유니커널 기반의 클라우드 운영체제 기술

김진미 정연정* 전승협* 차승준* 람닉** 정성인
한국전자통신연구원 책임연구원
한국전자통신연구원 선임연구원 *
한국전자통신연구원 Post-Doc **

I. 서론

기존의 클라우드 운영체제라고 하면 OpenStack, CloudStack과 같은 클라우드 운용 관리 소프트웨어를 뜻하는 바가 크다. 이는 클라우드 서비스의 요구에 맞게 서비스에서 시스템 인프라에 이르기까지 최적화된 기능으로 관리해주고 실행한다. 이러한 소프트웨어는 자원 활용을 극대화하고 인프라 구축을 유연하게 하며 클라우드 서비스 요구를 신속하게 처리하기 위해 연구·개발되어 왔다.

그러나 대부분의 클라우드 소프트웨어는 오래전부터 사용된 커널에서 동작한다. 그리고 그 커널과 응용에 이르는 소프트웨어 스택이 모든 서비스에 동일하게 그대로 탑재되어 사용되고 있어 민첩성과 보안 등에 취약할 수 있다. 이러한 소프트웨어 스택에는 특정 응용 프로그램에서는 전혀 필요로 하지 않는 유틸리티, 드라이버 및 커널 기능이 여전히 포함되어 있다. 실행하는 응용 프로그램과는 관련이 없는 기능에 의해 소비되는 디스크 공간과 메모리가 필요하게 되어 자원 낭비가 있다.

즉, 윈도와 리눅스와 같은 범용 운영체제에서는 여러 응용 프로그램을 탑재하고 실행하여 서비스를 시작하는데 드는 시간을 소모할 뿐 아니라 각 응용 프로그램의 작업 이미지는 점점 더 대용량의 저장 공간과 메모리를 필요로 하므로 클라우드에서 실행되는 작업부하를 최소화하기 어렵다.

이미지 용량이 커질수록 비용을 들여 스토리지 및 서버를 더욱 늘려야 하고, 보유하고 있는 장비가 많아질수록 전력량 소모가 많아지며 이를 유지하는데 필요한 더 많은 데이터센터 공간이 필요하다. 또한, 클라우드 서비스를 실현하는 각기 다른 응용 프로그램이 같은 운영체제에서 동작하므로 보안에 취약하여 작업별로 별도의 보안 장치가 필요하다.

* 본 내용은 김진미 책임연구원(☎ 042-860-4885, jinmee@etri.re.kr)에게 문의하시기 바랍니다.
** 본 내용은 필자의 주관적인 의견이며 IITP의 공식적인 입장이 아님을 밝힙니다.

이러한 문제를 해결하기 위해 클라우드 운용환경에서는 훨씬 작고 빠른 이미지로 변환하여 실행하는데 컨테이너가 사용되고 있다. 하지만 여전히 컨테이너도 범용운영체제 환경에서 실행하므로 충분하게 보안 문제를 해결하지 못하고 있다.

최근 유니커널이 관심을 받고 있는 이유는 단일 응용과 커널이 하나의 실행 이미지로 동작하여 여러 서비스를 처리하는 커널에 비해 프로세스가 엄격하게 분리되어 보안에 유리하다는 것이다. 또한, 실행 이미지가 작고 최적화될 기회가 많아 부트 시간이 빨라져 전통적인 운영체제에 비해 여러 이점을 제공하기 때문이다. 따라서, 유니커널이 작고 안전하며 빠르게 실행할 수 있다는 점에서 클라우드 컴퓨팅에 도움이 될 것이라는 기대감이 커지고 있다[1].

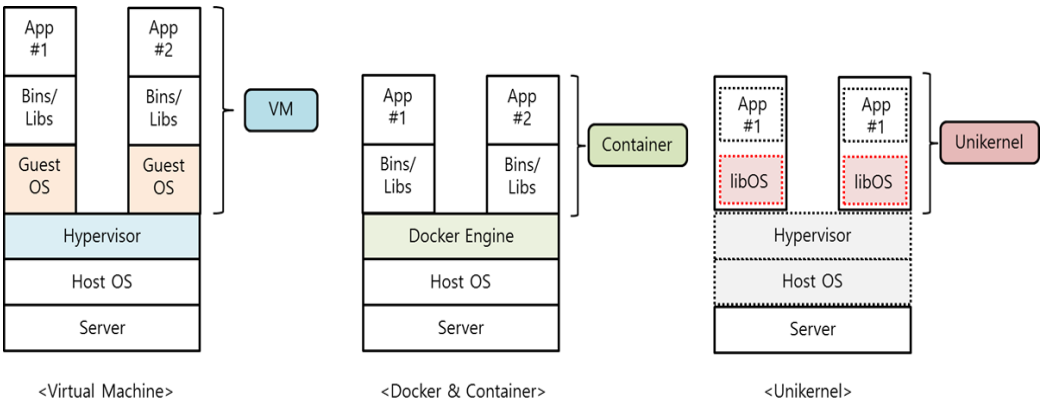
본 고의 II장에서는 클라우드 서비스 기반의 가상화 환경에서 클라우드 시스템의 운영체제 현황에 대해서 설명한다. 주로 클라우드 시스템 인프라스트럭처에 적용되는 가상화 기술의 현황과 유니커널의 일반적인 특징에 관해 설명한다. III장에서는 최근 클라우드 컴퓨팅에 활용되는 유니커널 기술 동향에 관해 기술하며 각 유니커널에 대한 특징에 대해 논한다. 끝으로 IV장에서는 결론 및 시사점을 제시한다.

II. 클라우드 시스템 운영체제 현황

작업 부하를 조정하며 클라우드 서비스를 관리하는 OpenStack, CloudStack, openNebula 등의 클라우드 운용 관리 소프트웨어에는 공통의 운영체제를 탑재한 시스템 구조로 인해 클라우드 컴퓨팅 서비스는 보안의 취약함과 무거운 실행 이미지 그리고 이로 인한 성능 저하의 문제에 직면하고 있다.

현재 클라우드 컴퓨팅 서비스는 범용의 운영체제를 기반으로 하고 있으며 수많은 소프트웨어 스택이 탑재되어 있다. 이로 인해 상당한 자원을 소비할 뿐 아니라 속도가 느려지며 공격 목표가 내포된 채로 응용 프로그램이 배포된다. 이를 보안하기 위한 방법으로는 해당 응용이 필요한 자원만을 사용할 수 있도록 최적화하여 자원의 사용 공간을 줄이고 이를 통해 공격이 되는 영역을 줄이는 것이다.

이러한 필요성 때문에 최근에는 가상머신에 비해 작업 부하가 작고 빠른 도커와 컨테이너가 결합하여 클라우드 컴퓨팅에 활용되고 있다. 그러나 이 또한 공유자원으로 인한 문제를 포함하고 있으며, 이에 따라 응용에 적합한 최적의 라이브러리만을 포함하여 가볍고 매우 빠르게 구동되며 보안에 유용한 유니커널이 더 적합할 수 있다.



〈자료〉 ⓒ ETRI

[그림 1] 가상화, 도커와 컨테이너, 유니커널의 구조

유니커널은 일반적으로 공유 라이브러리와 리눅스와 같은 공용 운영체제가 별도로 필요하지 않으나 개발 및 관리와 실행의 편리성으로 하이퍼바이저와 호스트 운영체제를 포함하는 구조도 가능하다. [그림 1]은 가상화, 도커와 컨테이너, 유니커널의 구조를 보여준다.

1. 도커와 컨테이너

전형적인 클라우드 컴퓨팅에서는 서비스에 해당하는 응용을 가상머신에 배치하여 호스트 시스템의 자원 경합을 최소화하고 비교적 독립적인 실행환경을 유지한다. 그러나 가상머신 자체도 보통 그 크기가 기가바이트급으로 작지 않고 이식 및 지속적인 소프트웨어 갱신 등 유지 관리가 쉽지 않다.

최근에는 운영체제 커널을 공유하는 작고 가벼운 실행 환경으로 응용을 분리하여 가상머신보다 자원을 훨씬 작게 사용하고 즉시 시작할 수 있는 컨테이너를 활용하여 훨씬 적은 노력과 비용으로 확장할 수 있게 되었다. [그림 1]과 같이 가상머신이 하드웨어 가상화를 제공하여 응용의 실행 환경을 분리하는 것과는 달리 컨테이너는 사용자 공간의 추상화를 통해 운영체제 수준의 가상화를 제공한다.

또한, 컨테이너를 단일 프로세스로 실행되도록 제한하는 도커는 컨테이너 기반의 가상화 플랫폼이다. 도커는 운영체제 위에 탑재되어 커널이 지원하는 기능으로 컨테이너를 생성하고 관리한다. 이러한 도커용 운영체제는 주로 경량의 운영체제를 탑재하여 최적화된 구조를 가진다.

CoreOS는 리눅스 커널 기반의 도커에 특화된 경량 운영체제이다. 응용 소프트웨어를 배치하고

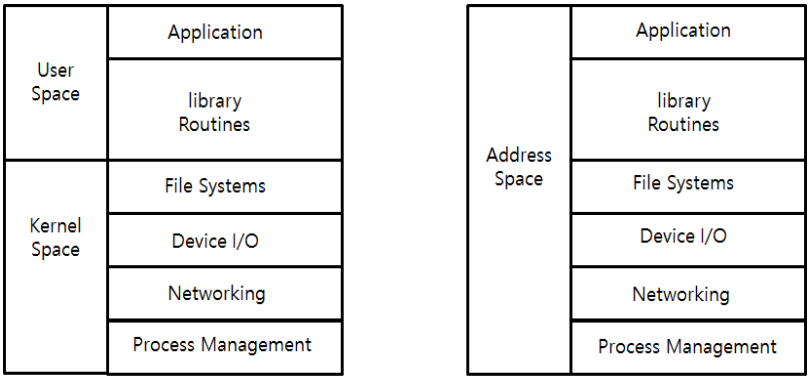
자동화를 지향한 클러스터 서비스를 관리하기 위한 운영체제이다. 일반 리눅스에 비해 40% 적은 OS 메모리를 사용하며 도커 관리에 최적화된 구조를 갖는다. 또한, 웹 인터페이스를 통한 모니터링과 확장성 있는 클러스터 관리에 적합하다. OS 업데이트 시에 운영 중인 서비스에 영향을 받지 않도록 2개의 부트 파티션을 가지고 있다. 마이크로소프트의 Azure와 아마존의 AWS(Amazon Web Services)를 위한 운영체제도 배포하고 있다[2].

대표적인 경량 운영체제로 특정 응용의 요구사항에 맞게 운영체제를 최적화한 소프트웨어 어플라이언스 형태의 패러다임으로 JeOS(Just enough operating system)이 있다. 플랫폼에는 어플라이언스에 포함된 특정 응용 프로그램과 이를 지원하는 최적화된 운영체제만 포함된다. 이로 인해 특정 응용 프로그램을 부팅하고 실행하기 위한 어플라이언스는 작고 빠르며 범용 운영체제에서 실행되는 응용 프로그램보다 더 안전하다. 대표적으로 썬에서 개발한 OpenSolaris JeOS와 Ubuntu JeOS가 이에 해당하며 가상화 환경에서 동작한다[3].

도커와 컨테이너가 결합하면서 사용이 용이하여 클라우드 컴퓨팅에서는 활발하게 활용되고 있다. 컨테이너는 반드시 호스트 시스템과 운영체제 커널을 공유해야하므로 커널 및 지원되는 프로세스가 이미 호스트에 있어 컨테이너 시작은 아주 빠른 편이나 보안에 관해서는 문제를 내포하고 있다. 기존의 공격 경로를 차단할 수 있는 대안이 필요하다.

2. 유니커널

컨테이너는 호스트 운영체제 커널을 공유하므로 커널코드가 실행하는 메모리 영역이 공유되는

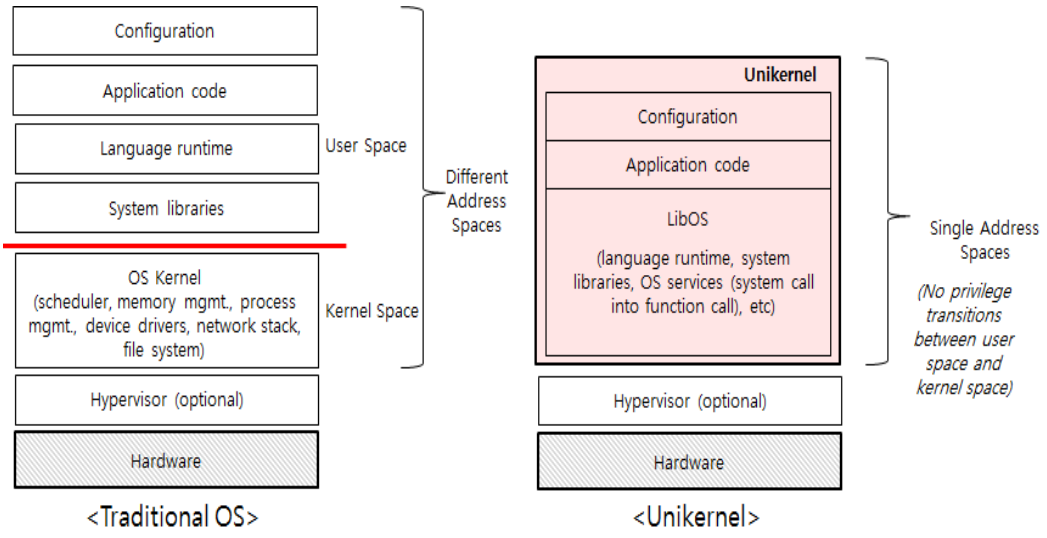


〈자료〉 © O'Reilly Media

[그림 2] 범용 시스템 기반 응용과 유니커널 시스템 응용의 소프트웨어 스택

문제가 있어 응용에 보안이 취약한 점이 발생할 수 있다. 이에 반해 유니커널은 개별 응용만의 독립적인 커널을 갖고 있어 보안에 유리하다[4]. 유니커널은 라이브러리 운영체제로 구현된 단일 주소 공간을 가진 이미지이다. 즉, 응용은 필요한 운영체제의 기능을 라이브러리 형태로 포함하여 그 이미지 자체가 해당 응용만의 특별한 커널이 된다. [그림 2]는 사용자 공간과 커널 공간이 분리된 범용 운영체제 응용의 소프트웨어 스택과 사용자와 커널 공간이 단일 주소 공간인 유니커널에서의 소프트웨어 스택이다[5],[6].

[그림 3]과 같이 유니커널은 응용과 커널의 기능을 내포한 완전히 독립적인 프로그램 환경이다. 클라우드 시대에 데이터는 이전에는 상상할 수 없었던 크기가 되었고 또한 해당 데이터에 대한 요청에 접근하고 처리하는 데 필요한 물리적 서버의 수가 엄청나게 커졌다. 이러한 환경에서 정보 요구량을 줄일 수는 없으므로 작고 빠른 실행과 보안에 비교적 안전한 유니커널 방안이 클라우드 컴퓨팅의 근본적 해결방안이 될 수 있다.



〈자료〉 ⓒ ETRI

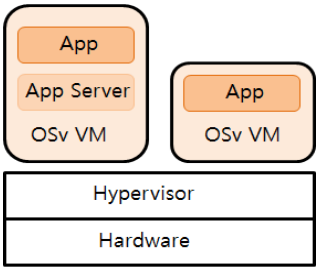
[그림 3] 범용 운영체제와 유니커널 스택

유니커널은 응용 프로그램을 실행시키는데 필요한 기능만을 포함하여 일반적으로 1메가바이트 미만의 작은 크기를 가진다. 이를 바탕으로 유니커널은 매우 빨리 시작할 수 있다. 또한, 악의적인 해커가 자주 이용하는 도구들을 포함할 여지가 없어 단일 크기 이미지의 공격 부분이 작아 보안을 크게 향상시킨다[7]. 이러한 특성들로 인해 데이터센터의 차세대 클라우드 환경에 있어 충분한 대안이 될 수 있다.

III. 클라우드 컴퓨팅 유니커널 기술 동향

1. OSv

OSv는 Cloudius System에서 만든 단일 프로세스를 지원하는 [그림 4]의 스택을 갖는 유니커널 솔루션으로 클라우드 가상머신에 특화되어 있다. 여러 시스템을 지원하는 기존 운영체제와 비교했을 때 성능 향상과 REST API를 통한 쉬운 관리 기능을 제공한다. 또한, 새로운 API를 가지고 있지만 Linux ABI를 지원하므로 수정되지 않은 Linux 응용 역시 대부분 실행하며, 특히 수정되지 않은 JVM 및 클라우드 서비스 응용을 실행할 수 있다. 클라우드에서 사용하는 응용 프로그램을 유니커널 인스턴스로 만들어 제공한다. 다양한 서비스를 지원하기 위해 KVM, Xen, Virtual box, VMware 등 여러 하이퍼바이저를 지원하며 다중 쓰레드를 허용한다. OSv가 주로 사용되는 클라우드 서비스 분야는 [표 1]과 같다[8].



<OSv cloud stack>

〈자료〉 ⓒ OSv

[그림 4] OSv 유니커널 클라우드 스택

[표 1] OSv 클라우드 서비스 사용 분야

분야	설명
가상 어플라이언스	패키지 응용 프로그램을 가상 컴퓨터 이미지로 제공하는 ISV(Independent Software Vendor)로 유용하다. ISV는 다른 플랫폼에서 요구하는 대규모 소프트웨어 및 구성 집합을 유지 관리하고 지원할 필요가 없다.
네트워크 함수 가상화	네트워크 장치를 가상화하려면 낮은 대기 시간과 높은 네트워크 처리량이 필요하다. OSv는 네트워크 채널 기반의 네트워크 스택을 사용하여 게스트 OS 수준에서 병목 현상을 제거한다.
자바 애플리케이션 서버	사용자는 REST API를 통해 애플리케이션 WAR 파일을 업로드 할 수 있으며 추가 구성없이 애플리케이션이 실행된다.
C, C++ 기반 일반 응용	C 및 C++ 응용 프로그램이 Makefile 등 몇 가지 설정 파일을 변경하여 쉽게 이식된다. OSv는 Linux용으로 수정되지 않은 공유 라이브러리를 사용할 수 있다. 이러한 응용 프로그램이 포함된 이미지는 기존 게스트 OS보다 낮은 오버헤드로 필요에 따라 다운로드하고 배포할 수 있다.

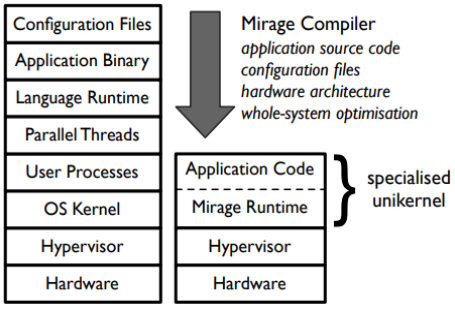
〈자료〉 ⓒ OSv

OSv 유니커널 역시 단일 응용과 커널이 하나의 실행이미지로 동작하므로 동일한 하드웨어 상의 다른 응용과는 데이터를 공유하는 메모리가 엄격히 분리되어 실행하기 때문에 보안에 유리하다. 또한 실행 이미지가 작고 최적화할 기회가 많아 전통적인 운영체제에 비해 성능 개선에 도움이 될 수 있다.

2. MirageOS

MirageOS는 클라우드 컴퓨팅 및 모바일 플랫폼에서 안전한 고성능 네트워크 애플리케이션을 위한 유니커널을 구성하는 라이브러리 운영체제로 OCaml 언어를 사용한다. 리눅스 파운데이션의 프로젝트 중 하나로 초기 유니커널의 개념을 정립하고 알리는데 많은 영향을 주었다. 레거시 응용을 지원하지 않으므로 MirageOS 라이브러리를 활용하여 응용을 새롭게 구현해야 한다. Linux, MacOS X와 같은 일반 OS에서 개발한 다음 Xen, KVM 하이퍼바이저에서 실행되는 단일 커널로 컴파일 할 수 있다. [그림 5]는 가상머신 어플라이언스 소프트웨어 스택과 유니커널 형태의 Mirage OS 소프트웨어 스택이다[9].

필요한 운영체제 기능들을 새롭게 구현하였기 때문에 코드 크기가 매우 작고 프로그램 언어의 특징을 반영할 수 있다는 장점을 가진다. 반면, 응용 프로그램을 작성하는데 있어서는 구현 언어 및 기능에 대해 잘 알아야 될 뿐만 아니라, 특정한 커널 기능을 따로 추가해야 하는 등 어려움이 있다. 현재 안정적으로 사용 가능하며 응용은 커널인 Mirage 런타임과 하나의 이미지로 컴파일되어 동일한 주소공간에서 실행된다. MirageOS의 응용은 실행에 필요한 라이브러리로만 구성되며, 커널과 응용 프로그램이 같은 주소 공간을 사용하기 때문에 문맥교환 시 추가적인 비용이 발생하지 않으므로 빠르게 실행할 수 있다.

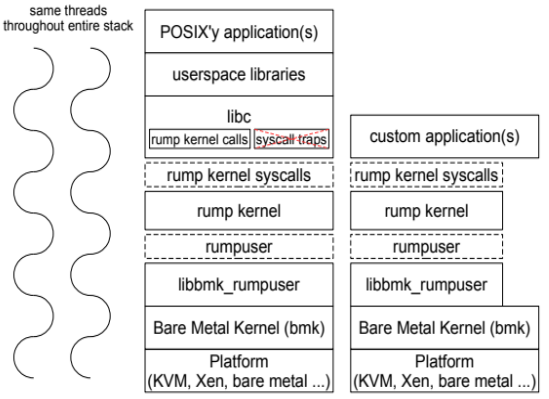


〈자료〉 © MirageOS

[그림 5] 가상머신 어플라이언스 소프트웨어 스택과 MirageOS 유니커널 소프트웨어 스택

3. Rumprun

Rump 커널을 기반으로 하는 Rumprun 유니커널은 메어메탈 뿐만 아니라 Xen, KVM 등 여러 하이퍼바이저를 지원하며 커널 수준의 드라이버도 지원한다. Rumprun은 [그림 6]과 같이 두 가지 모드를 지원한다. 왼쪽 모드는 레거시 응용을 지원하기 위해 프로그램을 수정하지 않고 실행할 수 있다. 오른쪽 모드는 사용자 프로그램에서 rumprun 인터페이스를 사용하여 프로그램을 해야 한다.



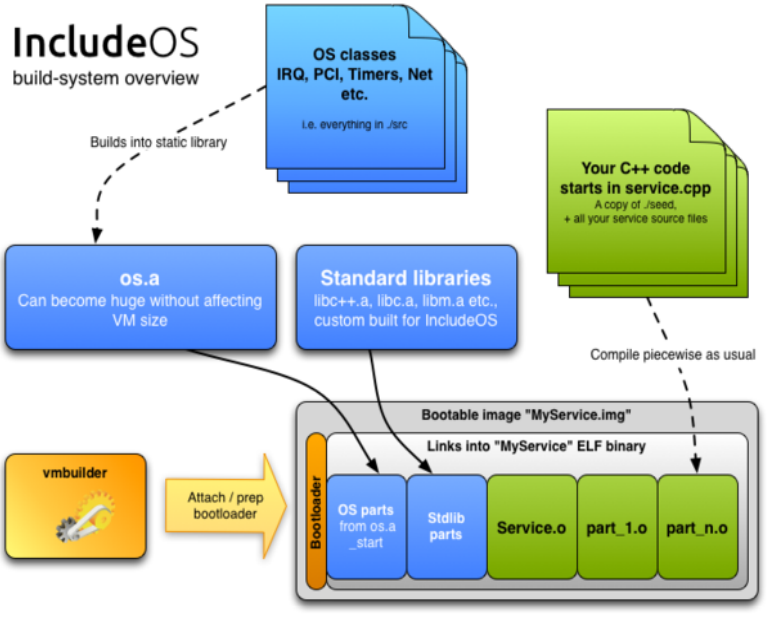
〈자료〉 ⓒ Rumprun

[그림 6] Rumprun 소프트웨어 스택

이 경우 응용 프로그램에서 OS 레이어를 직접 호출할 수 있어 더 빠르고 가볍다[10].

4. IncludeOS

대학 연구 프로젝트로 시작된 IncludeOS는 클라우드 환경에서 C++ 코드의 운영이 가능하며



〈자료〉 ⓒ IncludeOS

[그림 7] IncludeOS application 절차

서비스 오리엔티드 클라우드 서비스에 적합하다. KVM/QEMU와 VirtualBox에서 동작하며 C++로 작성한 응용은 [그림 7]과 같이 구축되어 부트 가능한 이미지로 컴파일되어 유니커널로 동작한다. 이를 통해 기존에 빌드된 응용 프로그램의 이미지를 바로 실행할 수 있다는 장점을 갖는다. 클라우드 서비스를 위한 최소한의 기능만을 제공하는 라이브러리 운영체제이다[11],[12].

IV. 결론 및 시사점

클라우드 컴퓨팅과 가상화의 인기에도 불구하고 여전히 인프라 기술을 최대한 활용할 방법을 찾고 있다. 일반적인 클라우드 컴퓨팅 환경에서는 여러 다른 응용 프로그램이 같은 운영체제 상에서 동작하는 경우가 많으므로 커널코드의 영역이 공유되는 문제가 있어 보안에 취약한 점이 발생할 수 있다. 이러한 이유로 클라우드 컴퓨팅에 활용되는 컨테이너 역시 게스트 응용 프로그램의 파티셔닝으로 제공되지만 호스트 운영체제를 공유하므로 보안 측면에서 안심할 수 없다.

유니커널은 작고, 빠르고, 안전하게 실행할 수 있다. 따라서 단일 응용과 커널을 동일 주소공간에 두어 다른 응용과의 완전한 격리를 제공하는 유니커널이 클라우드 환경에서 좋은 대안이 될 수 있다. 하지만 이 또한 대다수의 레거시 소프트웨어를 수정하지 않으면 실행이 어려워 POSIX API를 지원하는 등 상호보완이 되는 방안을 마련하고 있다. 향후 차세대 클라우드 컴퓨팅에서의 시스템 운영체제는 현재 활발하게 진행 중인 유니커널의 기술을 사용하여 많은 데이터센터에서 더 많은 시스템 자원 용량을 재확보할 수 있을 것으로 기대한다.

[참고문헌]

[1] 차승준 외, “유니커널의 동향과 매니코어 시스템에 적용,” ETRI, 전자통신동향분석, 33권 6호, 2018.
[2] Wikipedia, “Container Linux by CoreOS,” 2018, https://en.wikipedia.org/wiki/Container_Linux_by_CoreOS
[3] Wikipedia, “Just enough operating system,” 2018, hhttps://en.wikipedia.org/wiki/Just_enough_operating_system
[4] Lars Kurth, “Why Unikernels Can Improve Internet Security,” 2015, https://www.linux.com/news/why-unikernels-can-improve-internet-security
[5] Russell Pavlicek, “Unikernels,” O’Reilly Media, Inc., 2016.
[6] Wikipedia, “Unikernels,” 2018, https://en.wikipedia.org/wiki/Unikernel
[7] Anil Madhavapeddy. “Unikernels-Rethinking Cloud Infrastructure,” 2017, http://unikernel.org/
[8] A. Kivity et al., “OSv-Optimizing the Operating System for Virtual Machines,” USENIX Annual Technical Conference, 2014, pp.61-72.

[9] Anil Madhavapeddy et al., "Unikernels: Library Operating Systems for the Cloud," ACM SIGPLAN Notices, 2013, pp.461-472.
[10] Antti Kantee, "The Design and Implementation of the Anykernel and RUMP Kernels, 2nd Edition," 2016, http://book.rumpkernel.org/
[11] Serdar Yegulalp, "IncludeOS: Run cloud applications with less," InfoWorld, 2015.
[12] A. Bratterud et al., "IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services," IEEE 7th International Conference on Cloud Computing Technology and Science(CloudCom), 2015, pp.250-257.