응용 프로그램 패턴을 고려한 메모리 중복 제거 정책

*김도담, 홍경환, 신동군
성균관대학교 정보통신공학부
e-mail : rladls22@skku.edu, redcarottt@gmail.com, dongkun@skku.edu

Memory Deduplication Policy Considering Patterns of Mobile Applications

*Dodam Kim, Gyeonghwan Hong, Dongkun Shin
School of Information and Communication Engineering
Sungkyunkwan University

Abstract

Recent mobile devices such as smartphones and tablet computers require a large capacity of main memory to support more complex and versatile applications. This paper proposes the page sharing technique using KSM(Kernel Samepage Merging) for an efficient memory management in mobile devices. We evaluate the performance gain and overhead of KSM in a real Android device.

I. 서론

최근 스마트 폰과 태블릿 PC가 확산되면서, 모바일 응용 프로그램이 다양해지고 고도화되고 있다. 모바일 웹 브라우저는 HTML5나 WebGL을 수용함에 따라 더 복잡한 웹 페이지를 출력하고, 사진 뷰어와 동영상 플레이어는 풀 HD 급의 사진과 동영상을 보여주게 되었다. 이에 따라, 모바일 시스템의 메모리 용량은 더 많이 필요하게 되었다.

대부분의 모바일 기기 제조사들은 기기에 탑재되는 메모리의 용량을 늘려 메모리 사용량 증가에 대응하고 있다. 삼성전자는 갤럭시 시리즈의 메모리 용량을 2010년 출시된 갤럭시 S에서는 512MB, 2011년 출시된 갤럭시 S 2에서는 1GB, 2012년 출시된 갤럭시 S 3에서는 2GB로 매년 늘리고 있다.

메모리 반도체의 집적도에는 한계가 있기 때문에 용량을 늘리려면 더 많은 반도체를 부착할 수밖에 없다. 메모리 반도체를 더 부착하게 되면 전력 소모가 증가하고 회로의 크기가 커진다. 따라서 스마트 폰과 같은 휴대용 임베디드 시스템에서는 응용 프로그램의 수요에 따라 스마트 폰의 메모리 용량을 무작정 늘리기는 어렵고, 효율적인 메모리 관리 기법을 개발하여 메모리 사용량을 줄여야 한다.

현재 안드로이드 시스템은 LMK(Low Memory Killer)나 AMS(Activity Manager Service)를 통해 메모리 부족 시 프로세스를 제거하고, 자이고트(Zygote)를 통해 응용 프로그램 실행 초기에 발생하는 메모리 중복을 제거하여 메모리 부족을 해결하고 있다. 그러나 이 기법들은 응용 프로그램 실행 도중의 메모리 중복은 해결해주지 못하고 있다. KSM(Kernel Samepage Merging)이라는, 중복되는 물리 메모리 페이지를 실시간으로 제거하는 리눅스의 기능이 이러한 요구에 부합된다.

본래 KSM은 가상 머신 서버의 메모리 중복을 제거하기 위해 만들어진 기능이므로 임베디드 시스템에 본래의 설정을 그대로 적용할 수 없다. KSM은 여러 가지 매개변수를 조절하여 목적 시스템에 맞게 그 동작을 변경할 수 있기 때문에, 모바일 시스템에 가장 적합한 매개변수를 찾는 것이 중요하다.

본 논문에서는 KSM을 안드로이드 기반 모바일 기기 넥서스 10에 적용하였을 때 메모리 사용량 변화를 관찰하고, 이를 사용하는 응용 프로그램을 달리 하면

서도 분석하였다. 안드로이드에서 가장 효율적으로 동작할 수 있는 KSM 매개변수를 탐색하였다. 그리고 더 나아가 이러한 메모리 사용 효율의 개선이 사용자 반응 속도에 미치는 영향도 관찰·분석하였다.

II. 관련 연구

2.1 자이고트(Zygote)

자이고트는 안드로이드에서 각 프로세스 간에 같은 내용으로 존재하는 메모리를 공유하고, 새로운 프로세스의 생성 속도를 향상시키기 위한 프로세스다. 안드로이드는 응용 프로그램이 실행되어 새로운 프로세스가 생성될 때 자이고트 프로세스를 포크(fork)한다. 각 프로세스는 초기에 메모리의 구성 정보와 공유 라이브러리에 대한 링크 정보를 자이고트와 똑같이 가지며, 실행 중에 해당 메모리 영역이 쓰기가 수행되면 더 이상 메모리를 공유하지 않고 정보를 복사해서 사용한다.[2]

자이고트는 메모리 중복 제거라는 목적 면에서는 KSM과 같다. 그러나 자이고트는 프로세스 생성 초기에 발생하는 메모리 중복을, KSM은 현재 시스템 내부에 존재하는 메모리 중복을 제거한다는 점에서 서로 다르다.

2.2 안드로이드 메모리 중복

기존 연구[3]에서 안드로이드 환경에서 발생하는 메모리 중복이 나타나는 영역을 분석한 바가 있다.

안드로이드 환경에서 메모리 페이지는 표 1과 같이 네 종류의 영역으로 분류할 수 있다[3]. 발생하는 메모리 중복의 정적 분석 결과에 따르면, 안드로이드 환경에서 발생하는 메모리 중복의 발생에 두드러진 영향을 미치는 유일한 영역이 달빅 가상 머신이다[3].

그러나 이 연구는 KSM 도입의 타당성 분석에 불과하며, 실제로 모바일 기기에 적용했을 때 응용 프로그램 종류에 따른 영향까지 분석하지 않았다는 한계가 있다.

III. 배 경

3.1 안드로이드 메모리 사용량 관리 기법

LMK(Low Memory Killer)

LMK는 안드로이드의 메모리 사용량 관리 기법 중 하나다. 기존 리눅스의 메모리 관리 기법인 OOM 킬러(Out Of Memory Killer)는 사용 가능한 메모리 용량이 부족하다고 판단되는 경우, 가상 메모리를 가장 많이 할당 받은 프로세스를 제거하는 방식으로 동작한다. 그러나 OOM 킬러는 전화 프로세스, SMS 프로세스 등 스마트 폰 시스템 특성 상 우선순위가 높은 프로세스마저 메모리 부족 시 제거 될 위험이 있다는 단점을 가졌다. 따라서 안드로이드에서는 프로세스의 우선순위를 고려하는 독자적인 메모리 관리 기법인 LMK가 추가되었다.

안드로이드는 프로세스를 12 종류로 분류하고, 중요도에 따라 -16에서 15 사이의 우선순위를 부여한다. LMK는 사용 가능한 메모리 용량이 부족하다고 판단되는 경우, 이 우선순위를 참고하여 제거할 프로세스를 결정한다.

AMS(Activity Manager Service)

AMS는 안드로이드에서 실행되는 모든 액티비티(activity)와 서비스(service)를 관리한다. AMS는 액티비티 생애 주기를 관리하며, LMK에 이용되는 프로세스 우선순위를 조정하기도 한다. 또한 메모리 관리 관점에서 AMS는 프로세스의 최대 힙 메모리 크기를 제한하고, 히든 애플리케이션(hidden application)의 수가 일정 수준 이상 늘어나지 않도록 관리한다. 낮은 우선순위를 가진 히든 애플리케이션은 AMS에 의해 제거된다.

3.2 KSM(Kernel Samepage Merging)

KSM은 내용이 같은 물리 메모리 페이지를 하나로 합치는 리눅스 커널 기능이다. KSM을 사용하면 여러 가상 메모리 페이지가 같은 내용을 지닌 여러 개의 물리 메모리 페이지를 가리키는 것이 아니라, 하나의 물리 페이지를 공유하게 되기 때문에, 메모리 사용량이 줄어들게 된다.

KSM은 KSM 데몬(ksmd; KSM daemon)이라는 커널 프로세스가 상시 동작하면서 지속적으로 메모리 중

그룹	정의
커널	커널 코드와 데이터를 포함하는 페이지
달빅 가상 머신	자이고트의 페이지 혹은 자이고트에 의해 할당된 페이지
사용하지 않는 페이지	참조 카운터가 0인 모든 페이지
나머지	커널, 달빅 가상 머신, 사용하지 않는 페이지 중 어디에도 속하지 않는 페이지

표 1. 안드로이드 메모리 페이지의 분류[3]

복을 제거해나간다. KSM 데몬은 KSM 대상 페이지들을 탐색하며 내용을 확인하여 동일한 페이지 쌍을 찾아낸다. KSM은 이 작업을 위해 안정 트리(stable tree)와 불안정 트리(unstable tree)라고 하는 두 개의 트리를 이용한다. 안정 트리는 이미 KSM에 의해 공유되고 있는 페이지를, 불안정 트리는 KSM에 의해 인식되었으나 아직 공유되지는 않은 페이지를 포함한다.[4]

사용자 단계 프로그램에서는 'madvise' 시스템 콜을 통해 KSM 대상 메모리 공간을 직접 지정할 수 있다. 또한 KSM은 두 개의 매개변수, sleep_millisecs와 pages_to_scan을 제공하고 있어서, 이를 변경하여 KSM의 동작에 영향을 미칠 수 있다. pages_to_scan은 메모리 탐색을 시작했을 때 얼마나 많은 페이지를 탐색할지를, sleep_millisecs는 얼마나 자주 메모리를 탐색하는지를 결정한다[4].

KSM은 메모리 페이지를 탐색하는 과정에서 CPU 시간을 소모하여 다른 프로세스의 CPU 사용을 방해할 소지가 있다. 또한 KSM의 메타데이터가 메모리에서 차지하는 양도 무시할 수 없다. KSM 매개변수들을 조정할 때, CPU 시간과 메타데이터 메모리 사용량이라는 두 가지 오버헤드를 고려해야 한다.

Ⅳ. 실험 및 분석

4.1 실험 환경

본 논문의 실험에서 대상으로 한 안드로이드 기기는 넥서스 10이며, 운영체제는 리눅스 커널 3.4.5를 기반으로 한 안드로이드 4.2.2를 사용한다. 넥서스 10은 2GB의 메모리를 사용한다.

기존의 안드로이드 기기에 사용되는 리눅스 커널은 KSM을 사용하지 않으므로 KSM이 동작하도록 설정하였다. 그리고 달빅 힙 영역과 네이티브 스택 영역, 네이티브 힙 영역을 KSM 대상으로 지정하였다.

본 실험의 워크로드로 안드로이드 기본 웹 브라우저, 카메라, 갤러리, 유튜브(Youtube)를 이용하였다.

/proc/meminfo의 기록을 통해 전체적인 메모리 사용 상태를 확인할 수 있었다. 응용 프로그램의 불러오기 시간은 안드로이드의 시스템 이벤트 중 am_activity_launch_time 이벤트를 통하여 확인하였다. 짧은 클릭 이벤트의 반응 시간(response time)은 응용 프로그램 내부의 onClick() 함수의 수행 시간을, 긴 클릭 이벤트의 반응 시간은 onLongClick() 함수의 수행 시간을 통하여 확인하였다.

4.2 KSM 적용에 따른 메모리 사용량 변화

KSM이 메모리 성능에 미치는 영향을 확인하고, KSM 매개변수를 변경하여 안드로이드 환경에 적합한 값을 탐색하였다. 워크로드의 응용 프로그램을 순차적으로 실행하여 동시에 사용한 직후의 전체 메모리 사용 상태를 관찰하였다.

KSM 매개변수의 기본 값은 pages_to_scan이 100, sleep_millisecs가 20으로 지정되어 있다. 각 매개변수를 변경하면서 사용되지 않는 메모리(Free Memory)의 용량을 관찰하였다. 실험 시 변경하는 매개변수 이외의 매개변수는 기본 값으로 유지하였다.

그림 1은 매개변수 pages_to_scan의 변경에 따른 메모리 사용량의 변화를 나타내었다. pages_to_scan의 값이 커질수록 메모리 사용량이 절약된다. 그러나 pages_to_scan의 값이 80 이상으로 증가한 경우에는 더 이상 메모리 사용량이 절약되지 않는 포화 상태에 도달하였다. pages_to_scan의 값이 커지면 KSM 동작이 차지하는 CPU 시간이 늘어나므로 80이 최적 매개변수 값인 것으로 보인다.

그림 2는 매개변수 sleep_millisecs의 변경에 따른 메모리 사용량의 변화를 나타내었다. 충분한 시간이 경

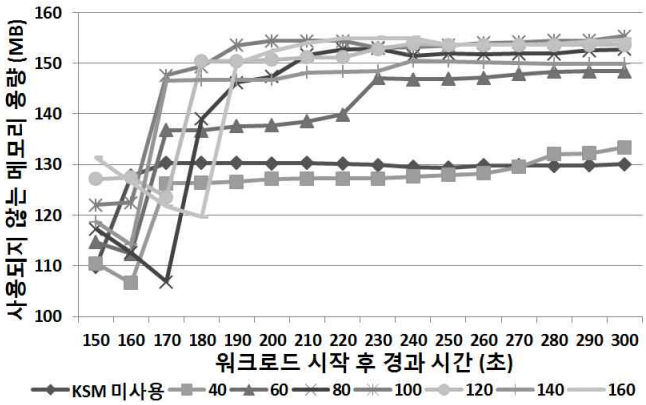

그림 1. pages_to_scan 변경에 따른 메모리 사용량 변화

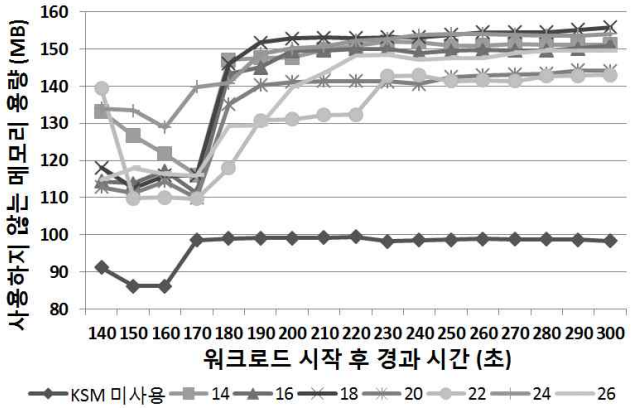

그림 2. sleep_millisecs 변경에 따른 메모리 사용량 변화

과한 경우 메모리 사용량은 거의 유사하다. sleep_millisecs는 KSM의 실행 주기를 결정하므로 메모리 사용량에는 영향을 미치지 않는다. 사용되지 않는 메모리 용량이 최댓값에 도달하는 시간은 sleep_millisecs가 20 이하일 때 일정한 양상을 보였다. KSM이 차지하는 CPU 시간을 최소화하기 위하여, 최적 sleep_millisecs 값으로 20을 선택할 수 있다.

4.3 KSM 적용에 따른 반응 시간 변화

KSM의 적용이 반응 시간에도 영향을 미치는지 관찰하였다. 워크로드를 수행한 뒤, 응용 프로그램들의 불러오기 시간과 터치 반응 시간을 KSM을 적용하지 않은 시스템과 적용한 시스템에 대해 측정하였다.

KSM은 CPU 시간을 소모하여 중복된 페이지를 탐색하고 제거하는 방식으로 동작한다. 그러므로 KSM을 적용하는 경우 메모리 사용량이 감소하는 대신 프로세스 실행에 소모되는 시간은 오히려 증가하는 trade-off 관계가 발생할 수 있다. 이러한 결과가 그림 4에서 발견되었다. KSM을 적용함에 따라 터치 이벤트에 대한 반응 시간이 증가하였다. 웹 브라우저는 17.9%, 갤러리는 130%, 유튜브는 87.1%의 터치 이벤트 반응 시간 증가를 기록하였다. 카메라의 경우 변화하지 않았다.

반면 응용 프로그램 불러오기 시간은 오히려 감소하는 경향을 나타내었다. 웹 브라우저의 불러오기 시간은 6.1% 증가하였으나, 카메라는 15.7%, 갤러리는 11.2%, 유튜브는 9.9% 감소하는 결과가 나타났다. 새로운 프로세스가 생성되는 동작은 대형 메모리 공간을 필요로 하므로 메모리 사용 상태의 영향을 받는 것으로 보인다.

V. 결론 및 향후 연구 방향

본 논문에서는 기존 안드로이드 플랫폼에서 채택하고 있는 메모리 관리 기법과 다른 방식인 KSM을 안드로이드 기기에 적용하는 실험을 제안하였다.

안드로이드 시스템에 KSM을 적용하였을 때, 메모리 사용량 측면에서 효율성이 증대되는 것을 관찰할 수 있었다. 적합한 KSM 매개변수를 탐색한 결과, pages_to_scan은 80, sleep_millisecs는 20을 적용할 때 가장 큰 효율을 얻으면서도 CPU 시간 소모를 최소화할 수 있다.

KSM의 CPU 시간 소모로 인하여 기기의 반응 시간은 증가하였으나, 응용 프로그램의 불러오기 시간은 오히려 감소하는 경향을 보였다.

프로세스를 제거하지 않고도 메모리를 효율적으로 이용할 수 있다면 더 많은 응용 프로그램이 동시에 메모리를 점유할 수 있으므로 안드로이드 기기의 성능 향상을 기대할 수 있다. 앞으로 KSM 동작과 모바일 디바이스 성능 사이의 관계를 명확히 규명하여 보다 모바일 환경에 적합한 메모리 관리 기법을 설계하고자 한다.

참고문헌

[1] P.Dubroy, “Memory Management for Android Apps”. Google I/O Developer Conference, 2011

[2] Ehringer, David. “The dalvik virtual machine architecture.″ Techn. report-March 2010, 2010

[3] Jonas Julino, “Analysing Page Duplication on Android”, Karlsruher Institut fur Technologie, 2012

[4] Andrea Ancangeli, “Increasing memory density by using KSM”, Linux Symposium, 2009

[5] Dan Bornstein, “Dalvik VM Internals”, Google I/O Session, 2008

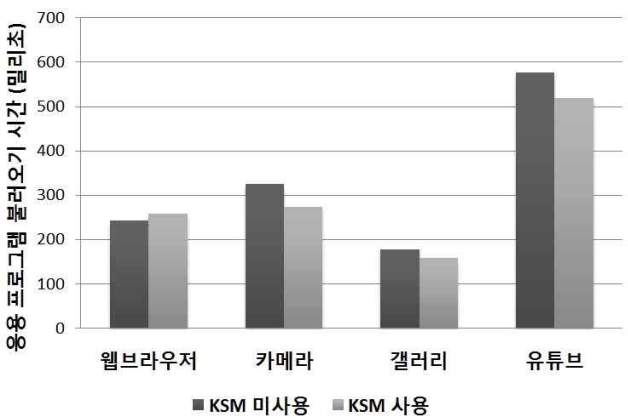

그림 3. KSM 적용에 따른 응용프로그램 불러오기 시간 변화

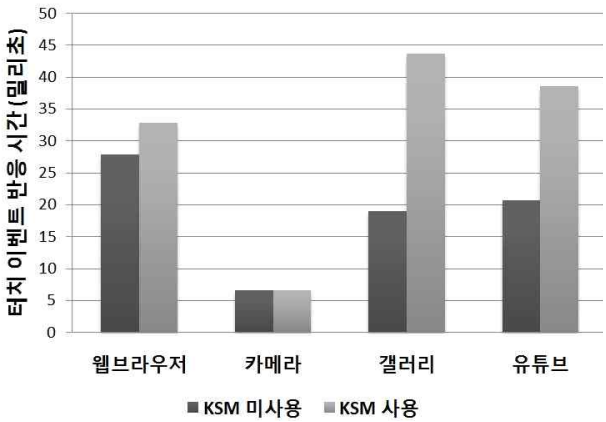

그림 4. KSM 적용에 따른 터치 이벤트 반응 시간