생하였다.

Ⅳ. 결 론

본 논문에서 안드로이드 단말의 그룹 스케줄링 기법을 분석하였다. 그리고 실험을 통하여 태스크 그룹의 변화에 따른 태스크별 CPU 점유율 변화를 확인하였다. 현재 Jelly Bean 안드로이드 버전은 apps와 background 그룹을 만든다. background그룹에 속한 태스크들의 전체 CPU 점유율이 5%이하로 제한되어 있다. 분석을 통하여 share값과 vruntime이 이러한 결과를 나타내는 이유임을 알았다. 또한 task_group, se, CFS RB 트리의 관계를 분석하여 share값이 그룹 스케줄링에 어떠한 영향을 미치는 지 파악하였다. 실험을 통하여 안드로이드 버전 변화에 따라 같은 nice 값의 태스크가 다른 태스크 그룹에 들어갔을 때 응용 프로그램의 성능 저하가 일어남을 보였다. 얻어진 실험 결과를 토대로 우리는 Jelly Bean버전에서 AudioTrack 태스크를 apps 그룹으로 옮겼다. 그 결과 AudioTrack 태스크를 사용하는 음원 재생 응용 프로그램에서 음 끊김 현상이 없게 하였다.

참고문헌

[1] H. Kim et al., "Aciom: Application Characteristics- aware Disk and Network I/O Management on Android Platform", EMSOFT 2011 pp. 49-58

[2] K. Salah et al., "Mitigating starvation of Linux CPU-bound processes in the presence of network I/O", The Journal of Systems and Software, 2012, Vol. 85, pp. 1899-1914

[3] /usr/src/linux/Documentation/cgroups/cgroups.txt
/usr/src/linux/Documentation/cgroups/cpuacct.txt
/usr/src/linux/Documentation/cgroups/devices.txt
/usr/src/linux/Documentation/cgroups/freezer-subsystem.txt
/usr/src/linux/Documentation/cgroups/memory.txt

[4] H. Kamezawa, "Cgroup and Memory Resource Controller", Japan Linux Symposium, 2008.

[5] http://doc.opensuse.org/documentation/html/openSUSE/opensuse-tuning/cha.tuning.cgroups.html

[6] https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html ~ ch03.html