ARM 기반 가상머신모니터 ViMo 상의 그림자 페이지 테이블 지연 동기화를 지원하기 위한 방법

전승협*, 안창원*, 이철훈**
*한국전자통신연구원
**충남대학교 컴퓨터공학과
e-mail : {shjeon00,ahn}@etri.re.kr,
{clee}@cnu.ac.kr

Implementation of supporting out of synchronization of shadow page table in ViMo hypervisor based on ARM

Seung-Hyub Jeon*, Chang-Won ahn*, Chul-Hun Lee*
*Electronics and Telecommunication Research Institute
**Dept. of Computer Engineering, Chung-Nam University

요 약

그림자 페이지테이블(shadow page table)은 MMU 를 가상화 함으로써 게스트 운영체제들이 하드웨어에서 제공하는 물리 메모리를 실제로 사용하는 것처럼 보이도록 하는 기술로 전가상화 지원 ARM 기반 가상머신 모니터인 ViMo 역시 게스트 운영체제간의 메모리 격리를 위해서 그림자 페이지 테이블을 사용한다. 본 논문에서는 그림자 페이지테이블의 성능을 향상시키기 위하여 ViMo 에서 사용하는 그림자테이블에 지연 동기화 기법을 추가하는 방법에 대해 설명하고 성능상의 이점을 보인다.

1. 서론

[1] 대부분의 운영체제들은 프로세서에서 제공하는 MMU 기능을 이용하여 가상 주소를 물리 주소로 변환하여 접근한다. 하지만, 완전 가상화 환경에서는, 가상머신 모니터는 MMU 장치를 가상화하여 수행되는 다수의 게스트 운영체제들로 하여금 직접 MMU 를 접근하는 것처럼 만들어 준다. 이런 메모리 가상화 기능을 제공하는 대표적인 방법이 그림자 페이지 테이블을 이용하는 것이다. 그림자 페이지 테이블은 게스트 운영체제들로 하여금 하드웨어에서 제공하는 물리 메모리를 실제로 사용하는 것처럼 여기도록 만들어준다. 이를 위해서는 게스트 운영체제의 페이지 테이블과 그와 연관되는 그림자 페이지 테이블은 동기화가 이루어져야 하며, 동기화에 따른 오버헤드가 존재하게 된다. 이 경우 게스트 페이지 테이블 매핑은 가상 주소와 의사 물리 주소와의 매핑을 가지게 되며, 실제 그림자 테이블 페이지 매핑은 가상 주소와 실제 물리 주소와의 매핑을 가지게 됨으로써, 게스트 운영체제는 하위의 메모리가 실제 물리 메모리인 것처럼 사용하게 된다.

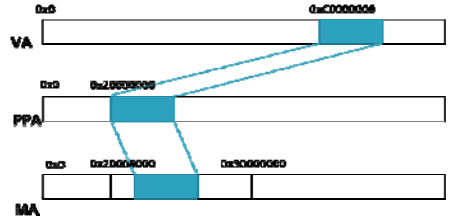
ARM 기반 가상머신 모니터인 ViMo[4]는 완전 가상화를 지원하는 가상머신 모니터로 ARM 메모리 구조에 기반하여 그림자 페이지 테이블을 지원하며, 그림자 페이지의 성능이 가상머신 모니터의 성능에 큰 영향을 미친다.

따라서, 따라서 본 논문에서는 ViMo 가 지원하는 그림자 페이지테이블의 구조에 대해 살펴보고 그 성능을 향상시키기 위하여 기존 XEN 등에서 구현되었던 out-of-sync 의 지연 동기화 방법을 ViMo 상에 구현 방법에 대해서 논한다. 그리고 실제 ViMo 에 out-of-sync 지연 동기화 기법을 구현 후 그 성능을 평가한다.

2. ViMo 그림자 페이지 테이블 구조

ARMv6 에서는 가상메모리를 1M, 64KB, 4K 페이지를 1 level, 2-level 페이지 테이블을 이용하여 memory management unit(MMU)를 통하여 가상주소(virtual address)-물리주소(physical address) 매핑 관계를 유지한다. 그러므로 가상머신 모니터는 다수의 게스트 운영체제간의 메모리 격리를 제공하기 위해서는 이런 MMU 를 게스트 운영체제에서 임의로 제어하지 못하도록 해야만 한다. 완전 가상화를 지원하는 ViMo 에서는 게스트 운영체제가 실제 MMU 를 사용 사용하는 것 보이도록 해주어야 한다.

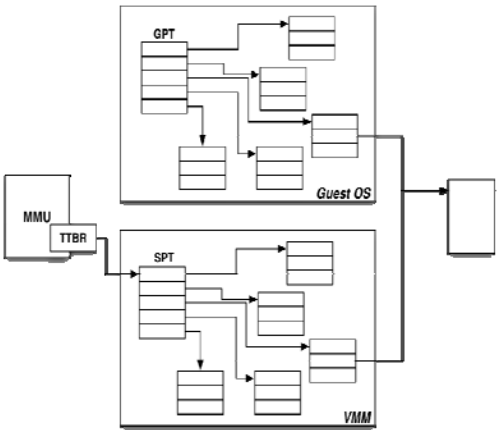
[1] * 본연구는 지식경제부의 IT 성장동력기술개발 사업의 일환으로 수행하였음.[KI002088, 공개 SW 기반 가상화 기술개발]



(그림 1) 가상메모리, 의사 물리메모리, 머신 메모리 관계

그림 1 처럼 VA-to-PA 관계 대신에 VA-to-MA(machine address) 관계를 가지도록 페이지테이블을 구성함으로써 게스트 운영체제는 자신이 MMU 를 사용하고 있는 것처럼 여기게 된다.

이와 같이 VA-to-MA 관계를 가지도록 만들어진 페이지테이블을 그림자 테이블(Shadow page table:SPT) 이라고 부른다.



(그림 2) Guest Page Table 과 shadow page table

ViMo 에서 그림자 페이지 테이블을 유지하기 위해서는 다음과 같은 방법을 사용한다. 게스트 운영체제에서 프로세서간 컨텍스트 스위칭시에 ViMo 는 Guest Page table(GPT) 과 매핑되는 SPT 을 생성한다. 그리고 기존 GPT 의 영역을 읽기전용 영역으로 만든다. 게스트 운영체제가 게스트 페이지 테이블 조작을 위한 연산을 수행할 경우 읽기 전용 영역에 쓰기를 시도했으므로 Data Fault 가 발생하고 ViMo 는 게스트 운영체제로부터 제어권을 넘겨받아 GPT 와 SPT 에 적절하게 반영한다. 이런 방법으로 GPT 와 SPT 는 항상 동기화 된다.

3. Out-of-sync 지연 동기화

GPT 와 SPT 를 항상 동기화 하는 방법은 매번 가상머신 모니터로의 모드 전환과 GPT 엔트리를 SPT 에 반영하기 위해서 매핑 관계를 계산하는 오버헤드가 존재한다. 이런 문제점을 해결하기 위해서 사용할 수 있는 방법이 out-of-sync 지연 동기화 기법이다.

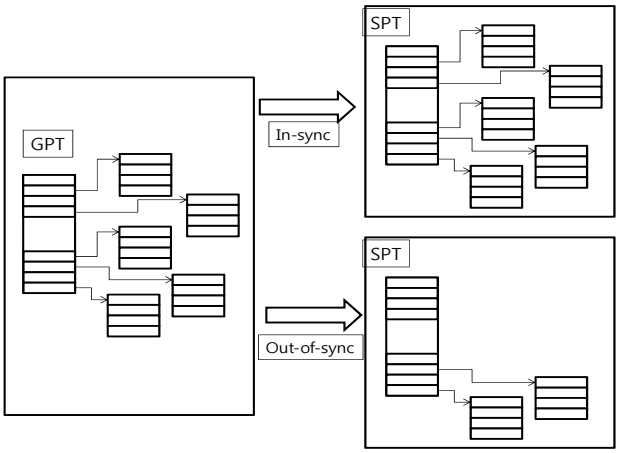
사용한 그림자 페이지테이블 out-of-sync 지연 동기화 기법은 다음과 같은 순서로 이루어진다.

① 게스트 페이지테이블의 쓰기 권한을 제거
② 게스트 운영체제가 페이지 테이블에 쓰려고 할 때 out-of-sync list 에 추가
③ 페이지 fault 가 발생할 경우 out-of-sync 리스트를 sync 한다.

②와 ③의 과정에서 Data Fault 의 타입을 분석해야 한다. ①의 과정에서 GPT 를 읽기 전용으로 만들었으므로 쓰기가 발생하는 경우 퍼미션에 관련된 abort 가 발생하지만, page fault 인 경우 translation fault 관련 abort 가 발생하기 때문에 적절히 구별해서 ②에 해당되면 GPT 가 쓰고자 하는 값을 GPT 에만 반영해주고, SPT 는 out-of-sync list 에 추가하고, Data Fault 가 발생한 다음 명령을 수행시키고 ③ 의 경우 out-of-sync list 의 모든 값들을 GPT 에 반영시킨 후에 관련 TLB 를 모드 flush 시킨 후에 Data Fault 가 발생한 명령어를 다시 수행시키면 된다.

out-of-sync 지연 동기화를 사용할 때 ViMo 는 두 가지 장점을 갖는다. 첫째, 매번 게스트 운영체제에서 페이지 테이블에 대한 연산을 수행할 때마다 동기화를 시키는 오버헤드를 줄일 수 있다. Out-of-sync 리스트는 게스트 운영체제에서 업데이트를 했지만, 사용을 시작하지 않은 메모리 영역이다. 그러므로 실제로 게스트 운용체제에서 사용할 때 까지 그림자 페이지로의 동기화를 지연시키고, 한꺼번에 다수의 그림자 페이지 테이블을 업데이트 시킴으로써, 성능상의 이점을 갖는다.

둘째, 프로세스간 문맥전환시에 발생하는 불필요한 GPT 와 SPT 간의 메모리 복사를 줄여 문맥전환 오버헤드를 줄일 수 있다. 문맥 전환시 그림 2 처럼 GPT 의 모든 내용을 SPT 에 반영해서 프로세스를 수행시키는 대신에 그림 3 처럼 실행에 필요한 일부분만을 동기화 시켜 프로세스를 수행 시킬 경우 GPT 와 SPT 의 모든 엔트리를 동기화 시키는데 오버헤드를 상당히 감소 시킬 수 있다.

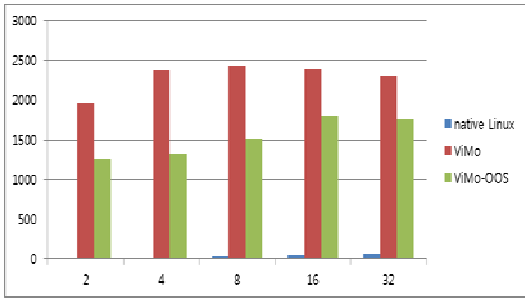


(그림 3) out-of-sync 를 이용한 문맥전환시 SPT

4. 실험

S5PC100 기반의 AchroHD 위에 구현된 ViMo 에 out-of sync 지연 동기화 기법을 적용 한 후 다음

lmbench (v3.0)의 lat_ctx(lat_ctx –s 32 24 6 8 16 32)를 이용하여 리눅스의 문맥 전환 시간을 비교한 것이다.



(그림 4) lat_ctx 결과

가로축은 32K 프로세스의 수를 나타내고 세로축은 문맥 전환 시간을 나타낸다. 실험 결과에서 알 수 있듯이 그림자 페이지를 통한 문맥전환의 경우 native Linux 보다 상당히 오랜 시간을 필요로 한다. 이것은 소프트웨어적으로 그림자 페이지 테이블을 구성할 때 성능상의 문제점이 발생할 수 있음을 의미한다. AMD 나 Intel 에서는 nested paging 이나 extended paging 과 같이 하드웨어적으로 메모리 가상화를 지원함으로써 성능 저하를 막는다. 하지만 현재 ARM 에서는 하드웨어적으로 메모리 가상화를 지원 하지 않으므로 성능상의 문제를

기존 ViMo 와 out-of sync 를 지원하는 경우 후자가 모든 경우에 30~40% 정도 성능상의 이점을 보인다. 이는 앞에서 설명했듯이 지연을 통한 동기화와 문맥 전환시 SPT 를 생성하는 오버헤드를 줄임으로써 만들어진 결과이다.

5. 결론

본 논문에서는 ARM 에서 전가상화를 지원하는 가상머신 모니터인 ViMo 의 그림자페이지 테이블의 성능을 향상시키기 위해 out-of-sync 지연 동기화 방법을 적용하는 살펴보았고, ViMo 에 구현하여 실험을 통하여 프로세스 문맥전환시 약 30~40% 성능 향상이 있음을 확인하였다.

하지만 여전히 기존 리눅스에 비하면 상당한 오버헤드가 존재하며, 이를 줄이기 위하여 그림자 페이지 캐쉬(cache)를 지원하고, ARMv6 에서 지원하는 TrustZone 을 이용하여 성능을 좀더 개선시킬 필요가 있다.

참고문헌

[1] “Understanding Full Virtualization, Paravirtualization, and hardware Assist”, White paper, pp4-5, VMware, 2007

[2] Joo-Young Hwang, Sang-Bum Suh, Sung-Kwan Heo, Chan-Ju Park, “Xen on ARM: System Virtualization using Xen Hypervisor for ARM-based Secure Mobile Phones”, IEEE 5th Consumer Communications and Networking Conference, pp. 257-261, 2008.

[3] Robert S. Phillips, “The Design of the XI shadow mechanism”

[4] Soo-cheol Oh, Kang-ho Kim, Kwang-Won Koh, Chang-won Ahn, “ViMo(Virtualization for Mobile): A Virtual Machine Monitor Supporting Full Virtualization For ARM Mobile systems”, Proc. Of Cloud Computing 2010

[5] 오수철,안창원 “임베디드 시스템 가상화를 위한 가상 인터럽트 콘트롤러의 구현 및 성능 분석”, 정보처리학회 2010

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

[7] AMD White Paper, “AMD-V Nested Paging”