리눅스 기반 임베디드 시스템의 성능 분석을 위한 소프트웨어 계층별 문맥 분석 기법

곽상헌*, 이호림*, 이남승*, 임성수*
*국민대학교 컴퓨터공학과
e-mail : hunnyk7@gmail.com, nerigo79@gmail.com, mainly82@gmail.com, sslim@kookmin.ac.kr

Hierarchical Software Context Analysis Technique for Performance Analysis of Linux Based Embedded System

Sangheon Kwak*, Horim Lee*, Namseung Lee*, Sung-Soo Lim*
*Dept. of Computer Science, Kook-min University

요 약

임베디드 시스템이 복잡해질수록, 성능 병목을 파악할 수 있는 성능 분석도구가 요구되고 있다. 현존하는 리눅스 기반의 소프트웨어 성능 분석 도구는 전체 시스템 관점에서 소프트웨어 계층에 따라 응용 소프트웨어의 실행 흐름을 파악하기에는 한계가 있다. 본 논문에서는 모든 소프트웨어 계층에서의 실행 흐름을 일관적으로 파악할 수 있는 문맥 분석 기법을 제안한다. 이 도구를 통해서 사용자 영역의 사용자 정의 함수와 라이브러리 함수, 커널 영역에서의 시스템 호출과 커널 이벤트에 대한 호출자와 피호출자에 대한 관계를 모두 파악할 수 있다.

1. 서론

최근 모바일 기기는 운영체제를 탑재하는 형태로 발전하고 있다. 하지만 운영체제가 탑재된 이동 단말 환경에서 소프트웨어를 개발하고 성능을 분석하는 것은 일반 PC 환경 수준으로 제공되지 못하여, 시스템의 성능 병목을 쉽게 파악할 수 없는 상황이다.

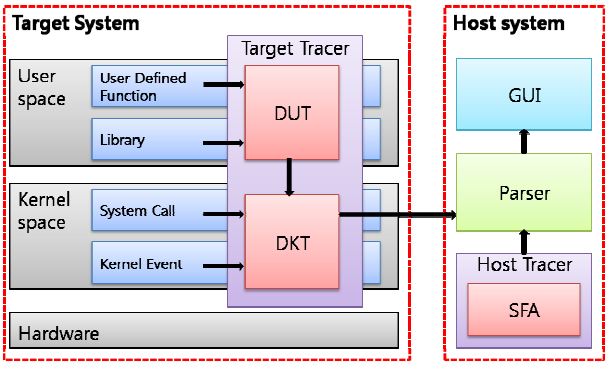
제한된 자원을 사용하는 이동 단말에서는 응용 소프트웨어가 호출하는 라이브러리 함수, 시스템 호출, 커널 이벤트의 응답시간이 응용 소프트웨어의 성능에 크게 영향을 미치기 때문에, 이 요소들을 함께 고려하는 것이 병목지점을 찾고 성능을 개선하는 데에 유용하다. 하지만, 기존의 리눅스 기반의 응용 소프트웨어 성능 분석은 한 소프트웨어 계층에서의 정보만을 분석하는 형태였다. 예를 들면, Gprof 는 사용자 영역에서 수행되는 사용자 정의 함수간 호출관계와 소요시간만을 분석하며, 해당 함수가 호출하는 라이브러리나 시스템 호출에 대한 정보는 알 수 없다. Strace 와 Ltrace 의 경우, 라이브러리 함수와 시스템 호출의 발생과 소요시간만을 분석해 주며, 이것이 어떤 사용자 정의 함수에 의해 발생한 것인지 알 수 없다. 커널 이벤트를 추적하는 LTTng 의 경우, 단지 커널 영역의 시스템 호출과 커널 이벤트의 발생과 수행시간 등의 분석만 수행하며, 사용자 영역의 어떤 사용자 정의 함수나 라이브러리 함수에 의한 커널 영역으로의 진입인지 알지 못한다.

본 논문에서는 리눅스 기반 임베디드 시스템에서 실행되는 응용 소프트웨어의 실행 문맥을 모든 소프트웨어 계층별로 파악할 수 있는 성능 분석 기법을 제안한다. 이 기법을 통해 응용 소프트웨어의 실행 흐름에 따라 발생하는 모든 사용자 정의 함수, 라이브러리 함수, 시스템 호출 및 커널 이벤트에 대한 정보를 추적할 수 있다.

실험 결과 응용 소프트웨어에 대한 모든 시스템 계층간의 함수와 이벤트 처리부의 호출 - 피호출의 관계를 모두 파악할 수 있는 호출 그래프를 도출하였다.

2. 계층별 추적 및 연계 분석 시스템

본 논문에서 제안하는 계층별 통합 성능 분석 도구는, (그림 2) 와 같이 호스트 시스템과 타겟 시스템이 연결되는 구조를 가진다.



(그림 1) 시스템의 전체 구조

타겟 시스템에는 타겟 시스템의 사용자 영역과 커

널 영역을 추적을 수행하는 DUT(Dynamic Userspace Trace, 동적 사용자 영역 추적기)와 DKT(Dynamic Kernelspace Trace, 동적 커널 영역 추적기)가 있다. 각 추적기는 응용 프로그램의 수행 시 동적으로 해당 영역의 정보를 추적한 후 결과를 호스트 시스템에 전달한다.

DUT 는 리눅스의 Ptrace 기법을 통해 실행 파일을 분석하여 사용자 정의 함수와 라이브러리 함수를 동적으로 추적한다. DKT 는 LTTng 의 마커를 소스 파일에 삽입하여 시스템 호출과 커널 이벤트를 동적으로 추적한다.

호스트 시스템에는 응용 프로그램의 심볼 값의 분석을 통해 정적인 함수 호출관계를 분석하는 SFA(Static Function Analyzer, 정적 함수 분석기)가 있다. 이것은 동적 추적으로 파악할 수 없는 소스 수준의 호출관계를 그려주는 역할을 하며, 이 정적 분석의 결과를 타겟 시스템에서 동적으로 추적한 결과와 조합하여 다양한 실행흐름 상에서의 실행 경로를 파악할 수 있다.

각 소프트웨어 계층의 정보를 개별적으로 얻게 되면 영역간의 호출-피호출 관계를 알 수 없기 때문에, 각각의 추적 기능을 시스템 호출을 통해 연동하여 일관된 실행 흐름으로 파악할 수 있게 하였다. DUT 에서 추적이 발생할 때마다 시스템 호출을 통해 DKT 로 추적 정보를 전달하면, DKT 의 결과를 분석만으로 DUT 가 추적한 정보를 모두 얻을 수 있어, 대상 어플리케이션에 대한 호출-피호출 관계를 파악할 수 있게 된다.

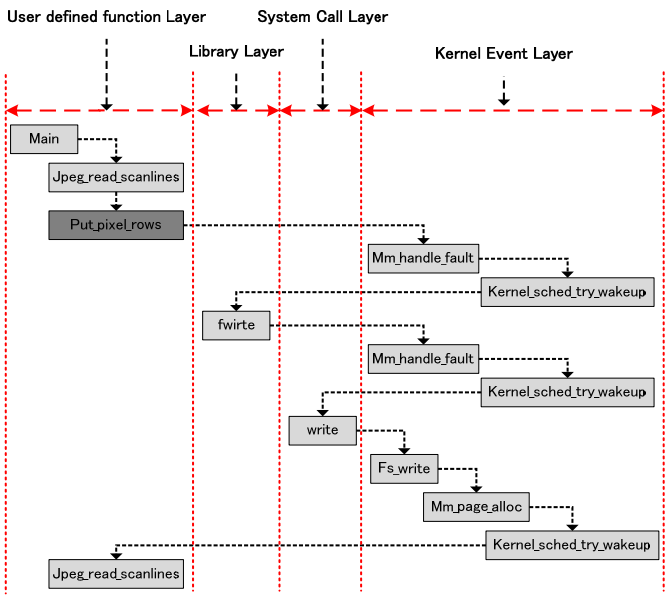
3. 실험결과 및 결론

본 실험은 PXA270 기반의 임베디드 참조 플랫폼을 타겟 시스템으로 하여, 리눅스 커널 2.6.26.3 버전을 탑재하여 수행하였다. 호스트 시스템은 Intel x86 기반의 일반 PC 에서 Vmware 를 통해 가상화한 Ubuntu8.04 를 사용하였다. Mibench 의 JPEG decoder 를 추적하여 각 계층별 호출정보 및 호출 문맥의 연관 관계를 도출하였다. (표 1)과 같이 사용자 정의 함수, 라이브러리, 시스템 호출, 커널 이벤트에 대한 발생 정보를 각 계층별로 파악할 수 있다. 그리고, 특정한 함수에 대한 전체 소프트웨어 계층간의 호출 - 피호출 관계를 분석할 수 있다. (그림 2)와 같이, 사용자 함수인 Put_pixel_rows 함수의 경우, main 함수에 의해 Jpeg_read_scanelines 함수 호출 이후 실행되며, 실행 코드를 메모리에 적재하는 과정에서 mm_handle_fault 가 발생하다가, 라이브러리 함수의 요청인 fwrite 를 수행하기 위해 fs_write 및 mm_page_alloc 등의 커널 이벤트 처리부 호출된 후 다시 사용자영역의 다음 함수를 수행하는 것을 확인할 수 있다.

제안하는 기법을 통해 사용자 영역의 함수 내부의 라이브러리 및 시스템 호출, 커널 이벤트의 발생 비율을 확인하여 성능 병목의 원인을 파악하는 데에 사용할 수 있다.

Name(User defined function)	calls	Time	%
jpeg_fill_bit_buffer	2002	0.020934	21.62
jpeg_idct_islow	1536	0.016064	16.59
Decode_mcu	256	0.011889	12.28
Dcompress_onepass	16	0.008209	8.48
Set_upsample	256	0.005493	5.67
Jpeg_huff_code	392	0.004171	4.31
Process_data_context_main	256	0.003928	4.06
Jpeg_read_scanlines	256	0.003886	4.01
Put_pixel_rows	256	0.003853	3.98
Name(Library function)	**calls**	**Time**	**%**
Fwrite	266	0.002913	38.53
Memset	258	0.002822	37.32
__ctype_b_loc	124	0.000134	16.92
Malloc	10	0.000111	1.77
Free	10	0.000084	1.47
memcpy	8	0.000032	1.11
Name(System call)	**calls**	**Time**	**%**
Write	49	0.000101	53.16
Exit_group	1	0.000018	9.47
Brk	14	0.000014	7.37
Open	6	0.000012	6.32
Read	5	0.000010	5.26
Close	5	0.000010	5.26
Mmap	9	0.000009	4.74
Name(Kernel event)	**calls**	**Time**	**%**
Kernel_sched_try_wakeup	26062	0.026062	51.38
Mm_handle_fault	14453	0.014750	29.08
Kernel_irq	2109	0.003945	7.78
Kernle_softirq	1208	0.003211	6.33
Kernel_timer_set	1256	0.001256	2.48
Fs_write	49	0.001173	2.31

(표 1) 본 도구로 JPEG decoder 를 추적한 결과



(그림 2) JPEG decoder 의 Put_pixel_rows 함수의 소프트웨어 계층별 실행 흐름