리눅스 환경에서의 함수 단위 동적 커널 업데이트 시스템의 설계와 구현

박현찬○ 김세원 유혁
고려대학교 컴퓨터학과
{hcpark, swkim, hxy}@os.korea.ac.kr

A dynamic kernel update system with a function granularity for Linux

Hyunchan Park○ Sewon Kim Chuck Yoo
Dept. of Computer Science and Engineering, Korea University

1. 서론

동적인 커널 업데이트는 현재 동작 중인 커널의 수행을 중단하지 않고 기능을 확장하는 방법이다. 이러한 기법은 커널 프로파일링, 커널 디버깅, 코드 패치의 설치, 그리고 코드 최적화 등을 동작 중인 커널에 적용하기 위해 사용될 수 있다[1]. 이렇게 커널의 기능에 동적인 확장성을 부여하고자 하는 연구는 크게 두 가지 방향으로 진행되고 있다. 첫 번째는 설계 단계에서부터 확장성을 고려한 새로운 커널을 만드는 방법으로, 대표적인 연구로는 SPIN[2], K42[3] 등의 연구가 있다. 두 번째 연구 방향은 기존 커널에 대해 확장성을 부여하는 연구로서, KernInst[1], GILK[4], SLIC[5] 등이 있다. 첫 번째 방법의 경우 완전히 새로운 커널 구조를 제안함으로써 문제를 단순하게 만들 수 있는 장점은 있지만, 널리 사용되는 기존 커널들과의 호환성 유지 문제로 인해 실제로 적용 가능한 해결 방법이 되기는 어렵다. 두 번째 방법은 기존 커널을 전혀 수정하지 않거나 약간의 수정을 통해 확장성을 제공한다. 하지만 사용자가 확장하고자 하는 코드를 어셈블리 언어로 작성해야 하거나[1][4], 시스템 콜이나 가상 파일 시스템(Virtual File System)처럼 인터페이스가 잘 정의(well-defined)되어 있어야만 확장이 가능한[5] 한계가 존재한다. 이러한 문제점들은 결국 현재 시스템에 확장성을 제공하는데 현실적인 어려움으로 작용한다.

우리는 이 논문에서 위에서 언급한 문제점들을 해결하기 위해 함수 단위로 커널을 업데이트할 수 있는 시스템을 설계하고 구현하였다. 이 시스템의 특징은 다음과 같다. 1) C 언어 레벨에서 기존의 함수 코드를 수정하고 그 내용을 동작 중인 커널에 반영할 수 있도록 하여 실제 업데이트 수행의 편의성을 높였다. 2) 트랩 기반 동적 재구성 기법[6](Trap-based dynamic instrumentation) 을 통해 함수 단위의 업데이트가 이루어진다. 이 기법은 하드웨어 환경에 독립적인 특성을 가지기 때문에 여러 환경에서 사용되는 리눅스에 적용하기 적합하다. 3) 이러한 주요 특징과 더불어 우리는 동적으로 수행된 업데이트를 시스템의 재기동 시에도 지속적으로 유지시키는 영속성(persistency)과 업데이트의 수행과 그 내용이 커널의 동작에 오류를 일으키지 않음을 검증하는 무결성(integrity) 등을 고려하여 이 시스템을 설계하였다.

2. 본론

함수 단위 커널 업데이트 기법을 이용한 업데이트 시스템의 설계에는 실제로 업데이트를 수행하기 위한 동적 업데이트 관리자 (Dynamic Update Manager)와 그 외 다른 부가적인 컴포넌트들-업데이트 어시스턴트 (Update Assistant), 업데이트 메인테이너 (Update Maintainer), 무결성 관리자 (Integrity Manager), 시스템 업데이트 모듈 (System Update Module)-이 포함된다. 각각의 컴포넌트들은 유저 영역 또는 커널 영역에 위치하고 이를 통해 두 부분으로 나누어 볼 수 있다. 유저 영역의 컴포넌트인 업데이트 어시스턴트는 사용자가 커널의 업데이트 시스템에 접근할 수 있는 인터페이스를 제공한다. 커널 영역에 있는 다른 컴포넌트들은 업데이트할 코드의 무결성을 검증하고, 업데이트를 실제로 수행하며, 동적으로 수행

된 업데이트의 내용을 유지하는 역할을 한다.

우리는 위와 같이 설계한 함수 단위 업데이트 시스템을 인텔 플랫폼 기반의 리눅스 커널 2.6.15 버전에 구현하였다. 동적 업데이트는 트랩 기반의 코드 스플라이싱 기법[6]을 사용한다. 따라서 업데이트의 수행은 본래 함수의 초반부에 분기를 위한 트랩 명령어를 삽입함으로써 이루어지고, 업데이트된 함수의 호출은 동적 업데이트 매니저에 포함된 트랩 핸들러를 통해 수행된다. 이러한 구조로 구현하고자 할 때 해결해야할 문제점은 크게 두 가지가 있다. 첫 번째는 함수 인자의 전달 문제이다. 트랩 핸들러는 인터럽트 호출에서의 복귀를 위한 추가적인 정보를 스택에 저장하는데, 이 때문에 일반적인 함수 호출 과정과 달리 스택을 통한 인자 전달이 정확히 이루어지지 않게 된다. 따라서 우리는 스택에서 함수 인자 위에 저장되는 추가적인 정보를 임시 공간에 옮겨둠으로써 문제를 해결하였다. 두 번째는 커널 외부에서 커널 함수들을 이용하여 업데이트 패치를 개발하고자 할 때 발생하는 커널 내부 심볼에의 접근 문제이다. 보안상의 이유로 커널 내 많은 함수들이 외부로 노출되어 있지 않기 때문에 업데이트의 개발 및 적용이 어렵다. 이 문제를 해결하기 위해 우리는 LKM 시스템에 커널 함수에 대한 자유로운 접근을 허용하는 다른 인터페이스를 추가하였다.

완성된 시스템을 실제로 사용하고 평가하기 위해 우리는 EXT3 파일 시스템을 업데이트하는 간단한 실험을 수행하였다. 우리는 파일 시스템의 특정 Inode의 내용이 변경되었을 때마다 호출되는 함수를 수정하여 간단한 메시지를 출력하도록 하는 내용의 업데이트를 수행하였다. 그리고 업데이트 모듈이 커널에 설치되고, 출력하도록 한 디버그 메시지가 문제없이 출력되는 것을 확인하였다. 이러한 함수 단위의 동적 업데이트 시스템을 이용한 업데이트 과정의 편의성은 같은 내용의 업데이트를 기존의 방식인 LKM을 이용해 수행했을 경우와 비교하여 크게 두 가지 장점을 가진다. 첫째로 수정하고자 하는 함수만을 포함한 바이너리 이미지만으로도 실행중인 EXT3 파일 시스템을 업데이트 할 수 있고, 둘째로 커널 메모리에 이미 로딩된 함수를 직접 수정하기 때문에 파일 시스템을 언마운트 하는 등의 시스템 서비스 가용성을 저해하는 작업이 필요치 않다.

3. 결론

본 논문을 통해 우리는 동적인 커널 업데이트를 함수 단위 업데이트로 수행할 수 있는 시스템을 설계하고 구현하였다. 이 시스템은 C 언어 레벨에서 함수 단위로 작업할 수 있는 편리한 환경을 제공하고, 이를 통해 기존의 연구들이 어셈블리 언어 레벨에서 바이트 코드 단위로 작업하던 한계를 극복하여 동적인 커널의 확장성이 실제로 적용될 수 있도록 하였다. 이를 위해 세부적으로 트랩 기반의 동적 커널 재구성 기법을 사용하였고 실제 구현 단계에서는 인터럽트 서비스 루틴 내로 함수의 인자를 전달하기 위한 기법, 노출되지 않은 커널 함수 모두에 접근할 수 있는 기법을 개발하였다. 또 제안한 시스템을 사용하여 간단한 실험을 통해 업데이트 과정이 실제 동적으로 이루어지는 것을 검증하고 수행 과정의 편의성이 증대되었음을 보였다.

참고문헌

[1] Ariel Tamches and Barton P.Miller, Fine-grained dynamic instrumentation of commodity operating system kernels, PhD thesis, University of Wisconsin, 2001

[2] B. Bershad et al., Extensibility, Safety and Performance in the SPIN Operating System, In Proceedings of the 15th ACM SOSP, pp. 267-284, 1995

[3] Andrew Baumann et al., Module Hot-Swapping for Dynamic Update and Reconfiguration in K42, LCA 2005

[4] D. J. Pearce, P. H. J. Kelly, T. Field, and U. Harder., GILK: A dynamic instrumentation tool for the linux kernel, In Computer Performance Evaluation, pp. 220-226, 2002

[5] Douglas P. Ghormley et al., SLIC: An extensibility system for commodity operating systems, In USENIX Annual Technical Conference, pp. 39-52, 1998

[6] Young-Pil Kim, Jin-Hee Choi and Chuck Yoo, A Trap-based Mechanism for Runtime Kernel Modification, 6th International Conference on Computer and Information Technology, 2006.