# 임베디드 시스템의 가상 머신 확장을 위한 메모리 압축의 필요성

이치영O 홍철호, 유혁
고려대학교 컴퓨터 전파통신공학과
{cylee O, chhong, hxy}@os.korea.ac.kr

# The Need of Memory Compression for Virtual Machine Scaliability in Embedded System

Chiyoung Lee O, Cheol-Ho Hong, Chuck Yoo
Department of Computer Science and Engineering, Korea University

**요 약**

임베디드 시스템 가상화는 PDA, 스마트 폰과 같은 장비에서 다양한 운영체제 및 응용프로그램이 동작하도록 컴퓨팅 자원에 대한 추상화를 제공한다. 반면 한정된 자원을 여러 가상 머신이 분할하여 사용함으로써 자원량의 제한이 더욱 심화된다. 특히, 메모리의 부족은 프로세스 실행에 반드시 필요한 자원으로 반드시 해결되어야 하는 문제이다. 본 논문은 메모리의 부족을 해결하기 위해 불필요한 메모리 공간에 대한 압축을 제안한다. 이는 가상화로 인한 메모리 분할과 프로세스의 메모리 상주 등의 이유로 인한 임베디드 시스템 가상화 환경에서의 메모리 부족을 해결할 수 있다. 본 논문은 이 메모리 압축 기법을 기술하고, 실제 가상화된 임베디드 시스템에서 경험한 메모리 부족 문제를 보인다. 이를 통해 메모리 절약 기법의 당위성을 증명하고, 향후 가상 머신 모니터에서의 메모리 압축 기법의 구현과 성능 평가의 기초를 다진다

## 1. 서 론

임베디드 시스템 가상화는 PDA, 스마트 폰과 같은 장비에서 다양한 운영체제 및 응용프로그램이 동작하도록 컴퓨팅 자원에 대한 추상화(abstraction)를 제공한다. 사용자는 이를 이용하여 폭넓은 응용프로그램 또는 서비스의 이용이 가능하다. 최근, 임베디드 시스템의 효율성을 극대화하기 위해 가상화 기법의 적용[1][2][3][4]이 늘고 있다. 시스템 가상화는 다양한 기능을 제공하면서 한정된 자원을 더 효율적으로 이용할 수 있도록 한다. 또한 가상화에 의한 고립성(isolation)을 제공하기 때문에 S/W 간의 간섭을 최소화할 수 있다[2]. 이런 장점으로 인해 임베디드 시스템에서의 가상화 연구가 점차 증가하고 있다.

그러나 한정된 자원에서 다양한 서비스를 제공하기 때문에 각각의 서비스에게 할당되는 자원의 양이 적어지는 단점이 있다. 이는 다양한 서비스 지원을 위해 필요한 가상 머신(virtual machine)의 확장성(scalability)에 큰 영향을 미친다. 메모리는 프로세스의 실행에 필수적인 자원이다. 메모리가 부족한 경우, 기존 운영체제는 페이지 스와핑(page swapping)나 사용하지 않는 프로세스의 강제 종료 등의 기법을 통해 해결할 수 있다.

특히, 임베디드 시스템은 하드디스크 대신 NAND 플래시 메모리(NAND flash memory)를 사용하기 때문에 페이지 스와핑이 용이하지 않다. NAND 플래시 메모리는 다시 쓰기(rewrite) 성능이 좋지 않고, 잦은 쓰기 작업으로 인해 플래시 메모리의 수명이 짧아지는 문제가 있다[5]. 그러므로 임베디드 시스템에서는 플래시 메모리에 대한 잦은 쓰기 작업을 발생시키는 페이지 스와핑 대신 프로세스의 강제 종료 기법을 주로 사용한다. 이 기법은 해당 프로세스를 참조하는 다른 프로세스들에게 영향을 미치기 때문에 예기치 않은 오류를 발생시킬 가능성이 있다. 임베디드 시스템은 데스크탑 환경에 비해 용량이 적은 메모리를 이용하기 때문에 메모리 부족으로 인한 영향이 더 크다.

이 문제를 해결하기 위한 기법으로 벌룬 드라이버(balloon driver)[6][7]가 있다. 이것은 Xen[8], VMware[9] 등의 가상 머신 모니터(virtual machine monitor)에서 사용하는 메모리 관리 기법이다. 벌룬 드라이버는 빈 메모리 공간들을 모아 가상 머신의 도메인에 관계없이 필요한 공간을 할당해 준다. 그러므로 이 기법에서는 빈 메모리 공간이 반드시 있어야 한다. 그러나 임베디드 시스템은 전체 메모리 크기가 작기 때문에 가상 머신의 수가 늘어나면 빈 메모리 공간을 확보할 수 없게 된다. 그러므로 메모리 공간을 최대한 절약할 수 있는 새로운 기법이 요구된다.

본 논문은 임베디드 시스템 가상화 환경에서 압축을 이용한 메모리 절약 기법을 설명하고 그 필요성을 밝힌다. 또한, xen-arm[3] 기반의 가상화 환경에서 메모리 공간의 압축의 가능성을 보인다. 이를 통해, 압축을 이

용한 메모리 절약 기법의 동기와 목적을 분명히 밝히고, 향후 기법의 구현과 성능 평가의 기초를 다진다.

본 논문의 구성은 다음과 같다. 2장에서는 메모리 공간 절약을 위한 연구들을 소개하고, 3장은 임베디드 가상화 환경에서 압축을 이용한 메모리 공간 절약 기법을 제안하고 그 필요성을 설명한다. 4장에서는 xen-arm 기반의 가상화 환경을 구축하고, 구축된 환경에서 메모리 압축이 가능한가를 보인다. 마지막으로 5장에서는 향후 연구 진행 방향에 대해 논한다.

## 2. 기존의 메모리 공간 절약을 위한 기법

메모리는 프로세스의 동작을 위해 반드시 필요한 하드웨어 자원이다. 대부분의 프로세스는 가상 메모리 공간을 가정하고 구현되기 때문에 실제 물리 메모리의 크기보다 큰 메모리를 요구한다. 데스크 탑 환경에서는 하드디스크를 이용한 페이지 스와핑을 통해 이를 해결한다. 그러나 임베디드 시스템에서는 하드디스크의 부재로 이 방법을 이용하기 어렵다. 또한, 가상화된 환경에서는 물리 메모리를 여러 게스트 운영체제들이 나누어 사용하기 때문에 사용가능 메모리 크기가 더 크게 줄어든다. 이 장에서는 메모리 부족 문제를 위한 기존의 해결책들을 소개한다.

Xen과 VMware는 메모리의 효율적인 사용을 위해 벌룬 드라이버[6][7]를 사용한다. 가상화된 시스템은 물리 메모리를 각각의 게스트 운영체제에게 분할하여 할당한다. 이 분할된 메모리 공간은 서로 다른 게스트 운영체제가 접근할 수 없다. 그러나 벌룬 드라이버는 각 게스트 운영체제가 가진 빈 메모리 공간을 모아 리스트(free list)로 유지한다. 게스트 운영체제에서 메모리 공간이 부족하게 되면 가상 머신 모니터에게 메모리 할당을 요구한다. 가상 머신 모니터는 벌룬 드라이버를 통해 빈 메모리 리스트에서 공간을 할당해 준다. 이를 통해 각 게스트 운영체제에게 유연한 메모리 할당이 가능하다. 하지만 적은 메모리 공간을 가진 임베디드 시스템에서는 극단적으로 모든 게스트 운영체제가 추가 메모리 할당을 요청하는 경우가 발생할 수 있다. 이 경우, 벌룬 드라이버가 관리할 빈 공간조차 남지 않아 메모리 부족 문제가 그대로 남게 된다. 이런 문제를 해결하기 위한 방법으로 메모리 공유 기법, 메모리 압축 기법 등을 사용할 수 있다.

Honeyfarm[10]은 가상화된 서버 환경에서 네트워크의 모니터링을 위한 시스템이다. 이 시스템을 구축하는데 메모리 공유 기법이 사용되었다. Honeyfarm은 수백개의 IP에 대해 각각 가상 머신을 맵핑하여 모니터링하기 때문에 매우 큰 확장성이 요구된다. 그러므로 메모리를 비롯한 자원의 절약이 매우 중요하다. 이를 해결하기 위해 쓰기 연산이 발생하지 않는 한, 가상 머신들이 같은 메모리 공간을 공유하도록 한다. 만약 메모리에 대한 쓰기 연산이 발생하면, Copy-On-Write(COW)을 이용하여 해당 공간을 분리시킨다. Honeyfarm에서 메모리 공유를 하는 목적은 IP를 가진 가상 머신을 최대한 많이 보유하는 것이기 때문에 모든 가상 머신은 동일한 종류이다. 그러나 임베디드 시스템 가상화의 목적은 다양한 서비스의 제공이므로 서로 다름 가상 머신이 존재해야 한다.

COW를 이용한 다른 메모리 공유 기법으로 서버 Xen 기반의 Content-based page sharing[11] 기법이 있다. 이는 메모리의 모든 페이지를 해싱(hashing)하여 같은 내용을 가진 페이지를 찾아 공유하는 기법이다. 모든 페이지를 해싱한 후, 해싱 값이 동일한 페이지에 대해 비트 비교(bitwise comparison)를 수행한다. 그 결과 내용이 동일한 경우, 두 페이지를 가리키는 모든 포인터를 하나의 페이지로 변경하고 남은 페이지를 제거하여 공유한다. 이를 통해 반환된 페이지의 크기만큼 메모리 공간을 절약할 수 있다. 이미 공유된 페이지에 대해 쓰기 연산이 발생하면, COW를 이용하여 페이지를 분리한다. 이 기법은 모든 페이지에 쓰기 권한(Write Protection)를 부여하여 쓰기 오류(Write Fault)를 발생시켜 COW 과정을 수행한다. 이것은 모든 메모리 쓰기 연산에 대해 페이지 오류 처리를 위한 오버헤드를 발생시킨다. 또한, 페이지 해싱 및 내용 비교 과정이 주기적으로 또는 페이지 할당/제거 때마다 발생해야 하는 오버헤드가 존재한다.

기존 메모리 압축 기법은 임베디드 시스템에서의 페이지 스와핑을 위해 사용되어 왔다. 메모리 압축 기법은 메모리 공유 기법들의 관리의 어려움을 보완할 수 있다. CRAMES[12], Compressed Swapping[5] 등의 기법이 이에 속한다. 두 기법 모두 디스크의 부재로 인한 페이지 스와핑의 어려움을 해결하기 위해 제안되었다.

CRAMES는 메인 메모리를 압축 영역(Compressed Area)과 비압축 영역(Uncompressed Area)으로 구분한다. 이들 영역은 다양한 크기의 블록들로 구성된다. 각각의 블록은 연속된 공간이지만, 압축 영역과 비압축 영역은 불연속적인 블록들의 리스트로 구성된다. 비압축 영역이 프로세스가 실행 중에 사용하는 페이지가 위치하는 공간이고, 압축 영역은 스왑 페이지(swapped page)가 저장되는 공간이다. 메모리와 디스크 사이에서 발생하는 페이지 스와핑을 메모리의 비압축 영역과 압축 영역 사이에서 발생시킨다. 이때, 압축과 복원으로 인한 오버헤드가 발생하게 된다. 이를 해결하기 위해 다양한 블록 기반 압축 알고리즘의 압축률과 메모리 요구량 등을 측정하여 가장 적은 오버헤드를 갖는 알고리즘(LZO algorithm)을 적용하였다.

Compressed Swapping은 NAND 플래시 메모리를 갖는 임베디드 시스템에서 페이지 스와핑을 수행할 수 있도록 하는 기법이다. CRAMES는 메인 메모리를 스왑 영역(swap area)으로 사용한 반면, 이 기법은 NAND 플래시 메모리를 스왑 영역으로 사용한다. NAND 플래시 메모리는 메모리 내용을 수정하는 성능이 좋지 않고, 쓰기

연산의 수행 횟수가 수명에 영향을 준다. 이로 인해 페이지 스와핑의 이용이 어려웠다. 그러나 Compressed Swapping은 스왑 페이지를 압축함으로써 플래시 메모리에 써야 하는 양을 줄였다.

두 기법 모두 압축을 이용한 페이지 스와핑으로 메모리 제약을 극복하고 있다. 그러나 이들은 모두 하나의 운영체제 커널에서 동작해야 하기 때문에 가상화된 시스템에서는 이용에 무리가 따른다. 모든 게스트 운영체제마다 해당 기법을 구현해야 하기 때문이다. 그러므로 임베디드 시스템 가상화 환경에 맞는 메모리 절약 기법이 필요하다.

## 3. 가상 머신 모니터의 페이지 압축을 이용한 메모리 관리

가상화 시스템은 가상 머신 모니터와 그 위에서 동작하는 가상 머신으로 구성된다. 가상 머신은 게스트 운영체제가 동작하기 위한 가상의 컴퓨팅 머신을 의미하고, 가상 머신 모니터는 이들 가상 머신의 동작에 필요한 자원을 제어 및 관리하기 위한 하위 계층을 의미한다. 대표적인 가상 머신 모니터로 Xen과 VMware가 있다. Xen은 게스트 운영체제의 수정이 요구(반가상화: para-virtualization)되는 반면, VMware는 게스트 운영체제의 수정 없이 가상화(전가상화: full-virtualization)된다. 반가상화는 비록 운영체제의 수정이 동반되지만, 가상화로 인한 성능 저하를 줄일 수 있는 장점이 있다. 임베디드 시스템은 데스크 탑이나 서버에 비해 처리 능력이 떨어지기 때문에, 본 논문은 성능 저하가 적은 Xen에 기반한 가상화 환경을 고려한다.

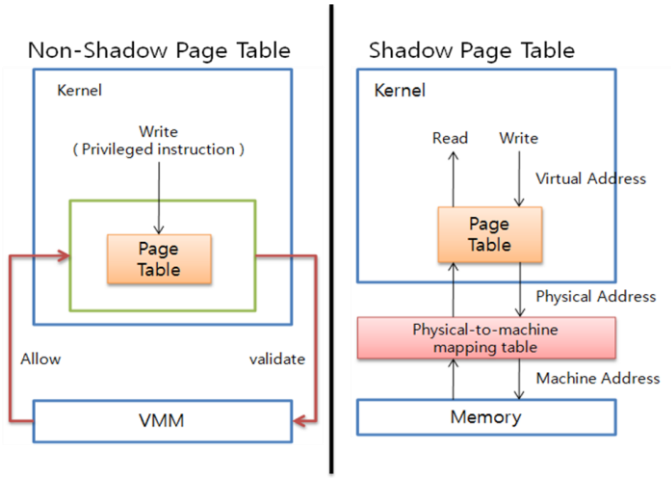


그림 1 Xen 기반 가상화 시스템의 페이지 테이블 구조

Xen은 그림 1과 같은 형태의 페이지 테이블 구조를 제공한다. 섀도우 페이지 테이블(Shadow page table)은 하드웨어 가상화(HVM: Hardware Virtual Machine) 지원에 의한 전가상화를 위해 사용되는 페이지 테이블 구조이다. 이 경우는 가상 머신의 메모리와 물리 머신의 메모리의 맵핑(Physical-to-Machine mapping)을 추가적으로 수행하여 게스트 운영체제의 수정 없이 메모리 접근을 제공한다. 그러나 임베디드 시스템에서는 아직 하드웨어 가상화를 제공하는 CPU의 보급이 많지 않다. 그러므로 본 논문에서는 섀도우 페이지 테이블을 이용하지 않는 반가상화를 위한 페이지 테이블 구조(Non-Shadow page table)를 이용한다. 이 경우, 각 게스트 운영체제는 페이지에 대한 접근을 위해 가상 머신 모니터에게 하이퍼 콜(Hyper call)을 통해 요청을 보낸다. 그러므로 게스트 운영체제는 하이퍼 콜을 통한 페이지 요청 메커니즘을 추가해야 한다. 요청을 받은 가상 머신 모니터는 해당 페이지에 대한 검증을 통해 허가 여부를 게스트 운영체제에게 전달한다. 즉, 페이지 접근 허가 요청으로 인해, 가상 머신 모니터는 물리 메모리 내의 모든 페이지에 대한 접근을 파악할 수 있다.

2장에서 설명한 바와 같이, 임베디드 시스템은 페이지 스와핑을 이용하는 경우가 드물다. 그러므로 한번 실행된 운영체제와 어플리케이션은 메인 메모리에 그대로 남아있게 된다. 즉, 운영체제가 부팅되는 순간 운영체제 전체가 메모리에 상주하고, 어플리케이션들이 실행되면서 남은 메모리 공간에 상주하게 된다. 많은 수의 어플리케이션이 실행된 경우 메모리 부족 문제가 발생한다. 또한, 임베디드 시스템은 전체 메모리 크기가 작으므로 적은 어플리케이션의 실행에도 문제가 발생할 수 있다. 게다가 가상화로 인해 여러 가상 머신이 메모리를 분할하여 사용하므로 문제가 더 심화된다. 적은 메모리를 가진 임베디드 시스템에서 메모리 공간을 효율적으로 이용하기 위해서 불필요한 메모리 공간 점유를 줄여야 한다. 본 논문은 사용하지 않는 메모리에 대한 압축을 통해 이를 달성하고자 한다.

페이지의 압축과 복원은 기존에 없던 처리 과정이 추가되는 것이므로, 복원 과정이 발생하지 않을 페이지를 압축하는 것이 가장 이상적이다. 그러므로 우선 메모리 내의 페이지 이용 패턴을 파악할 필요가 있다. 또한, 가상화 시스템은 여러 운영체제가 동작하기 때문에 이들을 모두 포괄할 수 있도록 가상 머신 모니터에서 페이지 압축 기법을 수행할 수 있어야 한다. 이를 위해 앞서 설명한 Xen의 페이지 테이블 구조를 이용한다. 섀도우 페이지 테이블을 이용하지 않는 구조에서는 모든 페이지 접근이 가상 머신 모니터의 간섭에 의해 이루어진다. 그러므로 가상 머신 모니터는 모든 페이지에 대한 접근 사실을 쉽게 파악할 수 있다. 가상 머신 모니터는 이 패턴 정보를 바탕으로 압축할 페이지를 선택하게 된다. 압축할 페이지의 선택은 다음의 기준을 따른다.

- 비사용 페이지 (Nouse page)
- 쓰기 연산이 없고, 읽기 연산이 적은 페이지 (Low-RO page)
- 읽기/쓰기 연산 모두 적은 페이지 (Low-RW page)
- 쓰기 연산이 없고, 읽기 연산이 많은 페이지

(High-RO page)
- 읽기/쓰기 연산이 모두 많은 페이지
(High-RW page)

페이지 압축/복원의 실행 시간은 읽기/쓰기 연산과 관계가 없다. 또한, 쓰기 연산이 메인 메모리에서 발생하기 때문에 쓰기로 인한 시간 지연보다는 페이지 압축/복원으로 인한 시간 지연이 더 크다. 그러므로 압축/복원의 횟수를 줄이는 것을 페이지에 대한 연산의 종류보다 우선해야 한다. 이들 페이지에 대한 압축을 통해 메모리 공간의 불필요한 점유를 줄일 수 있다.

## 4. 실제 임베디드 시스템에서의 메모리 부족 현상

이 장에서는 실제 임베디드 시스템에서 발생한 메모리 부족 현상을 보인다. 메모리 부족 현상이 실제로 발생함을 확인하고, 이로 인한 문제를 기술한다. 이를 위해 실제 임베디드 시스템에서 가상화 환경을 구축하고 두 개의 가상 머신을 동작시켰다.

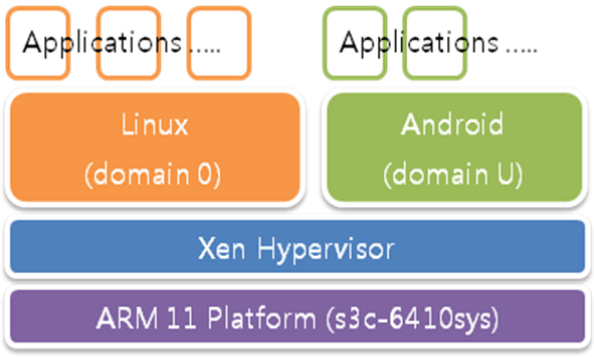


그림 2 실제 구현한 Xen-ARM 기반 가상화 시스템 구조

우리는 ARM11 기반의 CPU를 가진 s3c-6410sys 보드에 Xen-ARM[3]을 이용한 가상화 환경을 구축하였다. Xen-ARM은 Xen 기반의 가상화 플랫폼으로 임베디드 시스템 환경에 맞게 구현되었다. 이 Xen-ARM 가상 머신 모니터 위에 2개의 가상 머신을 동작시키고 게스트 운영체제로 반가상화된 리눅스 2.6.24와 반가상화된 안드로이드 1.1을 동작시켰다. 그림 2는 메모리 부족 현상을 확인하기 위해 임베디드 보드 상에 실제 구현한 Xen-ARM 기반 가상화 시스템의 구조를 나타낸다.

표 1 운영체제의 최소 메모리 요구량

| | 최소 메모리 요구량 | |
|---|---|---|
| | 비가상화 | 가상화 |
| 리눅스(non-GUI) | 8 MB | 22 MB |
| 안드로이드 | 96 MB | 104 MB |

실험 결과, 각각의 가상 머신에서 요구하는 메모리 양은 표 1과 같다. 이 표는 가상화하지 않은 상태에서 각 운영체제를 별도로 동작시킬 때와 가상화된 상태에서 동작시킬 때의 최소 메모리 요구량을 의미한다. 표에 나타난 바와 같이, 리눅스와 안드로이드는 각각 최소한 8MB, 96MB의 메모리를 필요로 한다. 이 두 운영체제를 가상화하게 되면 최소 메모리 요구량이 22MB와 104MB로 증가한다. 이 증가치는 가상화를 지원하기 위한 도구나 라이브러리들의 실행과 램 파일시스템으로 인해 발생한다. 리눅스는 관리 도메인으로 동작하면서 유저 도메인인 안드로이드를 실행시키기 위한 xen 도구들을 모듈로 실행한다. 반면, 안드로이드는 NAND 플래시 메모리의 파일 시스템을 이미 리눅스가 사용하고 있기 때문에 램 파일 시스템을 사용하게 된다. 이 때문에 약간의 메모리 증가가 필요하게 되었다. 실험 결과에 의하면, 최소 메모리 요구량은 118MB에서 126MB로 매우 크다.

실험한 환경에서는 128MB의 메모리를 사용했기 때문에, 최소 메모리 요구사항을 지키는 경우, 두 가상 머신의 동작이 가능하다. 그러나 실행하는 어플리케이션이 증가하게 되면 메모리 부족 현상이 발생한다. 이 경우를 재현하기 위해 두 가상 머신의 동작 중 안드로이드에서 계산기, 스네이크(snake) 게임, 메모장의 3가지 어플리케이션을 순차적으로 동작시켰다. 그 결과, 그림 3와 같은 프로세스 강제 종료가 실행되었다.

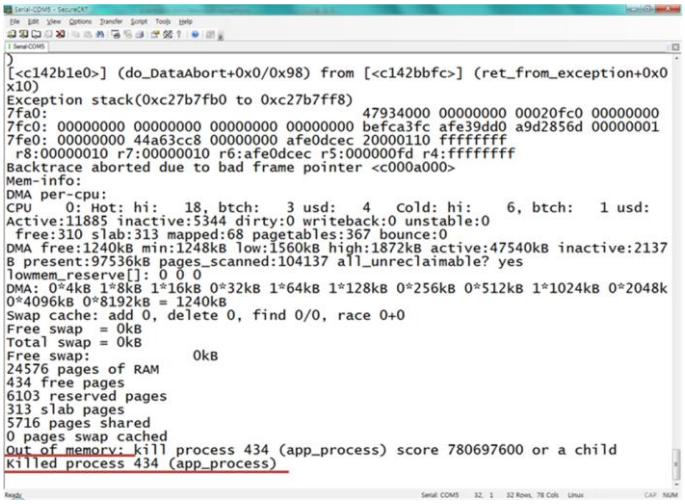


그림 3 Xen-ARM 기반 가상화 시스템의 프로세스 강제 종료

그림에 보이는 바와 같이, 메모리 부족으로 인해 “Out of memory” 오류가 발생하였고, 434번 프로세스인 app_process가 강제로 종료되었다. 운영체제의 동작을 위해 필요한 최소 메모리 요구량을 지켰음에도 불구하고, 어플리케이션의 동작에 의해 사용되는 메모리로 인해 메모리 부족 현상이 나타났다. 이는 메모리에 2MB의 여유 공간이 충분하지 못함을 의미한다. 그러므로 불필요한 메모리 공간의 압축을 통해 여유 공간을 확보해야 한다.

## 5. 결 론

지금까지 살펴본 바와 같이, 최근 증가 추세에 있는

임베디드 시스템 가상화는 메모리 부족으로 인한 문제가 잠재되어 있다. 이는 기반 운영체제와 관계없이 다양한 서비스를 제공하고자 하는 목적에 큰 걸림돌이다. 이를 해결하기 위해 메모리 공유나 압축을 통한 기법들이 연구되어 왔다. 그러나 공유 기법은 페이지 공유를 위해 추가적인 메모리 관리 작업이 많고 복잡하다. 반면 압축을 이용한 기법은 압축과 복원 과정 외에 추가되는 작업이 없어 관리가 용이하다. 하지만 압축과 복원으로 인한 오버헤드가 존재하고, 아직 가상화 환경에 적용된 기법이 거의 없다. 그러므로 가상화 환경에 맞는 페이지 압축 기법이 요구된다.

본 논문은 대표적인 임베디드 시스템 가상화 플랫폼인 Xen-ARM을 기반으로 페이지 테이블의 구조를 이용한 압축 기법을 제안하고 그 필요성을 보였다. 기존 Xen-ARM의 페이지 테이블 구조를 압축 기법에 이용함으로써 각 가상 머신에 별도의 수정이 필요하지 않다. 또한, 메모리 부족 문제를 실제 환경에서 경험함으로써 그 당위성을 보였다. 즉, 본 논문은 임베디드 시스템 가상화 환경에 맞는 페이지 압축 기법을 제안하고, 실제 환경에서 경험한 메모리 부족 현상의 관찰을 통해 그 필요성을 밝히는데 의의가 있다.

향후에는 제안한 압축 기법을 실제 구현하고, 이를 통한 메모리 공간의 확보 효과와 성능에 미치는 영향을 분석할 계획이다.

**참고문헌**


[1] Cheol-Ho Hong, Miri Park, Seehwan Yoo, Chuck Yoo, “Hypervisor Design Considering Network Performance for Multi-core CE Devices,” IEEE International Conference on Consumer Electronics (ICCE 2010), Jan. 2010.

[2] Seehwan Yoo, YoungPil Kim and Chuck Yoo, “Real-time Scheduling in a Virtualized CE Device,” IEEE International Conference on Consumer Electronics (ICCE 2010), Jan. 2010.

[3] J. Hwang, S. Suh, S. Heo, C. Park, J. Ryu, S. Park, and C. Kim, “Xen on ARM: System Virtualization Using Xen Hypervisor for ARM-Based Secure Mobile Phones,” Proceedings of the 5th Annual IEEE Consumer Communications & Networking Conference, USA, January 2008.

[4] HeiserGernot., “Hypervisors for Consumer Electronics,” Proceedings of the Consumer Communications and Networking Conference, 2009, pp. 1-5. Las Vegas, NV, IEEE.

[5] Sangduck Park, Hyunjin Lim, Hoseok Chang, and Wonyong Sung, “Compressed Swapping for NAND Flash Memory Based Embedded Systems,” LNCS - Embedded Computer Systems: Architectures, Modeling, and Simulation, Vol. 3553, pp. 314-323, July 2005.

[6] C. A. Waldspurger, “Memory resource management in VMware ESX server,” Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pp. 181-194, Dec. 2002.

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neuge-bauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” Proceedings of ACM Symposium on Operating Systems Principles(SOSP), Oct. 2003.

[8] Xen, http://www.xen.org/

[9] VMware, http://www.vmware.com/

[10] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, S. Savage, “Scalability, fidelity, and containment in the potemkin virtual honeyfarm,” ACM SIGOPS Operating Systems Review, vol. 39 n.5, December 2005.

[11] J. Kloster, J. Kristensen, and A. Mejlholm, “On the Feasibility of Memory Sharing: Content-Based Page Sharing in the Xen Virtual Machine Monitor,” Master's thesis, Department of Computer Science, Aalborg University, June 2006.

[12] L. Yang, R. P. Dick, H. Lekatsas, and S. Chakradhar, “Online Memory Compression for Embedded Systems,” ACM Transactions o Embedded Computing Systems, Vol. 9, No. 3, Article 27, February 2010.