Linux Containers Araştırma Raporu

Esra Çelik,

BULUT BİLİŞİM VE BÜYÜK VERİ ARAŞTIRMA LABORATUVARI (B3LAB)
TÜBİTAK - BİLGEM
celik.esra@tubitak.gov.tr

***Özet*— Bu çalışmanın amacı Bulut Bilişimde "Linux Containers" teknolojileri ile ilgili literatür araştırmaları sonucunda elde edilen bilgilerin kayıt altına alınmasıdır.**

***Index Terms*—Cloud, Linux Containers, LXC, Dockers**

I. *Giriş*

Günümüzde veri merkezlerinde yüksek işlem gücü ve yüksek depolama alanına sahip donanımlar çoklu kullanıcılar ya da uygulamalar tarafından aynı anda kullanılmaktadır. Bu durum kaynak paylaşımı ve soyutlaması problemlerini beraberinde getirmektedir. Bulut Bilişim uygulamalarının konusu olan bu soyutlama genellikle sanal makine kullanımı ile aşılmaktadır [2]. Sanal makina kullanımı çok katmanda soyutlamayı getirdiği gibi, bu katmanlardaki artış performans kayıplarına sebebiyet vermektedir [1]. Container tabanlı sanallaştırma ise kaynak kullanımında soyutlamayı, daha hafif (lightweight) bir yöntemle çözmektedir.

Sanal makinelerin tüm bir işletim sistemini barındırma gereklilikleri, çoğu zaman soyutlamanın yapılma amacını aşmaktadir. Örneğin, PaaS katmanında hizmet sunacak uygulamaların işletim sistemi üzerinde soyutlaması container'lar ile yapılabilmektedir [2].

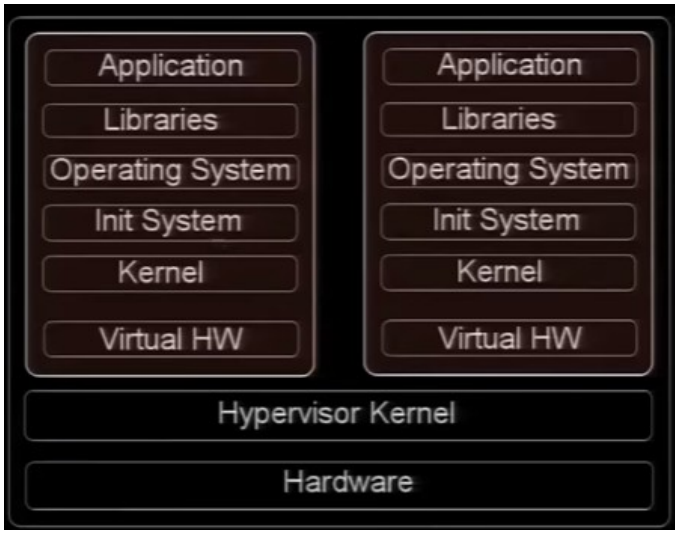
Ancak container'lar Bulut Bilişimde sanal makinaların kullanımına tam bir alternatif değildir, getirdikleri ekstra güvenlik problemleri sebebiyle IaaS katmanında verilen hizmetlerin sanal makinalar ile devam etmesi halen daha avantajlı görünmektedir.

Bu çalışmada container tabanlı sanallaştırmanın Bulut Bilişimde kullanım alanlarından, avantaj ve dezavantajlarından bahsedilecektir.

II. Sanallaştırma

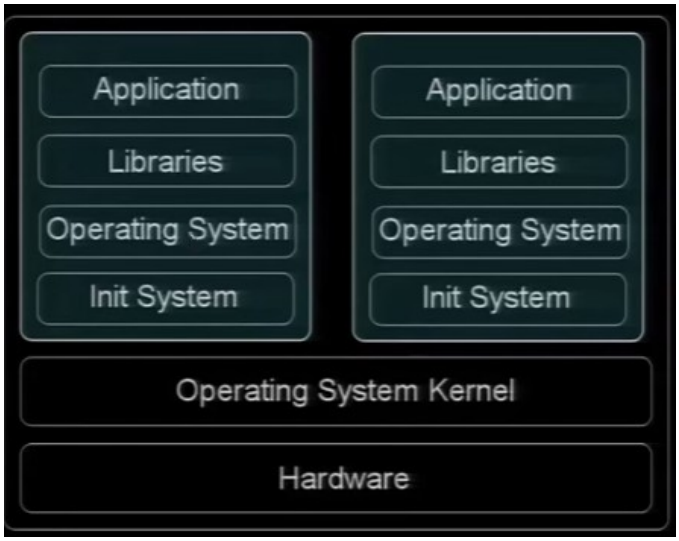
Bulut Bilişimde en yaygın kullanılan sanallaştırma yöntemi hipervizörler yardımıyla sanal makina kullanımıdır [1][2]. Ancak bu yöntemde Şekil 1'de görüldüğü gibi bir uygulamanın çalışması için pek çok katmanda fazladan kaynak kullanımına gidilmektedir.

Bir sanal makina donanım (baremetal) üzerinde çalışan hipervizör kernel ya da donanım üzerinde çalışan işletim sistemi üzerinde çalışabilmektedir. Sanal makinalar Bulut Bilişimde IaaS katmanında donanım tahsis ve yönetimi amacıyla kullanım için verimli bir yöntemdir [2]. Tek bir donanım üzerinde farklı kernele sahip işletim sistemlerinin yönetimi doğrudan kullanıcılara verilebilmektedir.



Şekil 1: Sanal makinalar [3]

Container'lar ise varolan bir işletim sistemi üzerinde gereksinimler doğrultusunda özelleştirme yaparak soyutlama sağlamaktadır. Sanal makina kullanımının tersine, bir container yalnızca bir process barındırabilecek kadar küçük olabilir. Aynı zamanda tüm bir işletim sistemi gibi davranan container'lar oluşturmak da mümkündür, bunlara "system container" adı verilir. Yaygın kullanımıyla bir uygulama barındıran container'lara ise "application container" denmektedir [1].

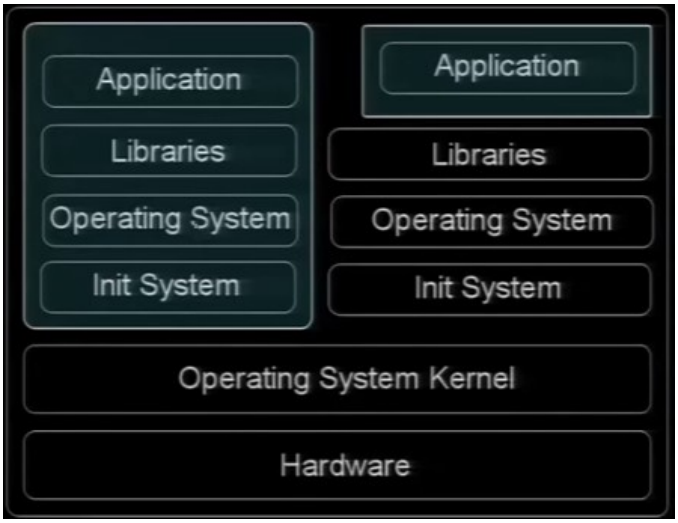


Şekil 2: System Containers [3]

Şekil 2'de bir "system container"a örnek verilmiştir. Bu örnekte aynı kernel üzerinde çalışan birden fazla işletim sistemi

container'lar yardımıyla aynı fiziksel makina üzerinde barındırılabilmektedir. Ancak böyle bir kullanım için container'lar üzerinde çalışacak işletim sistemlerinin aynı kernel'i desteklemeleri gerekmektedir.

PaaS katmanında container kullanımı ise donanım üzerinde çalışan işletim sisteminden sadece uygulama bazında soyutlamanın bile yeterli olabileceği durumları getirmektedir. Böylece kaynak paylaşımı artırılarak performans artımı sağlanmaktadır [1]. Şekil 3'de görülen uygulama soyutlaması container'lar ile yapılabilmektedir. Bu soyutlama kullanılacak kütüphaneleri [Libraries] de içerecek şekilde yapılabilir.



Şekil 3: Application Container [3]

III. Container'lar ile Soyutlama

Linux container'lar işletim sistemi üzerinde soyutlamayı sağlamak için bilindik Linux process kaynak tahsis yeteneklerini kullanmaktadır. Bunlar "namespace"ler ve "cgroup"lardır.

A. Namespaces

Linux namespace'ler işletim sistemi üzerinde çalışan bir process için görülebilir/ulaşılabilir nesnelerin sınırlandırılması için kullanılmaktadır [4]. Linux; filesystem, PID, network, user, IPC, hostname vs.. namespaceleri oluşturabilmektedir [3]. "clone()" sistem çağrısı ile kopyalanan global namespace, her bir container için birer altyapı oluşturmaktadır. Böylece örneğin, bir container içindeki filesystem namespace'in kendine ait root dizini ve mount tablosu olur, Bu, "chroot()" çağrısı gibidir, ancak daha güçlü bir mekanizma sunar [1].

B. cgroups (Control groups)

Kontrol grupları; kaynak kullanımının limitlenmesi, yönetilmesi ve hesaplandırılması için Linux kernel tarafından sağlanan bir özelliktir [5]. Her bir container için sistem kaynakları cgroups özelliği ile sağlanmaktadır. Ancak erişilebilir sistem kaynağının container içinde bilinememesi henüz çözülmemiş bir davranıştır. Örneğin bir container içinde çalışan bir process donanım üzerindeki tüm CPU'ları görebilmekte, fakat ne kadarının kendisi için tahsis edildiğini bilememektedir.

IV. Container'larda Güvenlik Problemleri

Namespace'lerin kullanımı sayesinde container'lar, erişim yetkileri olmayan nesneleri zaten görememektedir. Böylece kazara gerçekleştirilebilecek yetkisiz erişimler büyük ölçüde kısıtlanmaktadır. Container içinde root yetkilerine sahip bir kullanıcı, gerçek işletim sistemi üzerinde root olarak değerlendirilmemektedir.

Ancak namespace-farkındalığı olmayan sistem çağrıları container'lar için bilinen en büyük güvenlik açığını oluşturmaktadır. Açığı olan bir sistem çağrısı ile container içinden "exploit" yöntemiyle atak yapılarak ana işletim sistemi için tehdit oluşturulabilmektedir. Önlem olarak container içinden çağrılabilecek sistem çağrısı fonksiyonları "seccomp" ile sınırlandırılabilmektedir.

Bulut Bilişimde container kullanımı sırasında güvenlik açıklarından ve alınabilecek önlemlerden ileri düzeyde farkındalık önemlidir. Gerekli olan en alt düzey yetkilendirme, muhtemel atak alanlarını minimum düzeye indirecektir. Örneğin container teknolojisi; web uygulamaları, veritabanları ya da root yetkisi gerektirmeyen diğer uygulamalar için kullanılıyorsa; [11]

! Process üst yetkilendirmeyle çalıştırılmamalı
! SUID binary'leri temizlenmeli (ya da "nosuid mount" yapılmalı)
! Kernel güncel tutulmalı
! Ekstra güvenlik katmanları eklenmelidir.

"syscall" atak alanını "seccomp" özelliği ile kısıtlamak için;

! Mevcut "syscall" fonksiyonları limitlenmeli
! "syscall" argümanları limitlenmeli
! Limitli bir subset kullanımına geçilmelidir.

Fakat bu önlemler aşağıdaki soruları beraberinde getirmektedir;

? Kullanışlı kalmaya devam ederek, ne kadar limit konabilir?
? Güvenliğin sağlanması için gerçekten ne kadarı limitlenmelidir?

V. Container'ların Avantajları

- ✓ Çok hafif olması (lightweight)
- ✓ Konuk işletim sistemi ihtiyacı olmaması
- ✓ Daha az CPU, RAM, Storage alanına ihtiyaç duyması
- ✓ Sanal makinalara kıyasla donanım başına daha çok sayıda container oluşturulabilmesi
- ✓ Hızlı başlatma/sonlandırma yeteneği
- ✓ Anlık dikey genişleme yeteneği
- ✓ Küçük boyut sayesinde kolay taşınabilirlik (anlık yatay genişleme)
- ✓ Non-multitenant uygulamaların anlık olarak multi-tenant hizmet verecek hale getirilebilmesi [3]

VI. Container'ların Dezavantajları

- ✓ Ev sahibiden farklı bir kernele sahip işletim sistemi çalıştıramamaları
- ✓ Güvenlik açıkları
- ✓ Container içinden tahsis edilen sistem kaynağının bilinememesi

VII. Bulut Bilişimde Container'lar

Bulut teknolojilerinde container'lar farklı amaçlar için kullanılabilmektedir. Örneğin kolay yönetilebilirlik ve bağımsızlık için OpenStack servisleri container'lar üzerine kurulabilmektedir [10]. Diğer bir kullanım, PaaS servisi olarak uygulamaların container'lar üzerinde çalıştırılarak kiralanmasıdır. Burada en büyük avantaj, uygulamanın kendisi multi-tenant olmasa bile container'lar sayesinde multi-tenant hale getirilebilmesidir. IaaS katmanında ise container kullanımı günümüzde mevcuttur ancak henüz yaygınlığa kavuşmamıştır [3]. Henüz mevcut Openstack Nova container sürücüleri yeterli fonksiyonelliği sunmamaktadır.
Bu konuda sunulan bazı çözümler aşağıdadır:

- libvirt for LXC
- Nova-docker
- Heat resource for Docker
- libct (by Odin)
- libcontainer (by Docker)

Sonuç

Container'lar ihtiyaç duydukları düşük kaynak kullanımı ve hızlı başlatma/sonlandırma özellikleri nedeniyle oldukça kullanışlıdır. Güvenlik açığı problemlerinin çözülmesi ya da izole edilmesi ile pek çok alanda kullanılabilirler. Bulut Bilişim teknolojilerinde de kullanımının yaygınlaştırılması, gelişen teknolojinin takip edilmesi ve gelişimine katkıda bulunulması önem arzetmektedir.

Kaynaklar

[1] An updated performance comparison of virtual machines and Linux containers, IBM Research Report, July 21, 2014
[2] Claud Pahl, Containerization and the PaaS Cloud, IEEE Cloud Computing, 2015
[3] James Bottomley (CTO of Server Virtualization at Parallels), The Future of Containers in Linux and OpenStack, May 2014
[4]http://man7.org/linux/man-pages/man7/namespaces.7.html
[5] https://wiki.archlinux.org/index.php/Cgroups
[6] Magnum - Containers-as-a-Service for OpenStack, OpenStack Summit Vancouver, May, 2015
[7] Ask the Experts: Are Containers a Threat to OpenStack?, OpenStack Summit Vancouver, May, 2015
[8] Developing the next generation of Containerised applications with libcontainer, OpenStack Summit Vancouver, May, 2015
[9] Using Docker with OpenStack - Hands On!, OpenStack Summit Vancouver, May, 2015
[10] http://docs.openstack.org/developer/openstack-ansible/install-guide/overview-osa.html
[11] Jérôme Petazzoni (Working at Docker Inc.), Linux Containers, Docker and Security, Jan 31, 2014
[12] https://docs.docker.com/engine/articles/security/