효율적인 가상화 시스템 프로파일링을 위한 성능측정 프레임워크

Performance Measurement Framework for Efficient Virtualization System Profiling

장은태*, 최상훈*, 박기웅[1)]
Eun-Tae Jang, Sang-Hoon Choi, Ki-Woong Park

(05006) 서울특별시 광진구 능동로 209 세종대학교 시스템보안연구실*, 세종대학교 정보보호학과[1)]
{euntaejang, csh0052}@gmail.com, woongbak@sejong.ac.kr

요 약

최근 클라우드 컴퓨팅이 확산되면서 주목 받고 있는 기술 중 하나는 가상화(virtualizaion) 기술이다. 가상화 기술을 이용하여 시스템을 구축할 경우 하나의 호스트 운영체제에서 복수개의 운영체제를 구동시킬 수 있으며 효율적인 컴퓨팅 자원의 관리를 용이하게 한다는 장점이 있지만 하이퍼바이저(Hypervisor) 위에서 구동하는 운영체제들이 많아질수록 가상화 시스템의 전반적인 성능을 측정하는 것이 필요하며 이는 중요한 기술로 떠오르고 있다. 본 논문에서는 가상화 시스템의 효율적인 성능측정을 위해 기존 프로파일링 도구의 주요 기능을 분석하여 가상화 시스템에서 발생할 수 있는 이벤트에 대하여 모니터링 도구들이 수행할 수 있는 프로파일링 커버리지를 측정하고 분류하였으며, 더 나아가 모니터링을 수행하는 원격 시스템에서 수신 된 정보에 따라 가상화 시스템에 성능측정이 필요할 경우 적합한 프로파일링 도구를 게스트 시스템에 적재시켜 성능측정을 할 수 있도록 하는 프레임워크를 연구하였다.

Abstract

Virtualization technology is one of the technologies that have been attracting attention as cloud computing spreads recently. When a system is constructed using virtualization technology, mutiple operation systems can be operated in a single host operating system, thereby facilitating efficient management of computing resources. As more and more operating systems are running on the hypervisor, it is important to measure the overall performance of the virtualization system and this is becoming an important technology. In this paper, we analyze the main functions of the existing profiling tools to measure the performance of the virtualization system, and measure and classify the profiling coverage that the monitoring tools can perform for events that may occur in the virtualization system. In addition, we have studied a framework that enables performance measure-

※ 본 연구는 2019년도 과학기술정보통신부의 재원으로 정보통신기획평가원의지원(No.2018-0-00420, No.2019-0-00273) 및 한국연구재단 연구과제(NRF-2017R1C1B2003957)의 지원을 받아 수행된 연구임.

1) 교신저자

논문투고일자 : 2019. 06. 07 심사일자 : 2019. 06. 07 게재확정일자 : 2019. 06. 21

ment by loading appropriate profiling tools into the guest system when performance measurement is required for the virtualization system according to the information received from the remote system performing the monitoring.

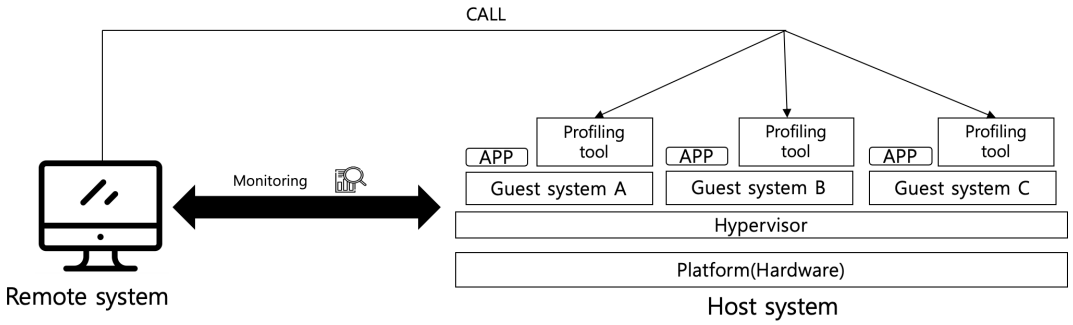
키워드: 프로파일링, 프로파일링 도구, 가상화, 하이퍼바이저, 이벤트 홀(Hole)
Keyword: Profiling, Profiling tool, Virtualization, Hypervisor, Hole

1. 서론

최근 클라우드 컴퓨팅이 확산되면서 주목 받고 있는 기술 중 하나는 가상화(virtualizaion) 기술이다 가상화 시스템을 이용할 경우 하나의 호스트 운영체제에서 여러 개의 다양한 운영체제를 구동시킬 수 있다는 장점이 있다. 이러한 운영체제들은 하이퍼바이저(Hypervisor)라는 플랫폼 위에서 동작하게 되며 하이퍼바이저 위에서 구동하는 운영체제들이 많아질수록 가상화 시스템의 전반적인 성능을 측정하는 것이 필요하며 이는 중요한 기술로 떠오르고 있다[10] [11] [12]. 현재 가상화 시스템을 프로파일링 하는 방법은 (그림 1)과 같이 원격에서 모니터링을 통하여 문제가 발생하였을 경우 해당 게스트 시스템에 설치되어있는 있는 프로파일링 도구를 실행시켜 성능측정을 하는 방법이 있다. 하지만 이러한 성능측정 방법은 각각의 운영체제마다 도구들이 설치되어 있어야 한다는 점에서 운영체제의 크기를 증가 시키며 오버헤드가 커지게 된다. 또한 만약 운영체제 A와 운영체제 B가 서로 같은 운영체제라한다면 이는 곧 하나의 가상화 시스템에 동일한 도구 두 개를 가지고 있게 된다는 문제점이 있다.

이러한 한계점을 극복하기 위하여 원격에서 모니터링을 수행하는 원격 시스템에서 가상화 시스템 위에서 동작하는 게스트 시스템들의 전반적인 성능을 측정할 수 있는 도구들을 포함하고 있으며, 모니터링을 통해 성능측정이 필요할 경우 문제범위를 파악하여 측정이 요구 될 때마다 해당 범위의 성능을 측정할 수 있는 도구를 구동중인 게스트 시스템에 적재시켜 운영체제, 하이퍼바이저, 응용프로그램, 하드웨어를 대상으로 프로파일링을 수행할 수 있는 프레임워크를 제안하고자 한다. 제안하고자 하는 프레임워크는 (그림 2)와 같다.

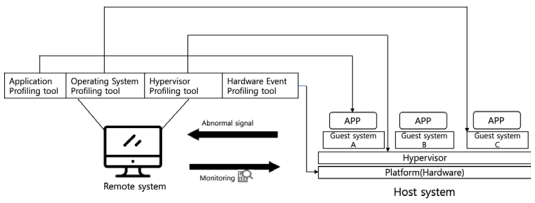
이를 위해서는 모니터링을 수행하는 원격 시스템에서 효율적인 프로파일링 도구를 포함하고 있는 것이 중요하며, 프로파일링 도구들이 시스템를 대상으로 성능측정을 할 경우 이러한 도구들이 대상의 성능측정을 위하여 사용하는 기술에 대하여 조



(그림 1) 기존 가상화 시스템에서의 프로파일링 방법

사해야한다. 또한 도구를 시스템에 효율적으로 적재하기 위한 연구를 수행하여 이를 프레임워크에 적용해야 한다.

본 논문에서는 원격 시스템에서 가상화 시스템 모니터링을 통하여 성능측정이 필요한 대상에 따라 프로파일링 도구를 효율적으로 적재하여 성능측정이 이루어 질 수 있도록 하는 프레임워크를 연구하였다. 본 논문의 구성은 다음과 같다. 2장에서는 프로파일링 분석대상을 응용프로그램, 운영체제, 하이퍼바이저, 하드웨어 이벤트로 분류하고, 3장에서는 현재 사용되고 있는 대표적인 프로파일링 도구들의 주요기능을 분석하며 4장에서는 프로파일링 도구를 효율적으로 적재하기 위하여 도구들이 시스템을 대상으로 성능측정을 할 경우 사용되는 기술에 대해 조사하여 이를 제안하는 프레임워크에 적용하였으며 5장에서는 결론을 제시한다.



(그림 2) 모니터링을 수행하는 원격 시스템에서 호스트 시스템의 성능측정을 위한 도구 적재

2. 가상화 시스템의 성능 측정을 위한 대상 분류

가상화 시스템의 성능을 측정하기 위한 프로파일링 대상을 다음과 같이 4개로 분류하였다.

- Application profiling : 커널 위에서 동작되는 응용 프로그램을 대상으로 수행하는 성능분석 조사이다. 리눅스에서 사용하는 명령어들 또한 커널에서 실행시키는 응용프로그램이며 개발자가 제작한 프로그램도 대상에 포함 된다. 프로그램이 실행될 때의 CPU, Memory의 성능을 분석한다. 또한 함수의 콜 횟수와 시간들을 분석하며 프로세스의 상태와 발생되는 인터럽트들을 조사한다.
- Operating system(kernel) profiling : 전반적인 운영체제 시스템을 대상으로한 성능분석 조사이다. 주로 한정된 시스템 자원을 효율적으로 관리하는지 프로세스 스케줄링을 효과적으로 하는 지 분석하며 응용프로그램과 하드웨어 사이의 상호작용이 잘 동작되고 있는지 프로파일링 한다.
- Hypervisor profiling : 다수의 운영체제를 동시에 실행하기 위하여 가상 프로세서, 파티션 정보, 논리 프로세서 정보를 분석하며 메모리가 효율적으로 관리되고 있는지, 시스템 자원을 효율적으로 분리하였는지 성능측정을 한다.
- Hardware events profiling : 하드웨어에서 발생된 특정 이벤트를 이용한다. CPU performance counter가 인터럽트를 발생시킬 때마다 성능과 관련된 데이터를 수집할 수 있으며, CPU clock, 캐시 미스, 캐시 일관성 이벤트, 분기 오 예측, 정지 된 사이클 등과 같은 이벤트를 기반으로 성능분석을 한다.

프로파일링 분석대상을 응용프로그램, 운영체제, 하이퍼바이저, 하드웨어 이벤트로 분류하였고, 분류 대상에 따라 수행될 조사와 수집되어야 할 정보들을 분류하였다.

3. 가상화 시스템 성능 측정을 위한 대상 분류

현재 성능측정을 위해 사용되고 있는 대표적인 프로파일링 도구들의 주요 기능을 분석 및 분류하여 도구들 중 어떠한 기능과 대상이 개발 될 프레임워크에서 이용될 수 있을지에 대한 방향을 제시하고자 한다.

3.1 프로파일링 도구

프로파일링 도구는 시스템의 성능을 측정하기 위한 도구이다. 현재 시스템의 성능 측정을 위해 개방

형 운영체제에서 사용되고 있는 도구들 중 Oprofile, perf 등 대표적인 8개의 도구를 분석하였다. 프로파일링 도구들을 시스템, 응용프로그램, 하드웨어 이벤트, 하이퍼바이저로 분류하였을 때, 각각의 도구들이 해당 기능 수행 여부는 <표 1>과 같다.

3.2 프로파일링 도구 주요 기능

- Oprofile[1] : 성능측정을 위해 CPU의 perfomance counter를 이용하여 성능을 측정한다. 커널, 응용프로그램, 하드웨어 인터럽트 핸들러, 공유라이브러리 등의 모든 code의 성능을 측정, 분석하여 여러 형태의 보고서를 생성하며 gprof의 함수호출 그래프 프로파일링을 제공할 수 있다. 또한 XEN 하이퍼바이저에 대한 프로파일링 기능을 제공한다. 이 도구는 리눅스, 유닉스, Windows server의 다양한 운영체제에서 동작이 가능하며, intelx86, AMD, Alpha, ARM, IBM PowerPC등 다양한 CPU를 지원한다.
- Gprof[2] : 특정 응용 프로그램에서 어떤 함수가 어떤 함수를 호출하는지 설명하고 소비된 시간을 기록한다. 하지만 분석을 위해서는 프로그램의 소스코드가 필수적이며, 언어도 제한적이다. 또한 오직 함수호출 그래프만 생성할 수 있으며, gprof를 사용하기 위해서는 응용프로그램을 다시컴파일 해야 한다.
- Rational PurifyPlus[3] : 성능 프로파일링 도구이며 메모리 디버거와 코드 커버리지 도구이다. 프로그램에서 C또는 C++로 작성 된 소프트웨어에서 메모리 접근 오류를 발견하기 위하여 사용하며, 리눅스, 솔라리스, 윈도우즈 등 여러 운영체제에서 지원한다.
- DynamoRIO[4] : 동적 프로그램 분석 도구 개발을 위한 프레임 워크이며, 안드로이드, 리눅스, 윈도우즈 운영체제에서의 유저레벨의 응용프로그램을 대상으로 한다. 원래 DynamoRIO는 동적 바이너리 최적화 시스템으로 만들어졌지만 이후 보안 및 디버깅 분석 도구에 사용되었다.
- CodeXL[5] : AMD에서 개발한 CodeXL은 통합 개발 도구 모음으로서, AMD, Intel CPU에서 지원되고 있으며 개발자로 하여금 CPU, GPU 및 APU의 이점을 활용 할 수 있도록 해준다. 이 도구는 GPU 디버깅 기능, CPU, GPU 프로파일링 기능, 커널 분석 등이 가능하며 리눅스와 윈도우즈에서 사용가능하다. 또한 XEN 하이퍼바이저에서 실행되는 운영체제에 대한 CPU 프로파일링 기능도 제공한다.
- Dtrace[6] : 사용자가 DTrace API를 이용하여 커널과 통신할 수 있도록 모듈을 제공하며, Application runtime 측정, 분기 제어, 커널 스케줄러 및 기타 민감한 서브시스템을 프로파일링하며, 가상화 기술을 분석할 수 있다.

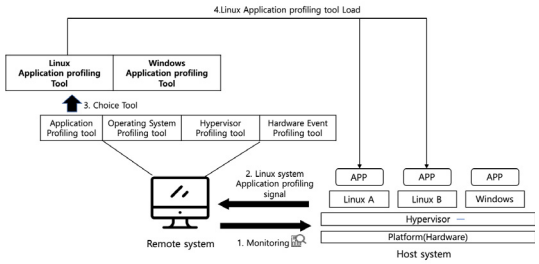
〈표 1〉 프로파일링 도구 목록과 분석대상에 따른 분류

Tool	Application	OS (kernel)	Hypervisor	Hardware Event	Open Source
Oprofile[1]	O	O	O	O	O
Gprof[2]	O	X	X	X	O
Rational PurifyPlus[3]	O	X	X	X	X
dynamoRIO[4]	O	X	X	X	O
Perf[5]	O	O	O	O	O
Dtrace[6]	O	O	O	X	O
CodeXL[7]	O	O	O	O	O
GCOV[8]	O	O	O	X	O

수집된 데이터에 대한 무결성을 제공하며 solaris, MacOS, FreeBSD 커널에서 작동한다.

- Perf[7] : 2011년부터 리눅스 커널에 포함되어 배포 되고 있는 도구이며, 활발하게 개발되고 있다. Oprofile과 함께 리눅스 시스템에서 가장 많이 사용되는 프로파일링 도구이며, Oprofile 도구와 같이 별도의 복잡한 설정 과정이 필요가 없다. Intel과 AMD CPU에서 사용이 가능하며, Perf 도구에서 개별적으로 정의한 4가지 영역의 이벤트 PMU를 통한 하드웨어 이벤트, 커널에서 제공하는 소프트웨어 이벤트, trace point이벤트, 사용자 정의 probe 이벤트를 수집하는 envent sampling을 통해 특정 프로그램 또는 시스템 전체 성능을 분석할 수 있으며, 어떤 함수와 소스코드가 CPU를 많이 사용하는지를 파악하여 시스템 성능 분석 결과를 GUI로 제공해준다. 어셈블리 또는 소스코드 단위로 시스템 상에서 발생 된 오버헤드를 분석할 수 있으며, kernel API tracing 기능을 제공한다. 또한 가상화 환경에서 KVM과 XEN하이퍼바이저에 대한 이벤트를 모니터링 할 수 있다.
- GCOV[8] : gcov는 프로그램 코드의 어느 부분을 최적화 시키는 것이 가장 좋은 선택인지를 판단하기 위한 테스트 커버리지 프로그램이며, 하이퍼바이저의 성능측정 기능을 제공한다. 일반적으로 gcc와 함께 사용되며 리눅스에서 사용할 수 있다.

주요 프로파일링 도구들의 주요기능을 분석했을 때 Perf, Oprofile, CodeXL가 시스템, 응용프로그램, 하이퍼바이저, 하드웨어 이벤트의 분석이 가능했다. Perf와 Oprofile은 리눅스에서 사용되었던 Gprof, Dtrace등의 기능을 장점을 포함하여 개발되었다고 한다. 하지만 오직 리눅스 운영체제에서만 사용이 가능하다는 점과 CLI 환경만 지원한다는 점에서 한계점이 존재했다. CodeXL은 CLI와 GUI를 모두 지원하기 때문에 사용자가 보다 쉽게 분석이 가능했으며, 윈도우즈 운영체제와 리눅스 운영체제에서 사용이 가능했다. 또한 Perf와 Oprofile 도구에서는 지원하지 않는 GPU 프로파일링, GPU 디버깅 등의 기능을 제공한다.



(그림 3) 가상화 시스템의 성능측정을 위한 도구 적재 순서도

(그림 3)에서는 프로파일링 도구의 주요기능에 따라 측정대상을 크게 <표 1>과 같이 분류하여 모니터링을 통해 성능측정이 되어야할 대상에 따라 도구를 선택하여 시스템에 적재한다.

4. 모니터링을 통한 프로파일링 도구 적재

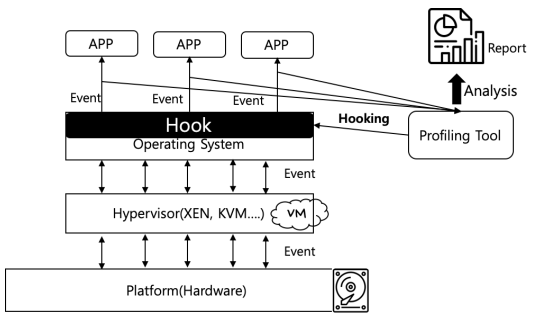
본 논문에서는 원격 시스템에서 모니터링을 통하여 전달 받은 이벤트를 분석하여 현재 수행되어야 할 프로파일링 도구를 가상화 시스템에 적재시키는 방법에 대한 필요성을 제시하였다. 가상화 시스템의 게스트 시스템에 설치 되어있는 프로파일링 도구를 실행하는 것이 아닌, 모니터링을 통해 실행 되어야 할 프로파일링 도구를 원격 시스템에서 전달 받은 이벤트를 분석하여 선택한 후 해당 시스템에 적재하는 프레임 워크를 구축하는 것을 목표로 한다.

본 장에서는 프로파일링 도구를 시스템에 효율적으로 적재시키기 위한 방법을 도출하기 위하여 시스템에 대하여 성능측정이 가능한 프로파일링 도구의 이벤트 수집 원리를 조사하며, 제안하는 프레임 워크에서의 기존 프로파일링 도구들이 사용하는 이벤트 수집원리를 그대로 적용할 경우의 문제점 도

출하고 제안하는 프레임워크에서 적용하고자 하는 프로파일링 도구의 효율적인 적재를 위한 방법 제시하고자 한다.

4.1 프로파일링 도구의 이벤트 수집 원리

프로파일링 도구는 다양한 기술을 이용하여 데이터를 수집한다. 이를 위한 기술에는 하드웨어 인터럽트, 코드 계측, 명령어 집합 시뮬레이션, 운영 체제 후킹 등 여러 가지 방법이 있다. 일반적으로 프로파일링 도구가 시스템으로부터 이벤트를 수집하기 위하여 공통적으로 사용하는 방법은 운영체제를 후킹하는 방법이 있다[9]. 이에 대한 원리는 (그림 4)와 같다.

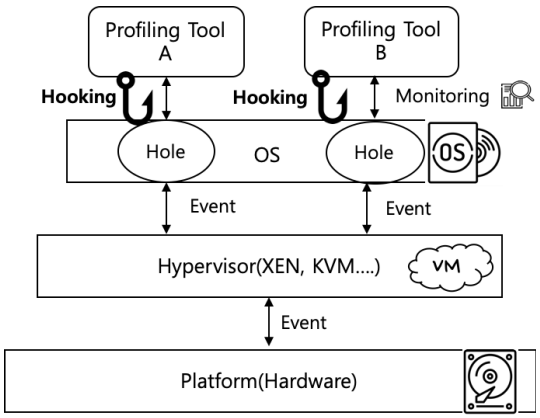


(그림 4) 운영체제 후킹을 통한 Event 수집

(그림 4)와 같이 프로파일링 도구는 운영체제 레벨에서 후킹을 사용하여 운영체제 계층의 밑에 위치한 하이퍼바이저와 하드웨어에서 발행하는 이벤트를 수집할 수 있다.

본 논문에서는 프로파일링 도구가 시스템의 전반적인 측정을 위하여 이벤트를 수집하는 일종의 통로를 이벤트 홀(Hole)이라는 단어를 사용하여 정의하였다.

(그림 5)에서는 시스템, 하이퍼바이저, 하드웨어를 대상으로 성능측정을 하기위해서 각각의 프로파일링 도구는 운영체제에 이벤트 홀을 통하여 이벤트를 수집한다.



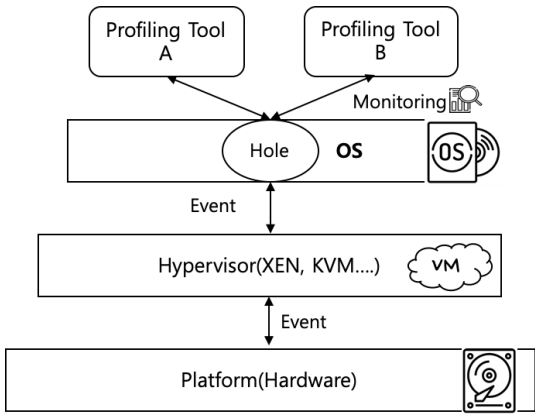
(그림 5) 프로파일링 도구의 이벤트 수집 원리

4.2 효율적인 프로파일링 도구 적재를 위한 프레임워크

(그림 5)에서 프로파일링 도구 A와 B가 모니터링을 통하여 이벤트 홀을 사용하여 하드웨어, 하이퍼바이저, 운영체제로부터 이벤트를 수집하는 것을 확인 할 수가 있다. 두 가지의 프로파일링 도구를 사용하여 시스템의 성능측정을 할 때, 이벤트 홀이 두 개가 만들어지는 것을 확인할 수 있다. 이는 곧 프로파일링 도구 두 개로 시스템의 성능측정을 할 경우 운영체제 후킹이 두 번 이루어진다는 것을 의미한다. 이러한 특징은 성능측정을 위하여 실행되는 프로파일링 도구의 수가 증가할 때마다 운영체제에 생성되는 이벤트 홀의 수도 점점 증가할 것이다.

본 논문에서는 하드웨어. 하이퍼바이저, 운영체제의 이벤트를 응용프로그램 레벨에서 수신할 수 있는 단 하나의 이벤트 통로만을 프레임워크에 적용하여, 실행되는 프로파일링 도구마다 이벤트 홀을 생성하지 않아도 필요한 이벤트를 받아올 수 있는 방안을 제시한다. 여러 프로파일링 도구를 실행시킬 경우 제안한 프레임워크의 이벤트 수집을 위한 하나의 이벤트 홀을 공유하여 성능측정을 한다면 도구마다 운영체제를 후킹하지 않아도 되기 때문에 오버헤드와 성능측정 시간이 감소하며, 성능측정이 이루어진 뒤 도구를 제거 할 경우에도 도구

가 운영체제 후킹을 통하여 이벤트 홀을 생성하지 않아도 되기 때문에 제거가 보다 간단하다. (그림 6)는 가상화시스템의 효율적인 프로파일링을 위하여 하나의 이벤트 홀을 만들어 시스템 이벤트 수집을 위한 모든 프로파일링 도구들이 해당 이벤트 홀을 공유하도록 설계한 프레임워크이다. 시스템의 성능측정을 위해 실행되는 프로파일링 도구가 많아질 수록 이벤트를 받아오기 위하여 운영체제를 대상으로한 후킹의 횟수가 증가할 것이다. 따라서 기존의 시스템 레벨의 성능측정을 위하여 프로파일링 도구가 사용하는 이벤트 홀을 미리 프레임워크에 적용시키는 방안을 도출하였다.



(그림 6) Hole을 공유하여 Event를 모니터링

5. 결론

가상화 기술은 최근 클라우드 컴퓨팅이 확산되면서 주목 받고 있는 기술 중 하나이며 가상화 기술을 사용하여 하나의 호스트 시스템의 하이버바이저에서 여러 개의 운영체제를 동작시킬 수 있다는 장점이 있다. 구동 되고있는 운영체제가 증가 할수록 가상화 시스템의 전반적인 성능을 효율적으로 측정하는 것이 필요하다. 이를 위해 모니터링을 수행하는 원격 시스템에서 성능측정을 위한 도구를 포함하여 각각의 구동중인 운영체제에서 도구가 설치되어있지 않아도 전달 받은 이벤트에 따라 프로파일링 도구를 가상화 시스템에 적재하여 성능을 측정할 수 있는 프레임워크를 제안하였다.

본 논문에서는 가상화 시스템의 성능측정을 위해 모니터링을 수행하는 원격 시스템에서 프로파일링 도구를 시스템에 효율적으로 적재하기 위한 연구를 수행하였다. 현재 성능측정을 위해서 사용되고 있는 프로파일링 도구 8개를 선택하여 해당 도구들의 주요기능을 분석하였으며, 이를 바탕으로 성능측정의 대상인 가상화 시스템에서 발생될 수 있는 이벤트를 분석하여 도구들이 수행할 수 있는 프로파일링 범위에 따라 분류하였다. 또한 프로파일링 도구가 시스템 레벨에서 성능측정을 위해 사용되는 방법인 운영체제 후킹기법을 사용하여 하나의 이벤트의 통로인 이벤트 홀을 만들어 이벤트를 수집하기 때문에 성능측정을 위해 호출되어 실행되는 프로파일링 도구가 증가할수록 이러한 이벤트 홀의 개수 또한 증가할 것이다. 따라서 제안하는 프레임워크에서는 운영체제에 이벤트를 수집할 수 있는 이벤트 홀을 만들어 프로파일링 도구가 시스템을 대상으로 성능측정을 할 경우에도 운영체제의 후킹 없이 가상화 시스템에서 발생하는 이벤트를 수집할 수 있는 방법을 제안하였다. 이와 같은 방법으로 프로파일링 도구들의 성능측정 시간과 오버헤드를 감소시키는 효과를 기대할 수 있을 것이며 성능측정 후의 제거를 할 경우에도 각각의 도구는 운영체제를 후킹하지 않고 만들어 놓은 하나의 이벤트 홀만 사용하여 이벤트를 수집하기 때문에 보다 제거가 간단하다.

이를 통해 본 논문에서 제안한 프레임워크는 향후 다수의 운영체제가 구동되는 가상화 시스템에서의 효율적인 성능측정이 가능할 것으로 기대 될 수 있다.

향후 연구방향으로는 두 가지로 나뉠 수 있다.첫째로는 (그림 6)에서 제안한 이벤트 홀을 실질적으로 구현하여 프로파일링 도구로 성능측정을 수행했을 경우 (그림 5)와 비교하였을 경우 소요되는 시간과 오버헤드가 얼마나 감소할 수 있을지 도식화하는 것과 각각의 조사한 도구들이 제안한 프레임워크에서 실행 될 경우의 발생할 수 있는 문제점을

해결을 위하여 도구들이 운영체제 후킹을 기법을 사용하여 이벤트 홀을 만들지 않더라도 프레임워크에 개발 되어있는 이벤트 홀을 이용할 수 있도록 시스템 라이브러리를 제작하여 기존의 도구들이 새로운 프레임워크에서 동작할 수 있도록 연구할 계획이다.

참고문헌

〔1〕 Cohen, William E. "Tuning programs with OProfile." Wide Open Magazine 1 (2004): 53-62.

〔2〕 Graham, Susan L., Peter B. Kessler, and Marshall K. Mckusick. "Gprof: A call graph execution profiler." ACM Sigplan Notices. Vol. 17. No. 6. ACM, 1982.

〔3〕 Gbajabiamila, Jameel, Animesh Kejriwal, and Yash Patodia. "An Evaluation of Rational PurifyPlus."

〔4〕 Garnett, Timothy. Dynamic optimization if IA-32 applications under DynamoRIO. Diss. Massachusetts Institute of Technology, 2003.

〔5〕 https://gpuopen.com/blazing-codexl-2-2/

〔6〕 McDougall, Richard, Jim Mauro, and Brendan Gregg. Solaris performance and tools: DTrace and MDB techniques for Solaris 10 and OpenSolaris. Prentice Hall,, 2006.

〔7〕 De Melo, Arnaldo Carvalho. "The new linux'perf'tools." Slides from Linux Kongress. Vol. 18. 2010.

〔8〕 Cerveira, Frederico, Raul Barbosa, and Henrique Madeira. "Experience report: On the impact of software faults in the privileged virtual machine." 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2017.

〔9〕 Seo, Jooyoung, Byoungju Choi, and Suengwan Yang. "A profiling method by PCB hooking and its application for memory fault detection in embedded system operational test." Information and Software Technology 53.1 (2011): 106-119.

〔10〕 백현지 "DaaS에서 자원 사용량 기반 효율적 자원 할당 방법" 한국차세대컴퓨팅학회 논문지 14.1, 78-88, 2018.

〔11〕 백승훈 "가상 데스크톱 인프라를 위한 데이터 손실 없는 지속형 인메모리 스토리지", 한국차세대 컴퓨팅학회 논문지. 10.5, 46-55, 2014.

〔12〕 최원석 "컨테이너 기반 클러스터 환경에서 효 율적인 자원관리를 위한 오케스트레이션 방법", 한국차세대 컴퓨팅학 논문지. 15.2, 71-78, 2019.

저자소개

◆ 장은태

- 2018년 세종대학교 정보보호학과 학사
- 2019년 세종대학교 정보보호학과 석사 과정
- 관심분야: 클라우드 컴퓨팅, 시스템보안, 시스템해킹

◆ 최상훈

- 2014년 대전대학교 정보보안학과 학사
- 2016년 대전대학교 전산정보보안학과 석사
- 2017년 세종대학교 정보보호학과 박사 과정
- 관심분야: 클라우드 컴퓨팅, 시스템 보안, 디지털포렌식

◆ 박기웅

- 2005년 연세대학교 Computer Science 학사
- 2007년 KAIST Electrical Engineering 석사
- 2012년 KAIST Electrical Engineering 박사
- 2008년 Microsoft Research Asia, Wireless and Networking Group, Research Intern
- 2009년 Microsoft Research, Network Research Group, Graduate Research Fellow
- 2012년 국가보안기술연구소 연구원
- 2012년~2016년 대전대학교 정보보안학과 교수
- 2016~현재 세종대학교 정보보호학과 교수
- 관심분야: 시스템보안, 모바일-클라우드 컴퓨팅, 보안 프로토콜, 디지털 포렌식 등