# Indirection 기법을 이용한 경량 동적 코드 변환 기법의 설계 및 구현

김지홍, 김인혁, 엄영익
성균관대학교 정보통신공학부
{ jjilong, kkojiband, yieom }@ece.skku.ac.kr

# Design and Implementation of Light-weight Dynamic Binary Translation Scheme using Indirection Mechanism

Jeehong Kim, Inhyuk Kim, Young Ik Eom
School of Information and Communication Engineering, Sungkyunkwan Univ.

코드 변환 기법(binary translation)은 임의의 명령어 집합(instruction set architecture)을 다른 명령어 집합으로 변환하여 실행하기 위해 기계어 코드 수준에서 변환하는 기법이다. 최근 그린 IT, 클라우드 컴퓨팅 등이 새롭게 주목 받음에 따라 가상화와 동적 분석을 위한 코드 변환 기법이 활발히 연구되고 있다.

MyBT[1]는 테이블 기반 매핑 기법을 이용한 IA-32 아키텍처 간의 경량 동적 코드 변환 기법이다. 베이직 블록 단위로 코드를 변환하고, 변환된 코드의 분기 명령어가 실행된 후 다음 명령어가 가리키는 포인터의 주소 값을 변환하는 데에 리눅스 래딕스 트리(Linux radix trees)와 유사한 방식으로 빠르고 정확하게 동적 주소 변환을 수행한다. 변환 후 재정리된 명령어의 상대주소를 O(1)의 복잡도로 계산하여 다음 명령어가 실행될 수 있다. 본 논문에서는 MyBT를 기반으로 indirect 기법을 이용한 경량 동적 코드 변환기법을 제안한다.

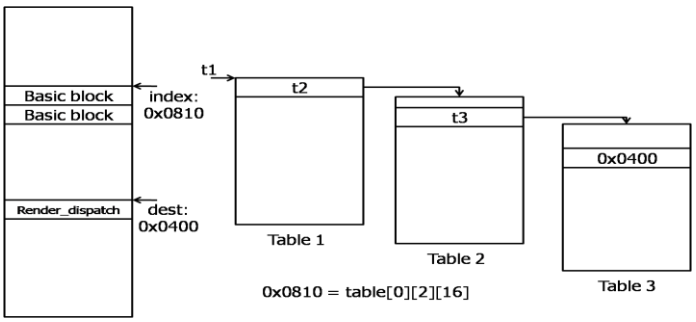


그림 1. MyBT의 동적 주소 변환

이에 따라 본 논문은 변환과정과 실행과정에서 많은 오버헤드를 발생시키는 특정함수들을 효율적으로 변환하기 위해 indirect 기법을 이용한 새로운 동적 코드 변환 기법을 설계하고, 이를 적용한 경량 동적 코드 변환 기법을 제안한다.

Indirect branch 뿐만 아니라 다른 명령어를 변환할 때의 변환과정의 오버헤드를 줄이기 위해 특정함수에 대한 동적 주소 변환 테이블을 미리 배치하는 Indirection 기법을 설계하였다. 이는 코드 변환을 준비하는 과정에서 먼저 코드를 추가해야 하므로 정적으로 할당해야 하는 메모리 크기가 커지는 단점이 있다. 하지만 기존의 동적 코드들이 갖는 변환 후 검증하는 과정에서의 실행 성능상의 오버헤드를 O(1)로 해결할 수 있다는 장점이 있다. 따라서 기존의 동적 코드 변환 기법들과 달리

본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음. (NIPA-2010-(C1090-1021-0008))

반복문을 많이 수행하는 명령어일수록 실행성능이 향상될 수 있다.

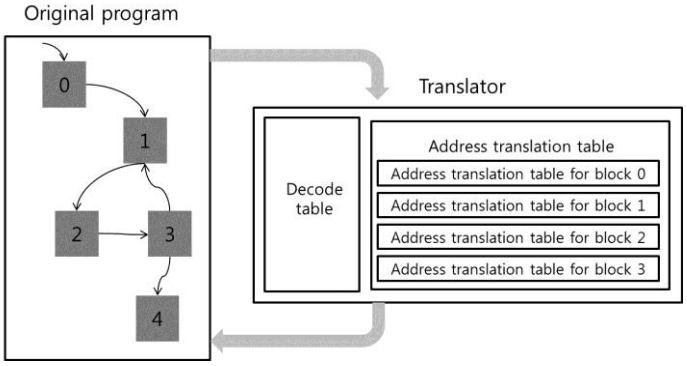


그림 3. Indirection 기법을 이용한 동적 코드 변환

기존의 동적 코드 변환기법 중 경량 동적 코드 변환기법을 목표로 하는 HDTrans, MyBT와 제안하는 Indirection 기법을 이용한 동적 코드 변환 기법의 성능을 비교하는 방식으로 평가하였다. 실험은 Intel(r) core(TM) 860 at 2.8GHz, 4GB RAM의 하드웨어 시스템 상의 Ubuntu 9.04, GCC version 4.3.3의 환경에서 실행하였다.

표 1. Indirection 기법을 적용한 함수

| 함수명 | |
|---|---|
| strlen | Fwrite |
| getenv | fprintf |
| strstr | memcpy |
| strcmp | vfprintf |
| malloc | fclose |
| fead | Printf |
| fopen | Free |
| fputc | funlockfile |

제안하는 Indirect 기법을 이용한 동적 코드 변환 기법은 분기문이 많은 명령어를 동적 코드 변환하여 실행하는 경우, 변환 시에 생기는 오버헤드와 늘어난 코드 길이에 따른 오버헤드 때문에 동적 코드 변환을 하지 않았을 때 보다 평균 1.6%의 오버헤드가 발생하였다. 하지만 기존의 기법들과의 비교를 통해 HDTrans와 약 0.5%, MyBT와 약 1% 성능 향상 결과를 확인할 수 있다. 이는 기존의 기법들이 다음 분기문에 대한 반환 주소를 검증하는 과정에서 추가 오버헤드가 발생한 반면에 제안한 Indirection 기법을 이용한 동적 코드 변환 기법은 이를 동적 주소 변환 테이블을 이용하여 O(1)의 복잡도로 반환했기 때문이다.

참고문헌

[1] 김지홍, 이동우, 김인혁, 엄영익,” 경량 동적 코드 변환 기법의 설계 및 구현” 한국컴퓨터종합학술대회, 2010. 6.

[2] Swaroop Sridhar, Jonathan S. Shapiro, Prashanth P. Bungale,”HDTrans: a low-overhead dynamic translator”, ACM Computer Architecture News, vol. 35, issue 1, pp. 135-140, March 2007