Sanal Makineler ve Linux Konteynerlerin Performans Karşılaştırması

Emre Can YILMAZ[1], Recai OKTAŞ[2]

[1]Ondokuz Mayıs Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Samsun

[2]Ondokuz Mayıs Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Samsun

ecylmz@bil.omu.edu.tr, roktas@bil.omu.edu.tr

Özet

- Günümüzde, bulut bilişimle birlikte yaygınlaşan tam sanallaştırmadan, hızla popülerliği artan konteyner tabanlı sanallaştırmaya geçişin olduğu gözlemlenmektedir.

- Bu geçişin nedenleri arasında tam sanallaştırmada kaynakların verimsiz kullanılması ve bunun sonucunda sanal makinelerin iş yükü performansının düşük olması gösterilmektedir.

Özet

- Bu çalışmada, Linux işletim sistemi çekirdeği özelinde tam sanallaştırma ile konteyner tabanlı sanallaştırmanın başarım kıyaslamaları yapılmıştır.

- Son üç yıl içerisinde, Linux konteynerleri için, standart bir çalışma zamanı olan, imaj formatı ve inşa sistemi olarak Docker ortaya çıkmıştır.

- Yapılan çalışmalar için Linux konteyner yönetim yazılımı olarak Docker ve son zamanlarda Docker’a alternatif olarak ortaya çıkan Rocket kullanırken, tam sanallaştırmada hipervizör olarak KVM (Kernel based Virtual Machine) kullanılmıştır.

Kernel Virtual Machine (KVM)

- Linux süreci içerisinde işletim sistemi çalıştırmayı sağlayan araçtır.
- KVM donanım sanallaştırması teknolojisini kullanarak, karmaşıklığı ve maliyeti düşürmektedir.
- QEMU aracılığıyla da işletim sistemi taklidini yapmaktadır.
- Sanal makinelerin kaynak yönetiminde yeterli verim alınamamaktadır.

Linux Konteynerler

- Sanal donanımlar üzerinde çalışan işletim sistemlerinin aksine, konteyner tabanlı sanallaştırma varolan işletim sistemini üzerinde çalışır. Bu konteynerleri sistemdeki diğer kullanıcılar yetkileri dahlinde görüntüleyebilir.

- Linux konteynerler, çekirdek isim uzayı özelliğini kullanır. Linux, dosya sistemi, PID, kullanıcı ve ağ isim uzaylarını gerçekler. Örnek verilecek olunursa, her dosya sistemi isim uzayı kendi kök dizinine sahiptir.

- Linux konteynerleri için pek çok yönetim aracı mevcuttur. Bunlardan başlıca olanları, LXC, systemd-nspawn, Docker ve Rocket’tir.

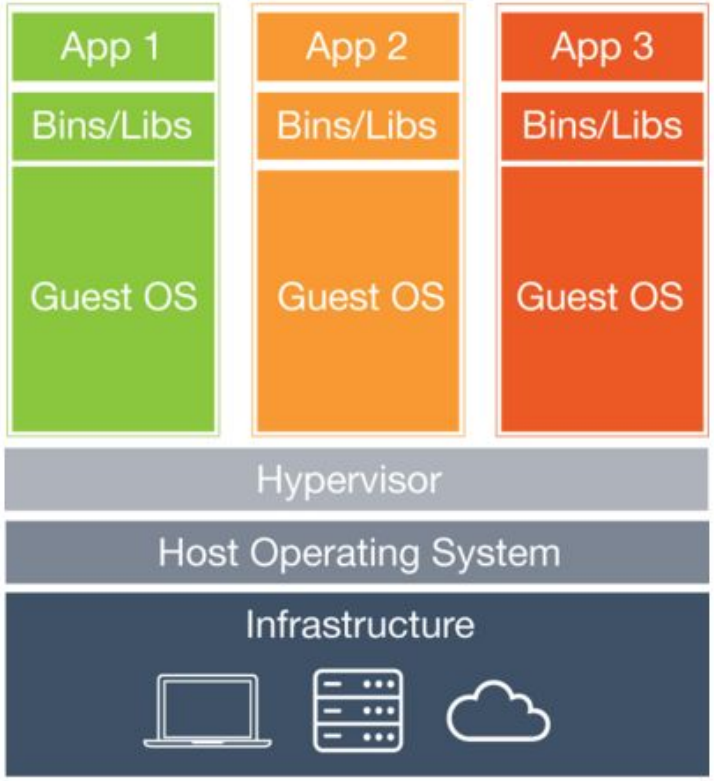
Docker ve Rocket

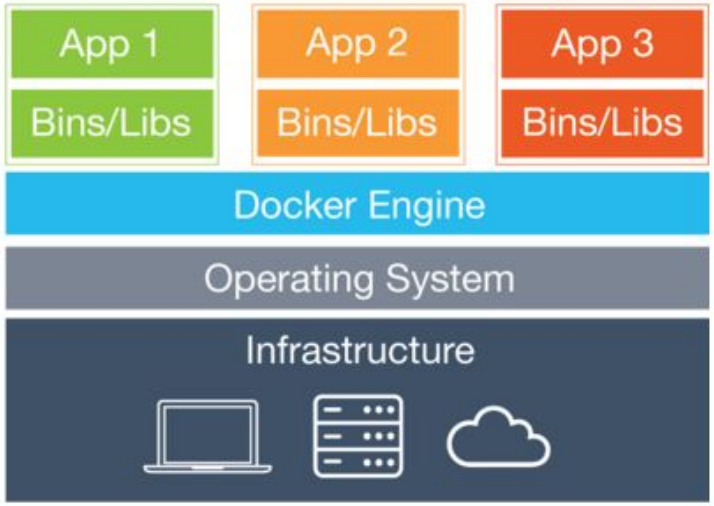
- Pek çok özelliklerinden ve kolay kullanımından dolayı Docker popülerliğini hızla artırmaktadır.

- Docker ile yaygınlaşan konteyner çözümlerine ek olarak Rocket ortaya çıkmıştır.

- Docker’dan farklı olarak güvenliğe önem vermesi ve uygulama konteynerlerini standart haline getirmek hedeflenmiştir.

- Bunun yanı sıra Docker’dan farklı olarak systemd-nspawn konteyner yönetim kütüphanesini kullanmaktadır.


App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
Infrastructure


App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System
Infrastructure

Değerlendirme

- Performans değerlendirmede pek çok yaklaşım vardır. Bu çalışmada, sanallaştırma teknolojileri ile fiziksel makinedeki(sanallaştırma olmayan) maksimum iş yükü altındaki durumlarını inceledik.

- Yapılan testlerin tamamı; Lenovo C560 57324249 modelli, Intel® Core™ i5-4570T CPU @ 2.90GHz × 4 işlemcili toplamda 4 çekirdeği bulunan, toplamda 12 GB RAM'e sahip bilgisayar üzerinde gerçekleştirildi. Bu sunucu üzerinde Ubuntu 14.04(Trusty) 64 bit, 3.13.0-62 Linux çekirdeği, Docker 1.8.2, QEMU 2.0.0, libvirt 1.2.2, rkt 0.5.5 kullanılmıştır.

- Docker ve rkt konteynerlerine cgroups aracılığıyla herhangi bir sınırlama konulmamıştır. CPU, bellek, ağ ve depolama konularında mikro kıyaslama betikleri kullandık.

CPU

- Test için PXZ aracı kullanılmıştır.

- PXZ, LZMA algoritmasını kullanan, paralel ve kayıpsız veri sıkıştırma aracıdır. Bu çalışmada PXZ programının 4.999.9beta sürümü kullanılmıştır.

- 2. seviye sıkıştırma tekniği kullanıldı.

CPU

- Docker, Rocket ve Linux’ta sonuçlar çok yakınken, KVM %23 daha yavaştır.

Uygulanan Test	Linux	Docker	Rocket	KVM
PXZ (MB/s)	7,94	7,00	7,35	6,2

Yüksek Başarımlı Hesaplama

- “High Performance Computing” yani yüksek başarımlı hesaplama testidir.

- Çalışmada, Intel Math Kernel Library (MKL) tabanlı Linpack programı kullanıldı.

- Linpack yoğun lineer denklem sistemlerini LU faktorizasyonu algoritmasını kullanarak çözer.

Yüksek Başarımlı Hesaplama

Uygulanan Test	Linux	Docker	Rocket	KVM
Linpack (GFLOPS)	26,94	26,86	25,93	18,49

- Sonuçlarda Linux, Docker ve Rocket'in performanslarının neredeyse aynı olduğu gözlemlenmektedir.
- Ancak KVM'deki durumun diğerlerine göre kötü olduğu gözükmektedir.
- Ortaya çıkan sonuçlar beklenen sonuçlardı.

Bellek Bant Genişliği

- Bellek bantı genişliği için “Stream” testi kullanılmıştır.

- Stream testi vektörler üzerinde basit işlemler yaparak bellek bant genişliği ölçümü yapmaktadır. Bant genişliği, bir iletim ortamının ya da haberleşme kanalının kapasitesini ifade etmek için kullanılır.

- Başka bir deyişle bir kanal üzerinde taşınabilecek en fazla frekansa sahip sinyal, kanalın bant genişliğidir.

- Test dört bileşenden oluşmaktadır (Add, Copy, Scale, Triad).

Bellek Bant Genişliği

Tüm ortamlarda neredeyse aynı sonuçlar elde edilmiştir.

Uygulanan Test		Linux	Docker	Rocket	KVM
Stream (GB/s)	Add	11.8211	11.8406	11.7503	11.5224
	Copy	11.4309	11.4464	11.1776	11.2618
	Scale	11.3928	11.4289	11.1505	11.2498
	Triad	11.7716	11.7792	11.7169	11.513

Rastgele Bellek Erişimi

- Stream testinde düzenli bir şekilde bellekten veri çekerek bu test uygulansa da Random Access testinde veriler bellekten rastgele seçilerek test gerçekleştirilir.

- Test, çalışma kümesi olarak önbelleğin veya TLB’nin erişemeyeceği kadar belleğin büyük bir kısmını seçerek başlar. Belekten 8 bitlik okuma düzenleme ve yeniden yazma işlemleri yapılır. Belleğe erişilirken kullanılan rastgele konumlar, bellek harcaması gerektirmeyen “linear feedback shift register” yöntemiyle belirlenmiştir.

Rastgele Bellek Erişimi

- Tüm platformlarda sonuçlar çok yakın çıkmıştır. Birbirlerinden bariz bir üstünlükleri bulunmamaktadır.

Uygulanan Test	Linux	Docker	Rocket	KVM
RandomAccess (GUPS)	0,0550	0,0515	0,0520	0,0545

Ağ Bant Genişliği

- Ağ bant genişliği testi için nuttcp aracını kullandık.
- Docker ve Rocket konteynerleri ağa NAT kullanarak bağlar.
- KVM yapılandırmasında virtio kullanılmıştır.
- Tüm platformlarda ulaşılan maksimum hız eşit çıkmıştır.

Ağ Bant Genişliği

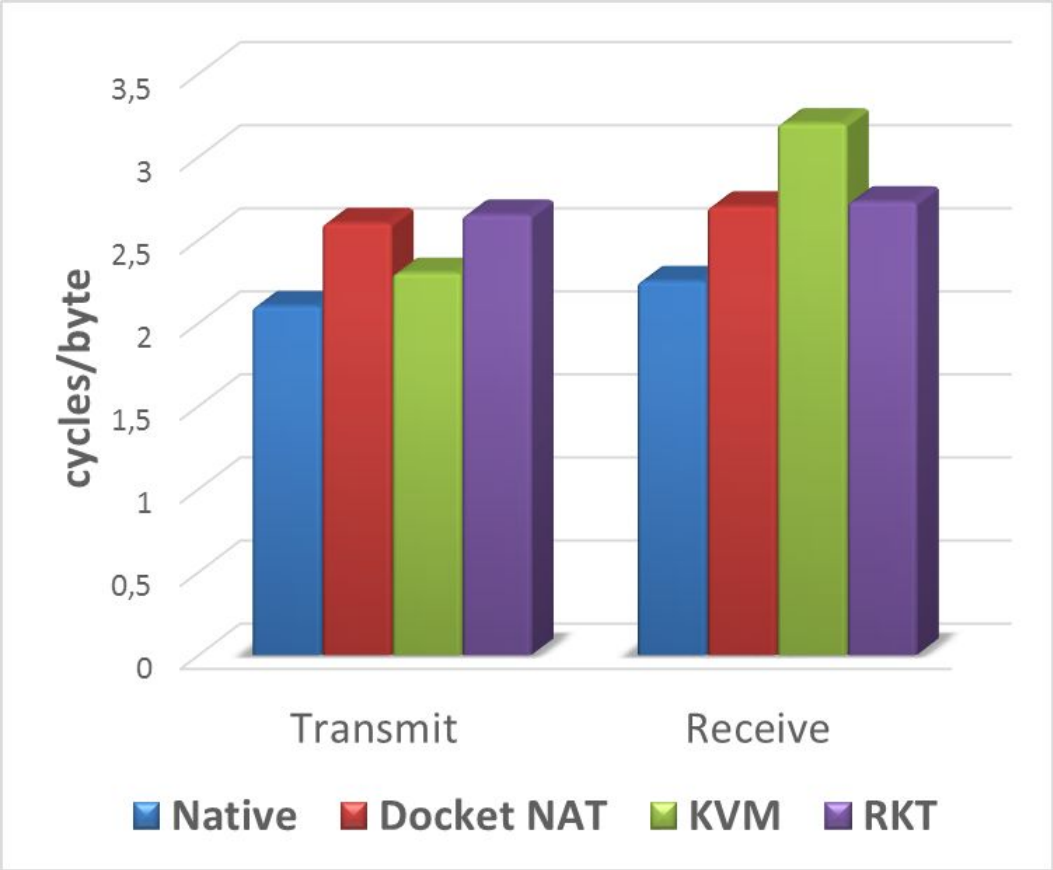
- Ölçümler sırasında CPU kullanım verimliliği sonuçları gözlemlenmiştir.

- Bu ölçümlerde “perf stat -a” kullanılmıştır.

- KVM veri iletimi işlemi sırasında daha verimli olurken, Docker’da önce köprü sonra NAT işlemi yapıldığından KVM’e göre verimi düşüktür.

- Ancak KVM veri alımı işlemi sırasında diğerlerine göre çok daha fazla işlem yükü oluşturmaktadır.

Ağ Bant Genişliği



TCP toplu transfer verimi (CPU cycles / byte)

Ağ Gecikmesi

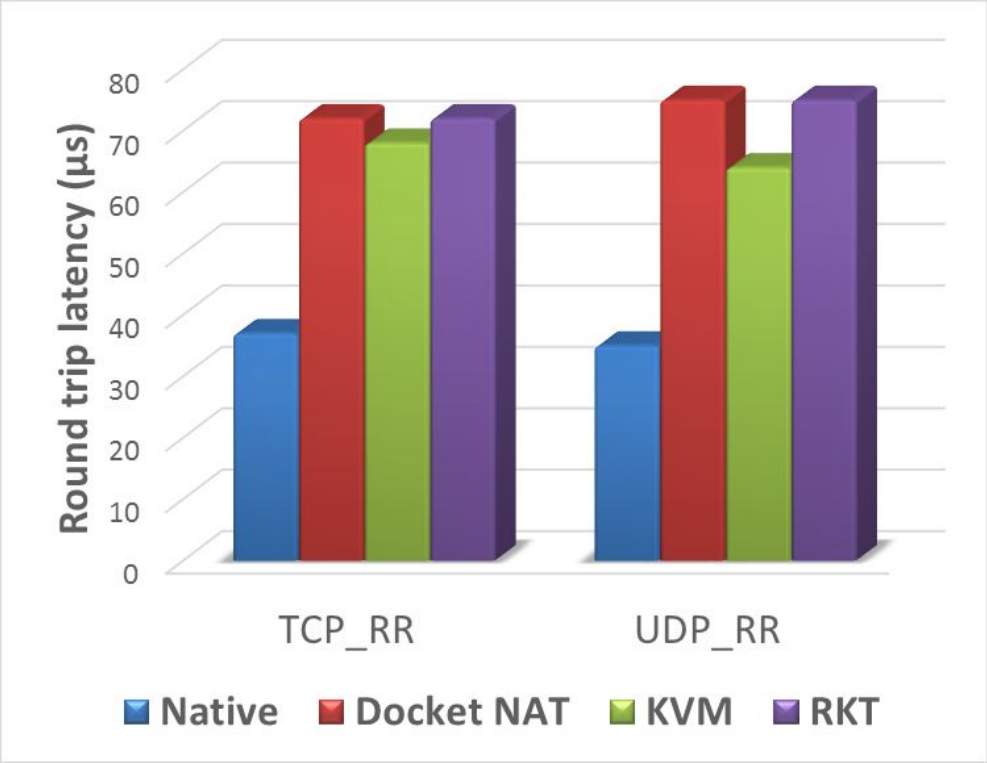
- Ağ gecikmesi tesi için “netperf” istek-yanıt aracını kullandık.

- Bu test sırasında sunucularda birinde netperf sunucusu(netserver) çalışırken, diğer sunucuda ise netperf istemcisi çalışır.

- İstemci 100 baytlık istekler gönderirken, sunucu 200 baytlık yanıtlar gönderirir. İstemci diğer isteği göndermeden önce sunucudan gelecek yanıtı bekler. Belirtilen bu adımlar 1 işlem olarak kabul edilir.

Ağ Gecikmesi

- Docker'da ve Rocket'da NAT kullanılması gecikmeyi yaklaşık 2 katına çıkarmıştır. KVM de ise sanallaştırma olmayan sunucuya göre yaklaşık %80 oranında daha fazladır.



Ağ round-trip gecikmesi (µs).

Blok I/O İşlemleri

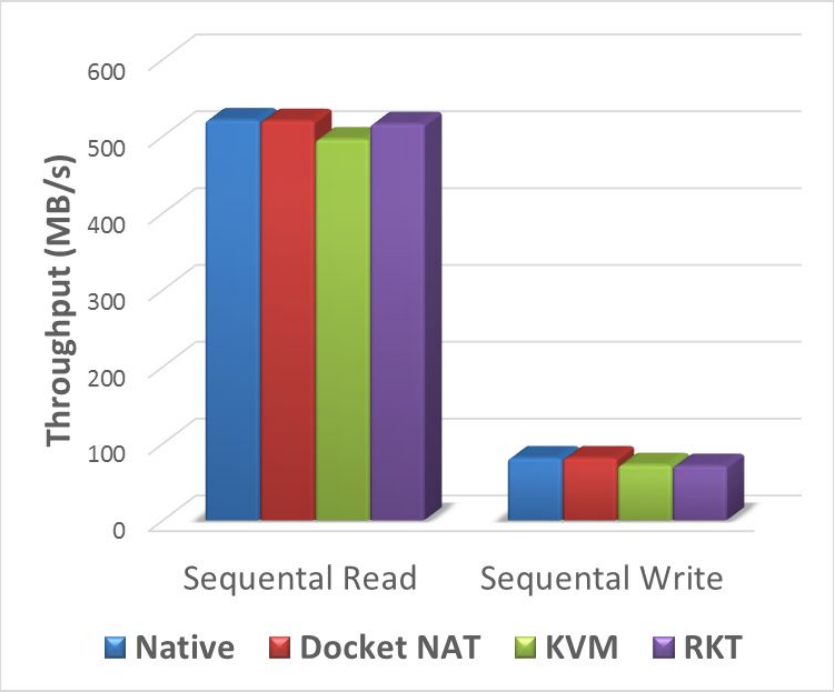
- Blok I/O işlemlerinde kullanmak üzere 120 GB SSD Samsung EVO 850 model disk kullandık.

- Dosya sistemi olarak ön tanımlı ayarlarıyla EXT4 dosya sistemi kullandık. Sanal makine, sisteme bağlanan diski kullanmak için “virtio”yu kullanmaktadır.

- Testi gerçekleştirmek için fio(v2.1.3) aracı kullanıldı.

Sıralı Okuma Yazma Performansı

- Sıralı okuma yazma performansları hemen hemen aynı olduğu gözlemlenmiştir.



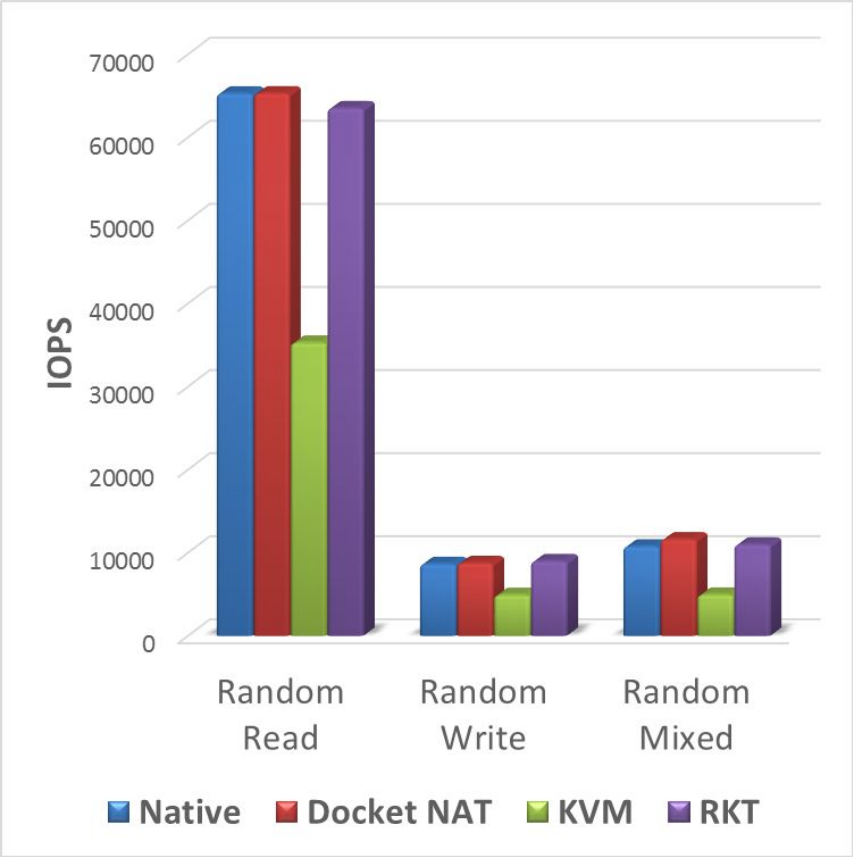
Sıralı I / O verimliliği (MB/s).

Rastgele Okuma Yazma Performansı

- 4kB’lık bloklar halinde rastgele okuma yazma performansları kıyaslanmıştır.

- Beklenildiği gibi Docker’da ve Rocket’de herhangi bir performans sorunu yokken, KVM her bir I/O işlemi için QEMU’ya başvurduğundan performans kaybı yaşamaktadır.

Rastgele Okuma Yazma Performansı



Rastgele I / O verimliliği (IOPS).

Sonuç ve Öneriler

- Donanım ve yazılımların son on yılda gelişmesiyle birlikte, Linux konteynerler ve sanal makineler olgunlaşmış teknoloji haline gelmişlerdir.

- Docker ve Rocket her açıdan sanal makinelerin performansına erişebildiği gözlemlenmiştir.

- Çalışmalarımız neticesinde KVM’in yoğun I/O işlemlerinde ağır kaldığı gözlemlenmiştir. Yüksek I/O işlemlerinin yapılacağı uygulamalar için çok da uygun olmadığı anlaşılmıştır.

Sonuç ve Öneriler

- Eğer sanal makineler üzerinde konteynerler çalıştırılmak istenirse, bu ekstra güvenlik katmanı yaratmasına karşın, QEMU'ya yetkisiz erişim sağlayan biri aynı zamanda konteyner içerisine de erişim sağlamaktadır.

- Bu çalışmada, tek bir sunucu üzerine sadece bir sanal makine veya sadece bir konteyner konulmuştur. Gerçek hayatta sunucular genellikle bu şekilde kullanılmaz.

Sonuç ve Öneriler

- Bulutta genellikle sağlayıcılar bu sunucuları daha küçük birimlere kullanarak müşterilerine sunar. Ayrıca bulut bilişimde kullanılan sunucuların bu çalışmada kullanılan sunuculardan çok daha üstün performansa sahip olması gerekmektedir.

- Bu çalışmada ortaya çıkan sayısal değerler, platformların birbirlerine göre oransal olarak kıyası söz konusudur.

Kaynak Kodlar

- Bu çalışmada çalıştırılan testler https://github.com/ecylmz/docker-rkt-kvm-comp/ adresinde bulunmaktadır.

Teşekkürler!

Emre Can YILMAZ

Ondokuz Mayıs Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Samsun

ecylmz@bil.omu.edu.tr