Uludağ Üniversitesi Mühendislik Fakültesi Dergisi, Cilt 25, Sayı 2, 2020 ARAŞTIRMA

DOI: 10.17482/uumfd.605560

BULUT ORTAMLARINDA HİPERVİZÖR VE KONTEYNER TİPİ SANALLAŞTIRMANIN FARKLI ÖZELLİKTE İŞ YÜKLERİNİN PERFORMANSINA ETKİSİNİN DEĞERLENDİRİLMESİ

Gökhan IŞIK [*]
Uğur GÜREL [*]
Ali Gökhan YAVUZ [**]

Alınma: 21.08.2019; düzeltme: 22.07.2020; kabul: 26.08.2020

Öz: Fiziksel kaynakların verimli kullanılabilmesini sağlayan sanallaştırma teknolojilerindeki ilerlemeler, bulut bilişim, nesnelerin interneti ve yazılım tanımlı ağ teknolojilerinin gelişiminde büyük pay sahibi olmuştur. Günümüzde hipervizör sanallaştırma çözümlerine alternatif olarak konteyner teknolojileri ortaya çıkmıştır. Bulut sağlayıcıları kullanıcılarına daha verimli ortamlar sunmak için hem hipervizör hem konteyner sanallaştırma çözümlerini destekleyen sistemleri tercih etmektedirler. Bu çalışmada, 'Alan bilgisine dayalı' metodoloji uygulanarak OpenStack bulut altyapısında işlemci, bellek, ağ ve disk yoğunluklu iş yüklerinin KVM hipervizör ve LXD konteyner sanallaştırma çözümleri üzerinde performans değerlendirmesi yapılmıştır. OpenStack bulut ortamında PerfKit Benchmarker ve Cloudbench kıyaslama otomasyon araçları vasıtasıyla ağ yoğunluklu iş yükü olarak Iperf, işlemci yoğunluklu iş yükü olarak HPL, bellek yoğunluklu iş yükü olarak Stream ve disk yoğunluklu iş yükü olarak Fio kıyaslama araçları kullanılarak performans testleri gerçekleştirilmiştir. Yapılan testler sonucunda OpenStack bulut altyapısında LXD sanallaştırma KVM sanallaştırmaya göre işlemci, ağ, sabit disk sürücüsünde sıralı okuma bant genişliği, saniyedeki giriş/çıkış sayısı ve sıralı yazma bant genişliği, saniyedeki giriş/çıkış sayısı iş yüklerinde daha iyi performans sergilemiştir. KVM sanallaştırma ise LXD sanallaştırmaya göre bellek, sabit disk sürücüsünde rasgele okuma bant genişliği, saniyedeki giriş/çıkış sayısında ve rasgele yazma bant genişliği, saniyedeki giriş/çıkış sayısında daha iyi performans sergilemiştir.

Anahtar Kelimeler: Sanallaştırma, bulut bilişim, hipervizör, konteyner, KVM, LXD

Evaluation of the Effect of Hypervisor and Container Type Virtualization on Different Workloads Performance in Cloud Environments

Abstract: Advances in virtualization technologies that provide efficient use of physical resources have played major role in development of technologies such as cloud computing, internet of things and software defined networks. Today, container technologies have emerged as alternative to hypervisor virtualization solutions. Cloud providers design a system that supports both hypervisor and container virtualization solutions to provide users with more efficient environments. In this study, performance evaluation of processor, memory, network and disk intensive workloads on KVM hypervisor and LXD container virtualization solutions in OpenStack cloud infrastructure has been performed by applying 'Domain Knowledge-based Methodology'. Performance tests were performed with PerfKit Benchmarker and Cloudbench benchmark automation tools by using Iperf as a network-intensive workload, HPL as a processor-intensive workload, Stream as a memory-intensive workload and Fio as a disk-intensive workload in the OpenStack cloud environment. As a result, LXD virtualization in OpenStack infrastructure

[*] Eskişehir Osmangazi Üniversitesi ESOGÜ Meşelik Yerleşkesi MMF Bilg. Müh. Bölümü, 26480 ESKİŞEHİR
[**] Yıldız Teknik Üniversitesi Davutpaşa Kampüsü Elektrik-Elektronik Fakültesi Bilg. Müh. Bölümü,İSTANBUL
İletişim Yazarı: Gökhan IŞIK (isik.gokhan27@gmail.com)

performed better than KVM in processor-intensive, network-intensive, sequential read bandwidth, sequential read input/output per second, sequential write bandwidth and sequential write input/output per second for HDD workloads. KVM virtualization in OpenStack infrastructure performed better than LXD in memory-intensive and random read bandwidth, random read input/output per second, random write bandwidth and random write input/output per second for HDD workloads.

Keywords: Virtualization, cloud computing, hypervisor, container, KVM, LXD

1. GİRİŞ VE AMAÇ

Bulut bilişim günümüzün gözde teknolojik alanlarındandır. Bulut bilişim sanallaştırma teknolojileri üzerine geliştirilmiştir. Farklı bulut altyapı çözümleri farklı sanallaştırma teknolojileri üzerine kurulmuştur. Bulut bilişim çözümleri, altyapılarında kullandığı farklı sanallaştırma teknolojilerinden dolayı işlemci, bellek, ağ ve disk yoğun uygulama tiplerine göre performans bakımından farklılık göstermektedir. Bu yüzden bulut altyapılarında en uygun performans elde edebilmek için uygulama tipi ile en uygun sanallaştırma çözümünün seçimi kurum ve kuruluşlar için çok önemli hale gelmiştir.

Bu çalışma kapsamında bulut bilişim çözümü olarak OpenStack tercih edilmiştir. OpenStack bulut bilişim çözümü KVM (Kernel-based Virtual Machine), LXC (Linux Container), VMware ve LXD (Linux Container Daemon) gibi birçok sanallaştırma çözümünü desteklemektedir. Bu çalışmada OpenStack bulut altyapısının tercih edilmesindeki en önemli etken birçok sanallaştırma teknolojisini destekleyen karma bir yapıda olmasıdır.

KVM ve LXD günümüzde yaygın olarak kullanılan ve tercih edilen sanallaştırma teknolojileri olmasından dolayı bu çalışma kapsamında tercih edilmiştir. Ayrıca KVM ve LXD yapısal olarak birbirinden farklı sanallaştırma çözümleridir. KVM hipervizör tipi, LXD ise konteyner tipi bir sanallaştırma çözümüdür.

Kurumlar veya uygulama geliştiriciler bulut bilişim çözümlerinin sunduğu ölçeklenebilirlik, yüksek erişilebilirlik, esneklik ve verimlilik özelliklerden faydalanarak uygulamalarını bulut bilişim altyapılarında barındırmayı hedeflemektedirler. Bu durumda da sanallaştırma teknolojilerinin performans değerlendirilmesi yapılırken sanallaştırma teknolojisinin kendi mimarisinin yanı sıra bulut bilişim çözümü ile nasıl bütünleştirildiği de göz önünde bulundurulmalıdır.

Bu çalışmada OpenStack bulut altyapısında KVM ve LXD sanallaştırma teknolojilerinin performans değerlendirilmesi işlemci, bellek, ağ ve disk yoğun iş yükleriyle yapılmıştır. İş yüklerinin oluşturulmasında ve test edilmesinde Cloudbench ve PerfKit Benchmarker araçları tercih edilmiştir. Testler PerfKit Benchmarker aracı ile 15 defa çalıştırılıp sonuçlar alınmıştır. Bu sonuçların ortalama değerleri temel alınarak karşılaştırma yapılmıştır. Ayrıca PerfKit Benchmarker aracı ile gerçekleşen testlerin doğruluğu Cloudbench aracı ile kontrol edilmiştir.

Bu çalışma kapsamında aşağıdaki katkılar sağlanmıştır.

- KVM ve LXD sanallaştırma çözümlerinin farklı iş yükleri için performans değerlendirmesi bulut bilişim çözümü olan OpenStack altyapısında gerçekleştirilmiştir.
- Mevcut uygulamaların bulut bilişim altyapısına taşınmasında uygulama tipine göre en uygun sanallaştırma çözümünün ne olacağı ortaya konulmuştur.

Bu makalenin ikinci bölümünde sanallaştırma, konteyner ve bulut bilişim teknolojileri hakkında genel bilgi verilmiştir. Üçüncü bölümde literatürde konu ile ilgili mevcut çalışmalardan bahsedilmiştir. Dördüncü bölümde deneyde kullanılan kıyaslama araçları açıklanmıştır. Beşinci bölümde çalışmada takip edilen metot, adımlarıyla birlikte açıklanmıştır. Altıncı bölümde deneyin tasarlanması ve uygulanması ve yedinci bölümde ise deney sonucunda ortaya çıkan bulgular ve bunların sebepleri ortaya konulmuştur. Sekizinci bölüm ise makalenin sonuç ve öneri bölümüdür.

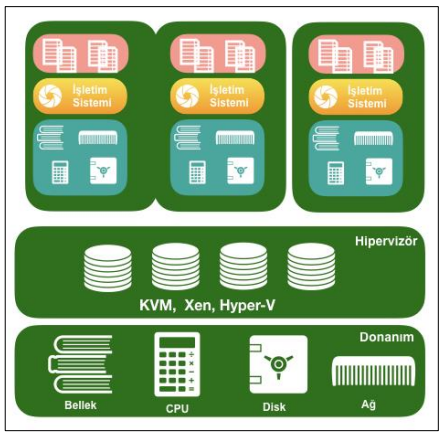
2. ÖN BİLGİ

Bu bölümde sanallaştırma, KVM, konteyner, Linux isim uzayları (Linux Namespaces), Kontrol Grupları (cgroups), LXC, LXD, bulut bilişim ve OpenStack konusunda temel bilgiler verilmiştir.

2.1 Sanallaştırma

Sanallaştırma bir fiziksel donanım üzerinde birden fazla sanal sunucu makinası veya uygulama çalıştırılmasını sağlayan yazılımsal bir teknolojidir. Fiziksel donanımların daha verimli, esnek ve performanslı bir şekilde kullanımını sağlamaktadır. Donanım ve yazılım bağımlılıklarını ortadan kaldırıp yeni servis geliştirme maliyetlerinde önemli kazanç oluşturmaktadır (Popek ve Goldberg, 1974).

Sanallaştırma, fiziksel kaynakların soyutlanmasını sağlamaktadır. Sanal makineler fiziksel donanımları paylaşabilmektedir. Sanallaştırma sistem esnekliği, verimliliği, ölçeklenebilirliği ve güvenilirliği sağlaması ve kurum maliyetlerini düşürmesi sebebiyle çok önemli bir yere sahiptir. Şekil 1'de hipervizör tipi sanallaştırma mimarisi gösterilmiştir.



Şekil 1:
Şekilde en altta bellek, işlemci, disk ve ağ gibi donanım kaynakları, üstünde KVM, Xen, Hyper-V veya VMware gibi hipervizörler ve bu hipervizörler üzerine kurulan sanal makinelerden oluşan hipervizör tipi sanallaştırma mimarisi gösterilmiştir.

Sanallaştırma teknolojisi getirdiği avantajların yanında birtakım maliyetler de oluşturmaktadır. Örneğin, sanallaştırma teknolojisinin yönetiminden sorumlu hipervizör diye adlandırılan kod parçacığı fiziksel sunucu üzerinde ek yük oluşturmaktadır. Ayrıca birçok sanal makine aynı fiziksel sunucu üzerinde çalıştığında, sanal makinelerin fiziksel kaynakların paylaşımı konusunda birbirlerini olumsuz etkilememesi açısından performans yalıtımı da kritik önem taşımaktadır. Diğer taraftan hipervizörün işlemci, bellek, disk ve ağ gibi fiziksel kaynakları nasıl zamanladığı performansı doğrudan etkilemektedir. Bu kaynaklardan her biri, sanallaştırma için farklı teknikler gerektirir ve sonucunda da hipervizörler arasında performans farklılıkları oluşur.

2.1.1 KVM

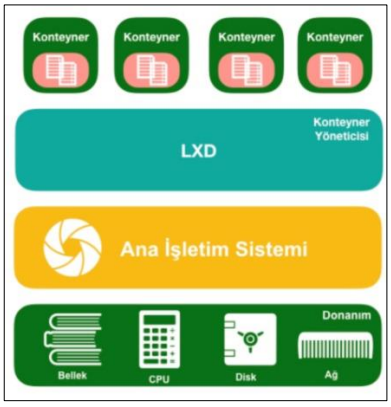
KVM, 32 (x86) bitlik işlemci mimarisine sahip ve Intel VT veya AMD-V sanallaştırma eklentileri içeren donanımlar üzerine kurulabilen Linux işletim sistemleri için çekirdek tabanlı açık kaynak hipervizör tipi sanallaştırma çözümüdür. KVM iki farklı yazılım parçasından oluşmaktadır. Birinci yazılım parçası ana sanallaştırma altyapısını ve işlemci tipine özel çekirdek

modüllerini sağlayan Linux çekirdeği modülüdür. Linux Çekirdeği modülü öncelikle sadece ayrıcalıklı CPU talimatları ile ilgilenir. Bu Linux çekirdeği modülü ayrıca sanal makinelerin oluşabilmesi için bellek yönetim birimi yazmaçları gibi alt seviyede emülasyon sağlamaktadır.

KVM'i oluşturan ikinci yazılım parçası ise sanal makinelere donanım emülasyonu sağlayan açık kaynak QEMU (Quick Emulator) yazılımının kopyası üzerine geliştirilen yazılımdır. KVM Linux çekirdek modülü tek başına sanal makinelere disk, ağ kartı vb. sürücüleri sağlamayıp yalnızca bir kullanıcı alanı programının bunu yapmasına izin vermek için gereken çekirdek modüllerini sağlamaktadır. KVM hipervizöründe sanal makinelere disk, ağ kartı sürücüleri vb. donanım emülasyonunu QEMU sağlamaktadır. QEMU tek başına eksiksiz ve bağımsız bir yazılımdır. Hipervizor olarak da görev yapabilen QEMU genelde makinelerin emülasyonunda kullanılan bir yazılımdır. Makinelere disk, ağ, VGA, PCI, USB vb. emülasyonu sağlamaktadır. Makinelere temelde, belirli bir işlemci için yazılan ikili kodu başka birine dönüştüren özel bir derleyici ile çalışır. KVM hipervizöründe ana sanallaştırma altyapısını Linux çekirdek modülü sağlarken sanal makinelere disk, ağ kartı vb. donanım emülasyonlarını QEMU sağlamaktadır.

2.2 Konteyner

Konteyner teknolojisi ismini denizcilik alanından almaktadır. Her bir ürünü göndermek için farklı bir arayış içine girmek yerine, mallar, aynı boyutlardaki konteyner denilen kaplara yerleştirilmektedir. Bu şekilde konteynerler yapılacak iş yükünün maliyetini de azaltmaktadır. Yazılım tarafında da denizcilik alanındaki mantıkla uygulamalar birbirinden yalıtılmak için konteynerlere hapsedilmektedir. Konteyner tipi sanallaştırma, işletim sistemi üzerinde bir konteyner yönetim katmanı barındıran işletim sistemi seviyesinde bir sanallaştırma çözümüdür. Konteynerler üzerinde barındığı fiziksel makinanın Linux çekirdeğini kullandığı için, bütün konteynerler aynı sürümdeki Linux çekirdeğine sahip olmak zorundadır. Bir konteyner ilgili yazılım ile beraber aynı zamanda kullanılan kütüphaneleri ve diğer yapılandırma dosyalarını da beraberinde içermektedir. Bu şekilde işletim sistemi ve özellikle donanımlar arasındaki farklılıklardan en az derecede etkilenmektedir. Konteyner teknolojisi, uygulamaların paket halinde taşınmasını ve çalıştırılmasını sağlayan bir yöntemdir. Amazon EC2, Microsoft Azure ve GCE gibi kamuya açık ana bulut bilişim sağlayıcıları, Docker, LXC ve LXD gibi konteyner yazılımları desteğini vermeye başlamıştır (DeMuro, 2018). Şekil 2'de konteyner tipi sanallaştırma mimarisi gösterilmiştir.



Şekil 2:

Şekilde en altta bellek, işlemci, disk ve ağ gibi donanım kaynakları, üstünde ana işletim sistemi, bunun üzerinde LXD gibi konteyner yöneticisi ve en üstte konteynerlerin bulunduğu Konteyner tipi sanallaştırma mimarisi gösterilmektedir.

Konteyner tipi sanallaştırma çözümleri iki Linux çekirdeği özellikleri olan Linux isim uzayları (Linux Namespaces) ve kontrol grupları (cgroups) üzerine geliştirilmiştir. Linux isim uzayları ile kaynak yalıtımı sağlanırken, kontrol grupları ile konteynerlerin kaynak yönetimi sağlanmaktadır.

Linux isim uzayları, günümüz konteyner tipi sanallaştırma çözümlerinin temelini oluşturan teknolojilerden biridir. Bu teknoloji, süreçleri birbirinden yalıtan bir Linux çekirdek özelliğidir. Fiziksel sistem donanım kaynaklarının bağımsız süreçler arasında yalıtılmasını sağlamaktadır

2006 yılında Google çalışanları Paul Menage ve Rohit Seth tarafından kontrol grupları üzerine çalışmalar başlatılmıştır. 2008'de, kontrol grupları mekanizması Linux çekirdeğinin 2.6.24 sürümüne eklenmiştir (Yemelianov, 2017). Kontrol grupları süreçlerin hiyerarşik bir şekilde gruplandırılmasını sağlayan bir Linux çekirdeği mekanizmasıdır. Çeşitli fiziksel kaynak türlerinin kullanımını, oluşturulan süreç grupları arasında sınırlandırıp bu kaynakların kullanımlarını izlemektedir. Kontrol grupları, işlemci zamanı, sistem belleği, ağ bant genişliği vb. kaynakları ayırmaya olanak tanır.

2.2.1 LXC

Tek bir Linux çekirdeği kullanılarak bir kontrol sunucusunda birden fazla yalıtılmış Linux konteyner çalıştırmak için işletim sistemi düzeyinde bir sanallaştırma yöntemidir (Linux Containers, 2019). Linux konteynerleri sanallaştırmayı kontrol grupları ve Linux isim uzaylarıyla sağlamaktadır. LXC, kontrol gruplarının sanallaştırılması için bir mekanizma olarak başlayıp sanal makinelerin bir hipervizör yükü olmadan ana makine üzerinde barındırılmasını etkin bir şekilde sağlamıştır. LXC, tüm Linux işletim sisteminin sanal makinelerde kopyalanması gerekliliğini ortadan kaldırarak yoğunluğu azaltıp kaynak verimliliğini arttırmaktadır. LXC, yalıtılmış bir alanda çalışan bir uygulamanın LXC örneğini hızlı bir şekilde oluşturarak gereksiz yere sistem yöneticilerine de yük getirmemektedir. Güçlü bir uygulama programlama arayüzü ve basit araçlar sayesinde, Linux kullanıcılarının kolayca sistem veya uygulama konteynerlerini oluşturmasını ve yönetmesini sağlar. LXC'nin amacı standart bir Linux kurulumuna mümkün olduğunca yakın ancak ayrı bir çekirdeğe ihtiyaç duymadan bir ortam yaratmaktır.

2.2.2 LXD

LXD, yeni nesil bir sistem konteyner yöneticisidir. LXD sanallaştırma çözümü, çeşitli şirketlerden ve bireysel katılımcılardan gelen katkılarla Canonical Ltd. şirketi tarafından geliştirilip yönetilmektedir. Go programlama dilinde yazılmış açık kaynak kodlu bir sanallaştırma çözümüdür. Sanal makinelere benzer bir kullanıcı deneyimi sunmakta ancak bunun yerine Linux konteynerlerini kullanmaktadır. LXD imaj tabanlıdır ve birçok Linux dağıtımı için LXD destekleyen hazır işletim sistemi imajları barındırmaktadır. LXD'nin çekirdeğini arka planda çalışan LXD Daemon diye adlandırılan yazılımsal bir program oluşturur (Graber, 2016). Rest uygulama programlama arayüzü bulunmaktadır. LXD, Linux işletim sistemi ile ağ üzerinden Rest uygulama programlama arayüzü ile haberleşerek Linux konteynerler oluşturup yönetmektedir. LXD güvenilir, ölçeklenebilir ve ileri düzeyde sistem kaynağı kontrolü sağlayan bir sanallaştırma çözümüdür. LXD, LXC'nin bir yeniden yazımı değildir, aslında yeni ve daha iyi bir kullanıcı deneyimi sunmak için LXC'nin üzerine kurulmuştur. IaaS bulut altyapısı olan OpenStack üzerinde Linux konteynerler oluşturmak için nova-lxd projesi geliştirilmiştir. Openstack bulut altyapısında LXD sanallaştırma çözümüyle kolayca Linux konteynerleri oluşturulabilmektedir.

2.3 Bulut Bilişim

Bulut bilişim, ağ üzerinden bir bulut hizmetleri platformu vasıtasıyla kullandığın kadar öde ücretlendirme modeli ile hesaplama gücü, depolama, ağ ve diğer IT kaynaklarının talep üzerine bir hizmet olarak sunulmasıdır. Bulut bilişim, bir fiziksel makinada birden çok sanal makinayı

çalıştırabilen bir yapı sağlayan sanallaştırma teknolojileri üzerine kurulmuştur. Sanallaştırma teknolojilerinin tüm özelliklerinden faydalanılmaktadır. Bulut bilişimin temel avantajları düşük maliyet, esneklik, verimlilik, ölçeklenebilirlik, erişim kolaylığı ve çevre dostu olmasıdır (Qian ve diğ., 2009).

Bulut bilişim, uzaklığa ilişkin kısıtlamaları ortadan kaldırarak donanım kaynaklarının ölçeklenebilirlik ve esneklikle paylaşılmasına olanak tanıyan özellikle telekomünikasyonda baskın bir kavramdır. Mell ve Grance (2011)'e göre bulut modeli, sağlanan kaynaklara dayanan üç hizmet modelinde sınıflandırılır: SaaS (Software as a Service), PaaS (Platform as a Service) ve IaaS (Infrastructure as a service). Uygulamaların veya yazılımların kendisi SaaS modelinde hizmet olarak müşteriye sunulurken, müşterilerin uygulamalarını oluşturmak veya geliştirmek için bir platform PaaS'de servis olarak sağlanmaktadır. Öte yandan, IaaS, hesaplama, depolama veya ağ kaynaklarının havuzlarını oluşturur ve tüketicilerin ihtiyaç doğrultusunda onları tedarik etmesine izin verir. SaaS'a örnek olarak Salesforce, Cisco, Dropbox; PaaS'a örnek olarak, Cloud Foundry, Kubernetes, Redhat Openshift ve IaaS'a örnek olarak Amazon EC2, Google Compute Engine ve OpenStack verilebilir.

2.3.1 OpenStack

OpenStack; açık kaynak kodlu bir bulut bilişim projesidir (Awasthi ve diğ., 2016). Bütün bulut türlerini desteklemektedir. OpenStack ölçeklenebilir ve yönetim kapasitesi yüksek platformlar oluşturulmasına olanak sağlamaktadır. OpenStack birbiriyle ilişkili servisler kullanarak altyapı hizmeti için çözüm sunmaktadır (OpenStack Docs, 2019). Her servis birbiriyle bütünleştirilmesini kolaylaştıran uygulama programlama arayüzüne sahiptir. İhtiyaçlara göre servislerin bir kısmı veya tamamı yüklenebilir. Temel OpenStack servisleri aşağıdaki gibidir:

- Yönetim Paneli (horizon): Kullanıcı arayüzünü aktif ederek yönetici ve kullanıcılar için OpenStack kaynak ve hizmetlerini yönetmeyi sağlar.
- Hesaplama (nova): OpenStack servislerinde yapılan işlemlerin yaşam döngülerinin hesaplanması, planlaması ve sonlandırılması işlemlerini düzenler.
- Ağ (neutron): Ağ bağlantıları ve diğer OpenStack servisleri ile bağlantı kurulmasına olanak verir.
- Nesne Depolama (swift): Yüksek erişimli, ölçeklenebilir, dağıtık ve kısmen kararlı bir nesne deposudur.
- Blok Depolama (cinder): Bu depolama alanı çalışır durumdaki sanal makinalar ve konteynerler için blok depolama sağlar.
- Kimlik Yönetimi (keystone): OpenStack hizmetleri için kimlik doğrulama ve yetkilendirme hizmeti sunar.
- İmaj Yönetimi (glance): Oluşturulacak sanal makineler için kullanılacak işletim sistemi imajlarını yönetir.

3. LİTERATÜR ARAŞTIRMASI

Bulut bilişim alanındaki ilerlemeler bulut hizmeti sağlayan bulut sağlayıcılarının da sayısının hızlıca artmasına yol açmıştır. Günümüzde kullanıcılarına bulut hizmeti sağlayan Amazon, Google, IBM, Digital Ocean, Rackspace, Huawei vb. gibi birçok firma bulunmaktadır. Bulut sağlayıcıları arasındaki en iyi performanslı bulut hizmeti sunma konusundaki rekabet, sanallaştırma ve bulut bilişim teknolojilerinde büyük ilerlemelerin kaydedilmesini sağlamıştır. Farklı bulut sağlayıcıları altyapılarında farklı sanallaştırma çözümlerini kullanmaktadır. Hipervizör tipi sanallaştırma çözümleri farklı iş yüklerine göre farklı performans sonuçları ortaya çıkarmaktadır. Literatürde hipervizör tipi sanallaştırma çözümlerinin performans değerlendirmesi konusunda sayısız çalışma bulunmaktadır. Fakat hipervizör tipi sanallaştırma teknolojilerindeki geliştirmeler devam ettiği için literatürdeki mevcut çalışmaların çoğu günümüz performans

sonuçlarını göstermemektedir. Ayrıca bulut bilişim alanındaki ilerlemelerle hipervizör tipi sanallaştırma çözümlerine alternatif konteyner tipi sanallaştırma çözümleri ortaya çıkmıştır.

Konteyner tipi sanallaştırma teknolojilerindeki ilerlemeler konteyner tipi sanallaştırma çözümleri alanında birbirine alternatif olacak konteyner tipi sanallaştırma çözümlerinin çeşitliliğine yol açmıştır. Konteyner tipi sanallaştırma teknolojileri farklı iş yüklerinde farklı performans gösterebilmektedir. Konteyner tipi sanallaştırma çözümlerinin gelişimiyle literatürde de konteyner tipi sanallaştırma çözümlerinin değerlendirmesi konunda çalışmalar yapılmıştır.

Kozhirbayev ve Sinnott (2017), Docker ve Flockport LXC konteyner sanallaştırma teknolojilerinin çeşitli bileşenlerdeki performans değerlendirmesi üzerine çalışma yapmıştır. Casalicchio ve Perciballi (2017) Docker sanallaştırma çözümünün işlemci ve disk performansı üzerine bir çalışma sunmuştur. Bu çalışmalarda konteyner teknolojileri birbirine benzer performans göstermiştir. Yukarıdaki çalışmada aynı tipteki sanallaştırma çözümleri arasında bir performans değerlendirmesi yapılmıştır. Konteyner tipi sanallaştırma çözümlerine ek olarak herhangi bir hipervizör tipi sanallaştırma çözümüyle beraber bir performans karşılaştırması yapılmamıştır. En önemlisi bulut altyapıları üzerindeki performans değerlendirmesi bulunmamaktadır.

Literatürde hipervizör tipi ve konteyner tipi sanallaştırma teknolojilerinin birlikte değerlendirildiği birçok çalışma yapılmıştır. Sampathkumar (2013) LXC, Xen ve KVM sanallaştırma teknolojilerinin işlemci, bellek ve disk performansı üzerine performans değerlendirmesi yapmıştır. LXC konteyner tipi sanallaştırma çözümünün yüksek kaynak yalıtımının gerektirmediği uygulamalarda en iyi performansı gösterdiğini, KVM'in Xen'göre bellek yoğunluklu uygulamalarda daha iyi olduğunu ve Xen'in de KVM'e göre disk yoğunluklu uygulamalarda iyi performans gösterdiğini ortaya koymuştur. Bu çalışma bize bulut altyapıları üzerinde herhangi performans değerlendirmesi sunmamaktadır. Morabito ve diğ. (2015) KVM, Docker, LXC ve OSv sanallaştırma çözümlerinin performans karşılaştırması üzerinde durmuştur. KVM üzerinde disk yoğunluklu iş yüklerinde farklı kıyaslama araçlarında farklı sonuçlar ortaya çıktığı gözlemlenmiştir. LXC ile Docker sanallaştırma çözümünün de KVM'e göre daha iyi performans gösterdiği ve kendi aralarında da göz önüne alınmayacak kadar performans farklılığı ortaya çıkmıştır. Morabito ve diğ. (2015) çalışmasında yazarlar performans karşılaştırmasının yeni ortaya çıkan konteyner tipi sanallaştırma çözümü olan LXD üzerinde de yapılmasını önermektedir. Felter ve diğ. (2015) KVM ve Docker sanallaştırma çözümlerinin performans karşılaştırmasını hem mikro kıyaslama araçları ile hem de Redis ve Mysql uygulamalarıyla yapmıştır. Her durumda Docker, KVM ile ya eşit ya da daha iyi performansta çalışmıştır. Bu çalışmada herhangi bir bulut üzerinde bir performans değerlendirmesi olmadığı gibi LXD sanallaştırma teknolojisi de göz önünde bulundurulmamıştır. Wang ve diğ. (2017) Xen ve Docker sanallaştırma çözümlerinin performans karşılaştırmasını hem mikro kıyaslama araçlarıyla hem de yüksek performans işleme uygulamalarını karşılaştıran HPL (High Performance Linpack) ve çevrimiçi veri işleme yapan uygulamaların performansını karşılaştırmada kullanılan YCSB (Yahoo! Cloud Serving Benchmark) kıyaslama aracı ile yapmıştır. Sanallaştırma teknolojileri işlemci ve bellek bakımından birbirine yakın performans sergilemişlerdir. Ancak Xen hipervizörü sistem operasyonlarındaki gecikmelerden dolayı belleğe yazma ve diske yazma konusunda daha az performans göstermiştir. Çoğu çalışma gibi bu çalışmada da bulut altyapıları üzerinde herhangi bir değerlendirme olmamıştır. Barik ve diğ. (2016) Oracle VM VirtualBox ve Docker sanallaştırma teknolojilerinin performans karşılaştırması konusunda bir çalışma yapmıştır. Bu çalışmaya göre performans ve verimlilik bakımından konteyner tipi sanallaştırmanın hipervizör tipi sanallaştırmaya kıyasla daha performanslı olduğu ortaya çıkmıştır. Güvenlik söz konusu olduğunda, hipervizör tipi sanallaştırmanın tam yalıtıma sahip olması nedeniyle hipervizör tipi sanallaştırma çözümleri konteyner tipi sanallaştırma çözümlerine göre daha tercih edilebilir durumdadır. Gupta ve Gera (2016) Docker, LXD ve VMware ESX sanallaştırma çözümleri üzerine bir çalışma yapmıştır. Yazarlar LXD'nin geliştirilmesi ile sanallaştırmanın Docker'dan farklı olarak tek bir işletim sistemine bağımlılığının kaldırıldığını ifade etmişlerdir. Yapılan testler

sonucunda Docker ve LXD, işlemci ve bellek performansı konusunda birbirine çok yakın sonuçlar ortaya çıkmıştır. Disk yoğunluklu uygulamalarda ise daha kapsamlı testlerin yapılmasının gerekliliğinden bahsetmiştir. Gupta ve Gera (2016) çalışmasında LXD incelenmesine rağmen bulut üzerinde bir performans değerlendirmesi yapılmamıştır.

Farklı tasarım amaçlarına ve çalışan uygulamaya göre, konteynerler, uygulama konteynerleri (Docker vb.) ve sistem konteynerleri (LXC vb.) olarak sınıflandırılır. Ruan ve diğ. (2016), konteynerleri farklı açılardan kullanmanın uygun yolunu keşfetmek için bir performans çalışması yürütmüştür. Uygulama konteynerleri ve sistem konteynerleri arasındaki performans farklılıklarını ölçmek için bir dizi testler gerçekleştirilip, daha sonra ana sunucu makinesi ve konteynerler arasındaki ekstra sanal makine katmanının ek yükünü değerlendirmiştir. Ayrıca Amazon ECS (Amazon Elastic Container Service) ve GKE (Google Kubernetes Engine) servis kalitesi ölçülmüştür. Sonuçlar, sistem konteynerlerinin disk yoğunluklu iş yüklerinde uygulama konteynerlerinden daha iyi olduğu ortaya çıkmıştır. Çünkü uygulama konteynerleri katmanlı dosya sistemi nedeniyle yüksek disk giriş/çıkış gecikmesi yaşamaktadır. Konteynerler sanal makinede çalıştırıldığında disk performansında %42,7'ye varan azalma ve ağ gecikmesinde %233 artış meydana gelmektedir. Ayrıca ECS'nin GKE'den daha iyi bir performansla çalıştığı ortaya çıkmıştır. Bu çalışmada yazarlar bulut platformlarının konteynerleri doğrudan ana makina üzerinde çalıştırarak daha iyi performans elde edebileceklerini söylemişlerdir. Bu çalışma bulut ortamında konteyner tipi sanallaştırma çözümlerinin karşılaştırıldığı ilk çalışma olması bakımından diğer çalışmalardan ayrılmaktadır.

4. DENEYDE KULLANILAN KIYASLAMA ARAÇLARI

Bu çalışma kapsamında kıyaslama araçları olarak IBM tarafından geliştirilen Cloudbench ve Google geliştiricileri tarafından geliştirilen PerfKit Benchmarker araçları kullanılmıştır. İş yüklerini buluta taşımadan önce, bulutun performansının iş yüklerinizi nasıl etkilediğini öğrenmek için performans testlerinin yapılması uygulamanın bulut üzerinde nasıl bir performans göstereceğini anlamak için önemlidir. Bulut ortamları, ihtiyaç duyulan karşılaştırma testlerini yapmak için herhangi bir standart, araç veya yöntem sunmamaktadır. Bu problemlerden yola çıkılarak Cloudbench ve PerfKit Benchmarker yazılımları geliştirilmiştir (IBM Developer, 2018). PerfKit Benchmarker, çoğu büyük açık veya özel bulut sağlayıcılarında, kıyaslama yazılımları ile performans testleri yapmak için ihtiyaç duyulan tüm kaynakların kurulumunu gerçekleştiren ve bu performans testlerini otomatikleştiren bir yazılım çatısıdır (PerfKitBenchmarker, 2015). Kıyaslama yazılımlarını çalıştırmak için gerekli olan ağların, alt ağların, güvenlik duvarlarının ve güvenlik duvarı kurallarının, sanal makinelerin ve sürücülerin kurulumunu otomatikleştirmektedir. Ayrıca kıyaslama yazılımlarını otomatik çalıştırdıktan sonra kurulan altyapıyı ortadan kaldırmaktadır. Literatürde bulut altyapılarının performans kıyaslamasında PerfKit Benchmarker aracının kullanıldığı çalışmalar bulunmaktadır (Kang ve diğ., 2017). Cloudbench ve PerfKit Benchmarker araçlarının temel özellikleri Tablo 1'de açıklanmıştır.

Deney kapsamında ağ yoğun iş yükleri için Iperf, disk yoğun iş yükleri için Fio (Flexible I/O) ve işlemci ile bellek yoğun iş yükleri için ise HPCC (High Performance Computing Challenge) kıyaslama araçları kullanılmıştır.

Tablo 1. Cloudbench ve PerfKit Benchmarker araçlarının özellikleri

	Cloudbench	**PerfKit Benchmarker**
Desteklenen Bulut Altyapıları	Amazon EC2, Digital Ocean, Google Compute Engine, IBM Softlayer, CloudStack ve Rackspace Cloud	Google Cloud Platform, AWS, Microsoft Azure, Alibaba Cloud, Digital Ocean, OpenStack, Cloudstack ve Rackspace
Desteklenen iş yükleri	bonnie, btest, coremark, ddgen, filebench, Fio, Iperf, netperf, postmark, xping, memtier, oldisim, wrk, hpcc, linpack, multichase, parboil, scimark , cassandra_ycsb, mongo_ycsb, ibm_daytrader, open_daytrader, specjbb, sysbench, redis_ycsb, giraph ve hadoop	Aerospike, cassandra_ycsb, cassandra_stress, cloudsuite3.0, hadoop_terasort, hpcg, Iperf, memtier, mesh_network, mongodb, mongodb_ycsb, multichase, netperf, oldisim, pgbench, silo, tomcat, Tensorflow, wrk, ycsb, cluster boot, unixbench, SPEC CPU 2006, coremark, hpcc, Scimark2, bonnie, Fio, silo ve copy throughput

4.1 Iperf

Ağın bant genişliği ve hızını ölçen kıyaslama aracıdır. Saniyede yapılan veri transferinin değerini ölçer. Iperf istemci ve sunucu işlevselliğine sahiptir ve iki uç arasındaki bant genişliğini bir veya her iki yönde ölçmek için veri akışları oluşturabilmektedir. Iperf testleri yapılırken bir tarafta istemci, diğer tarafta sunucu makinesi olup, istemciden sunucuya istekler gönderilerek ağın hızı ve bant genişliği ölçülür. Iperf, C ile yazılmış açık kaynaklı bir yazılımdır ve Linux, Unix ve Window gibi çeşitli platformlarda çalışmaktadır. Bu çalışma kapsamında Iperf testleri Cloudbench ve PerfKit Benchmarker tarafından otomatik olarak gerçekleştirilmiştir.

4.2 Fio

Fio, ilk olarak Jens Axboe tarafından geliştirilen açık kaynaklı bir sentetik kıyaslama aracıdır. Fio, çeşitli giriş/çıkış iş yükleri oluşturmaktadır. Tümü kullanıcı tarafından sağlanan seçeneklere bağlı olarak gerçekleştirilen sıralı okumalar ve yazmalar veya rasgele okuma ve yazmalar, senkron ve asenkron yazmalar bu iş yüklerindendir. Fio'da test yapılmak istenen senaryolar alacağı parametrelerle birlikte bir dosyaya kaydedilir. Sonrasında ise fio yazılımına bu dosya gösterilerek ilgili senaryolardaki testler gerçekleştirilir. Fio, farklı iş yükü türlerinin üretilebileceği çeşitli seçenekler sunar. Fio, depolama sisteminde giriş/çıkış performans testlerini hızlı bir şekilde gerçekleştirmek için en kolay ve çok yönlü bir araçtır. Farklı tipteki giriş/çıkış yüklerinin benzetimini sağlamaktadır. Bu çalışma kapsamında disk performansı karşılaştırması için sıralı okuma, sıralı yazma, rasgele okuma ve rasgele yazma iş yükleri kullanılmıştır.

4.3 HPCC

HPCC, gelecekteki Petascale bilgi işlem sistemlerini incelemek için geliştirilmiş olup modern bilgi işlem iş yüklerinin gerçekçi bir ölçümünü sağlamak için tasarlanmıştır. HPCC, çeşitli bellek erişim düzenlerini ölçen bir kıyaslama paketidir. İşlemci ve bellek gücünü ölçen 7 çeşit testten oluşmaktadır.

- HPL: Doğrusal bir denklem sistemini çözmek için kayan nokta işlem hızını ölçen bir kıyaslama aracıdır. Kayan nokta işlem hızı saniyede bir trilyon yani teraflops biriminden hesaplanmaktadır.
- DGEMM: çift duyarlıklı gerçek matris-matris çarpımının kayan nokta işlem hızını ölçer.
- Stream: Sürdürülebilir bellek bant genişliğini ve basit vektör çekirdeği için karşılık gelen hesaplama oranını ölçen basit bir sentetik kıyaslama aracıdır. Bellek bant genişliğini saniyede GB biriminden hesaplamaktadır.
- PTRANS (Parallel Matrix Transpose): Ağın toplam iletişim kapasitesinin yararlı bir testidir.
- RandomAccess: Belleğin tamsayı rasgele güncelleme oranını ölçer.

- FFT: Çift duyarlıklı karmaşık tek boyutlu Ayrık Fourier Dönüşümünün kayan nokta uygulama hızını ölçer.
- Communication bandwidth and latency: Birkaç eşzamanlı iletişim düzeninin gecikme ve bant genişliğini ölçmek için b_eff kıyaslama aracı temelli testler kümesini oluşturmaktadır.

Bu çalışma kapsamında işlemci gücünü ölçmek için HPCC aracı altındaki HPL, bellek gücünü ölçmek için ise Stream kıyaslama araçları kullanılmıştır. HPL kıyaslama aracı ile saniyede gerçekleşen teraflops biriminden kayan nokta işlemi hesaplanmaktadır. Stream kıyaslama aracı ile de sürdürebilir bellek bant genişliğini ölçmek için "add", "copy", "scale" ve "triad" vektörel işlemleri uygulanarak saniyede GB biriminden sonuç elde edilmektedir. Yapılan dört vektörel işlem ile ilgili bilgiler Tablo 2'de bulunmaktadır.

Tablo 2. Stream kıyaslama aracı üzerinden gerçekleşen vektörel işlemler

Vektörel işlem	Açıklaması
copy	C[i] = A[i]
scale	B[i] = scalar * C[i]
add	C[i] = A[i] + B[i]
triad	A[i] = B[i] + scalar*C[i]

5. METOT

Bu çalışmada OpenStack bulut altyapısında KVM ve LXD sanallaştırma teknolojilerinin performans değerlerinin karşılaştırılması ve farklı iş yüklerinde uygun sanallaştırma teknolojisinin seçimi yapılabilmesi için bu performans değerleri üzerinde analizler yapılması amaçlanmıştır. Bu çalışmada kıyaslama yazılımları aracıyla işlemci, bellek, ağ ve disk yoğun iş yükleri oluşturulup sistemin çıktısı olan performans değerleri yorumlanmıştır. Performans değerlendirilmesini yaptığımız bu çalışmayı Deneysel Bilgisayar Bilimi (ECS) alanında değerlendirebiliriz (Feitelson, 2006).

Performans değerlendirmesinin yapılabilmesi amacıyla deney ortamlarının hazırlanması ve deneyin uygulanması sürecinde araştırma metodolojisi olarak Li ve diğ. (2016) tarafından tasarlanan Alan Bilgisine Dayalı Metodoloji (DoKnowMe) takip edilmiştir. Bu metodolojide takip edilecek adımlar gereksinimlerin tanımlanması, performans özelliklerinin ve kriterlerinin tanımlanması, deneyde kullanılan metrik ve kıyaslama yazılımı araçlarının listelenmesi ve seçimi, deneysel faktörlerin listelenmesi ve seçimi, deneyin tasarlanması ve uygulanması, deneyin analizi ve yorumlanması adımlarıdır.

Deneylerin gerçekleştirilmesi sürecinde takip ettiğimiz metodolojiye göre deneyin yapılış amacının net ifadelerle belirtilmesi gerekmektedir. Deneyde ulaşılmak istenen hedefin belirtilmesi için de bir takım sorularla deneyin gereksinimleri tanımlanmalıdır. Deneyin gereksinimleri olarak aşağıdaki sorulara cevap aranmaktadır.

Soru 1: OpenStack bulut ortamında KVM sanallaştırma teknolojisi tercih edildiği zaman hangi iş yüklerinde LXD'ye göre daha iyi performans göstermektedir?

Soru 2: OpenStack bulut ortamında LXD sanallaştırma teknolojisi tercih edildiği zaman hangi iş yüklerinde KVM'ye göre daha iyi performans göstermektedir?

Bulut ortamlarında performans değerlendirmesi genelde işlemci, bellek, ağ ve disk yoğun iş yükleri üzerinden yapılmaktadır (Li ve diğ., 2012). Bu nedenle bu çalışma da işlemci, bellek, ağ ve disk yoğun iş yükleri oluşturularak gerçekleştirilmiştir. OpenStack bulut ortamında KVM ve LXD sanallaştırma teknolojileri işlemci yoğun iş yükü ile kayan nokta işlem sayısı, bellek yoğun iş yükü ile sürdürebilir bellek bant genişliği, ağ yoğun iş yükü ile ağ bant genişliği ve disk yoğun iş yükü ile sıralı okuma/yazma bant genişliği, rasgele okuma/yazma bant genişliği, sıralı okuma/yazma giriş/çıkış sayısı ve rasgele okuma/yazma giriş/çıkış sayısı performans özellikleri ve kriterleri temel alınarak değerlendirilmiştir.

Kıyaslama yazılım araçlarının tercih edilmesinde çalışacağı deney ortamına uygunluğu, sistem üzerinde fazla ek yük getirmemesi ve yaygın olarak kullanılma gibi kriterler bulunmaktadır. Bu çalışmada kıyaslama yazılımlarını yöneten ve bu kıyaslama yazılımlarını otomatik olarak çalıştırabilen Cloudbench ve Perfkit Benchmarker araçları kullanılmıştır. Kıyaslama yazılımı araçları olarak ise bu her iki kıyaslama aracının desteklediği HPCC, Iperf ve fio tercih edilmiştir. Kullanılan kıyaslama araçları dördüncü bölümde detaylı olarak açıklanmıştır. Tablo 3'te iş yükü tipine göre kullanılan kıyaslama yazılımı araçları ve ölçülen değerler listelenmiştir.

Tablo 3. İş yükü tipine göre kullanılan kıyaslama yazılımı araçları ve ölçülen değerler

İş yükü tipi	Ölçülen performans değeri	Birim	Kullanılan kıyaslama yazılımı
Ağ	Ağ bant genişliği	Mbits/sec	Iperf
İşlemci	Kayan nokta işlem sayısı	Tflop/s	HPL
Bellek	Bellek bant genişliği	GB/s	Stream
Disk	Sıralı okuma bant genişliği	KB/s	Fio
Disk	Sıralı yazma bant genişliği	KB/s	Fio
Disk	Sıralı okuma giriş/çıkış sayısı	iops	Fio
Disk	Sıralı yazma giriş/çıkış sayısı	iops	Fio
Disk	Rasgele okuma bant genişliği	KB/s	Fio
Disk	Rasgele yazma bant genişliği	KB/s	Fio
Disk	Rasgele okuma giriş/çıkış sayısı	iops	Fio
Disk	Rasgele yazma giriş/çıkış sayısı	iops	Fio

Deney sonucunda ortaya çıkan sonuçların tekrarlanabilmesi ve güvenilirliği için deney faktörlerinin iyi bir şekilde belirlenmesi gerekmektedir. Bu çalışmada belirlenen deney faktörlerin birincisi KVM ve LXD sanallaştırma teknolojilerinin her ikisinin de aynı özellikteki sunucular üzerine kurulumunun gerçekleştirilmesidir. Kullanılan sunucuların özellikleri Tablo 4'te gösterilmiştir.

Tablo 4. Kullanılan sunucu özellikleri

Özellik	Çeşidi
Marka	HP Proliant XL170r Gen9
İşlemci	Intel(R) Xeon(R) CPU E5-2670 V3 @ 2.30GHz 24 Core 48 Thread
Bellek	HPE SmartMemory 2133 MHz 256 GB
Disk	6 TB
Ağ Bağlantı Kartı	2 HPE Eth 10/25 GB 2p 640SFP28 Adptr
İşletim Sistemi	Ubuntu 16.04 LTS(Xenial Xerus)

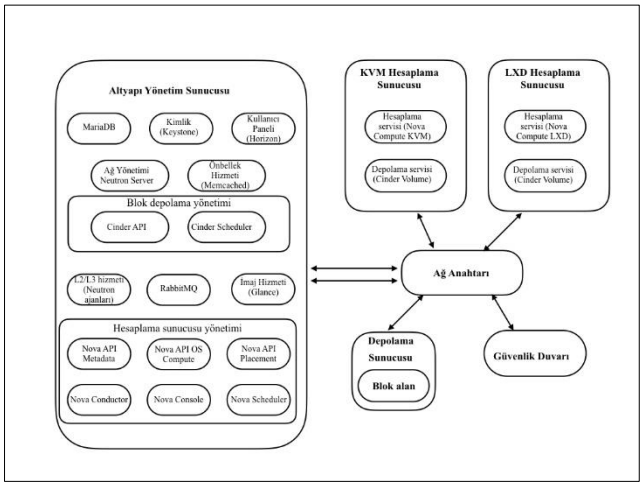
Bu çalışmada belirlenen diğer faktör de iş yüklerinin çalıştırılacağı KVM üzerindeki sanal makine ve LXD üzerindeki konteynerin aynı özelliklerde olması gerekmektedir. Deneydeki sanal makine ve konteyner Ubuntu 16.04 LTS işletim sisteminde iki vCPU, dört GB bellek ve 40 GB disk özelliklerine sahiptir. Bu çalışmada belirlenen son faktör ise her iki sanallaştırma teknolojisi

üzerinde çalışacak iş yükünün aynı olması gerekmektedir. Aynı komut ve parametrelerle betikler çalıştırılmıştır.

Takip ettiğimiz Alan Bilgisine Dayalı Metodolojisinin bir sonraki adımı olan deneyin tasarlanması ve uygulanması altıncı bölümde, deneyin analizi ve yorumlanması adımı ise yedinci bölümde detaylı açıklanmıştır.

6. DENEYİN TASARLANMASI VE UYGULANMASI

Farklı tiplerde iş yüklerinin sanallaştırma çözümleri üzerinde performans kayıplarının değerlendirilmesi için OpenStack Pike sürümünde bir bulut altyapısı kurulumu gerçekleşmiştir. Dört sunucu üzerine Ubuntu 16.04 LTS işletim sistemi kurulumu gerçekleştirilmiştir. Birinci sunucuya OpenStack Yönetim servisleri, ikinci sunucuya blok depolama servisi, üçüncü sunucuya OpenStack KVM hesaplama servisi ve dördüncü sunucuya da OpenStack LXD hesaplama servisi kurulmuştur. OpenStack kurulumu için "bash" betikleri yazılıp, kurulum otomatik şekilde gerçekleştirilmiştir. Farklı sanal makineler içerisine Cloudbench ve PerfKit Benchmarker araçlarının kurulumu gerçekleştirilmiştir. Testler KVM ve LXD hesaplama sunucuları üzerinde eşit sayıda sanal makine oluşturacak şekilde gerçekleştirilmiştir. Yönetici sunucusunda veri tabanı olarak MariaDB, ön bellek hizmeti olarak Memcached, mesaj kuyruğu servisi olarak RabbitMQ kurulmuştur. OpenStack servislerinden de kimlik servisi (keystone), kullanıcı paneli (horizon), ağ yönetimi (neutron server), blok depolama yönetimi (cinder API ve cinder scheduler), L2/L3 hizmeti (neutron ajanları), imaj (glance) ve hesaplama sunucusu yönetici hizmetleri (nova API metadata, nova API OS compute, nova API placement, nova conductor, nova console ve nova scheduler) kurulumu gerçekleştirilmiştir. KVM hesaplama sunucusunda sanal makinaların oluşumundan ve yönetiminden sorumlu nova compute KVM ve blok depolama servisi olan cinder volume servisleri kurulumu gerçekleşmiştir. LXD hesaplama sunucusunda sanal makinaların oluşumundan ve yönetiminden sorumlu nova compute LXD ve blok depolama servisi olan cinder volume servisleri kurulumu gerçekleşmiştir. Tasarlanan bulut mimarisi Şekil 3'de gösterilmiştir.

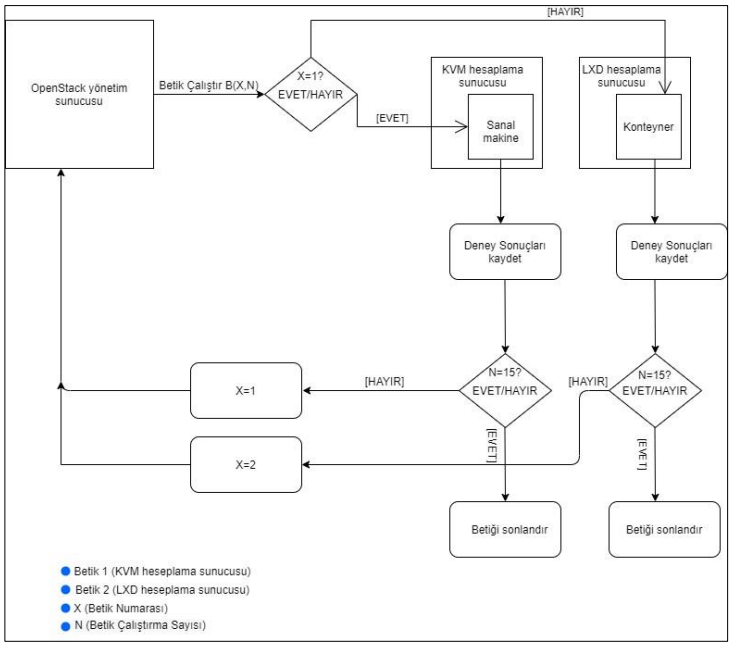


Şekil 3:

Şekilde altyapı yönetim, KVM hesaplama, LXD hesaplama ve depolama sunucularından oluşan OpenStack bulut ortamı sistem mimarisi gösterilmiştir.

Bulut ortamı tasarlanıp kurulduktan sonra altyapı yönetim sunucusuna PerfKit Benchmarker ve Cloudbench araçları kurulmuştur. PerfKit Benchmarker ve Cloudbench araçlarını kullanan betiklere, iş yükü tipine uygun kıyaslama aracı ve parametreler tanımlanıp, ilgili sanal makine

veya konteyner üzerinde bu betikler çalıştırılarak performans değerleri kaydedilmiştir. Her betik 15 kere çalıştırılmıştır ve bu sonuçlar kaydedilmiştir. Deney akış şeması Şekil 4'te gösterilmiştir.



Şekil 4:
Şekilde deney akış şeması gösterilmiştir.

Deney sonucunda elde edilen performans değerleri, MS Ofis Excel programı üzerinde tablolar halinde her bir iş yükü için ayrı ayrı tutulmuştur. Elde edilen 15 deney sonucundaki performans değerlerinin ortalaması alınıp, bu elde edilen yeni değerler üzerinden deney analiz edip yorumlanmıştır.

7. DENEYİN ANALİZİ VE YORUMLANMASI

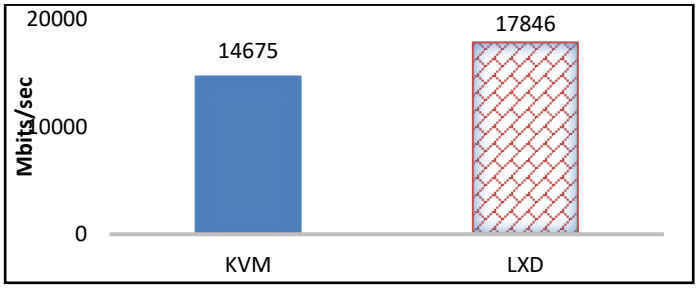
Bu bölümde ağ, işlemci, bellek ve disk performans test sonuçları ve yorumları detaylarıyla verilmiştir.

7.1 Ağ performans test sonuçları ve yorumlanması

Ağ performansını belirlemek için Iperf kıyaslama aracı kullanılmıştır. Iperf testleri hem Cloudbench hem de PerfKit Benchmarker araçları ile KVM ve LXD hesaplama sunucuları üzerinde 15 defa çalıştırılıp ortalama bant genişliği değerleri alınarak gerçekleştirilmiştir. Cloudbench ve PerfKit Benchmarker araçları otomatik olarak hesaplama sunucularında Iperf istemci ve sunucu sanal makinelerini oluşturup Iperf istemci makinadan Iperf sunucu makinesine istekler göndererek testleri otomatik gerçekleştirmiştir.

Şekil 5'te gösterilen değerler 60 saniyede istemci makinesinden sunucu makinesine megabits cinsinden aktarılan ortalama veri miktarını ifade etmektedir. Yüksek değerler daha iyi ağ performansını göstermektedir. Yapılan 15 test sonucunda Şekil 5'de görüleceği üzere LXD hesaplama sunucusu üzerinde ortalama 17846 Mbits/sec ve KVM hesaplama sunucusu üzerinde ortalama 14675 Mbits/sec bant genişliği elde edilmiştir. Ağ performansını değerlendirdiğimizde Şekil 5'teki gibi LXD konteyner tipi sanallaştırma çözümünün daha iyi performans gösterdiği ortaya çıkmıştır. Bu sonucun ortaya çıkmasındaki en önemli etken LXD'nin ağ oluşumunda ve yalıtımında Linux ağ isim uzaylarını kullanmasıdır. Herhangi bir hipervizör yükü bulunmamaktadır. KVM'de Linux çekirdeğinin de özelliklerini kullanmasına rağmen bir

hipervizör yükü bulunmaktadır. Bu hipervizör yükü de ağ üzerinde gecikmelere sebep olmaktadır. Bu yüzden LXD konteyner tipi sanallaştırma çözümü KVM'e göre daha iyi performans göstermiştir. Ayrıca Şekil 5'teki ağ bant genişliği performans değerleri incelendiğinde OpenStack bulut ortamındaki nova-lxd eklentisi, KVM'deki hipervizör yükü kadar ek yük getirmemiştir.

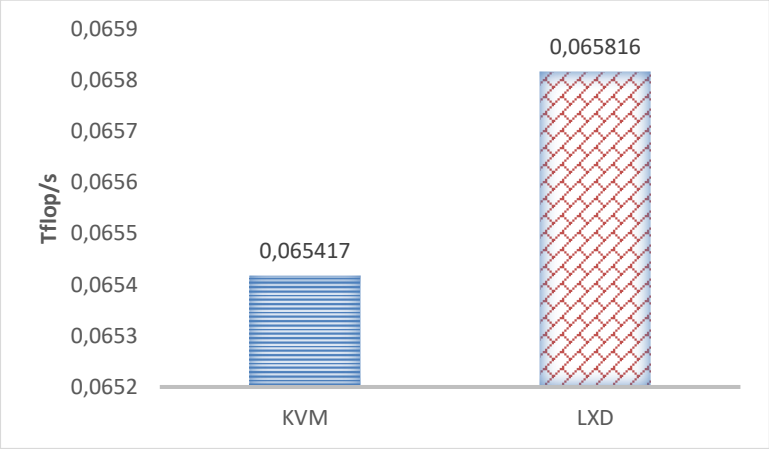


Şekil 5:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin Iperf kıyaslama aracı ile yapılan ağ performansı ölçümlerinin karşılaştırma sonuçları verilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).

7.2 İşlemci performans test sonuçları ve yorumlanması

İşlemci performansı için HPCC kıyaslama aracı kullanılmıştır. HPCC kıyaslama aracında işlemci performansını ölçen HPL kıyaslama yazılımı kullanılmıştır. HPL, bir lineer denklem sistemini çözmek için kayan nokta işlem sayısını ölçmektedir. HPL testleri PerfKit Benchmarker aracı ile LXD ve KVM hesaplama sunucuları üzerinde 15 defa çalıştırılıp HPL kıyaslama aracı için saniyedeki ortalama kayan nokta işlem sayısı temel alınarak gerçekleştirilmiştir. PerfKit Benchmarker aracı otomatik olarak hesaplama sunucusunda HPCC sanal makinesini oluşturup bu makine üzerinde HPL testlerini otomatik gerçekleştirmiştir. Şekil 6'da gösterilen değerler işlemciler için saniye başına yapılan trilyonda bir konumlandırma işlemi sayısını ifade etmektedir. Yüksek değerler daha iyi işlemci performansını göstermektedir. HPL için ayrı ayrı gerçekleştirilen 15 test sonucunda Şekil 6'da görüleceği üzere LXD hesaplama sunucusu üzerinde HPL için ortalama 0,065816 Tflop/s, KVM hesaplama sunucusu üzerinde HPL için ortalama 0,065417 Tflop/s değerleri elde edilmiştir.

İşlemci performansını HPL kıyaslama aracı ile değerlendirdiğimizde Şekil 6'daki gibi LXD konteyner tipi sanallaştırma çözümünün biraz daha iyi performans gösterdiği ortaya çıkmıştır. Fakat performans farkı göz ardı edilebilecek kadar düşüktür. LXD ve KVM sanallaştırma teknolojilerinin benzer sonuçlar vermesi her iki sanallaştırma teknolojisinin de Linux çekirdeği seviyesinde sanallaştırma sağlamalarıdır. Ayrıca Şekil 6'daki saniyedeki ortalama kayan nokta işlem sayısı performans değerleri incelendiğinde OpenStack bulut ortamındaki nova-lxd eklentisi sistem üzerine KVM'deki hipervizör yükü ile hemen hemen aynı ek yük getirmiştir. KVM ve LXD varsayılan olarak işlemci kullanımını mümkün olduğunca verimli bir şekilde kullanmayı amaçlayan CFS (Completely Fair Scheduler) işlemci zamanlaması algoritmasını kullanmaktadır.

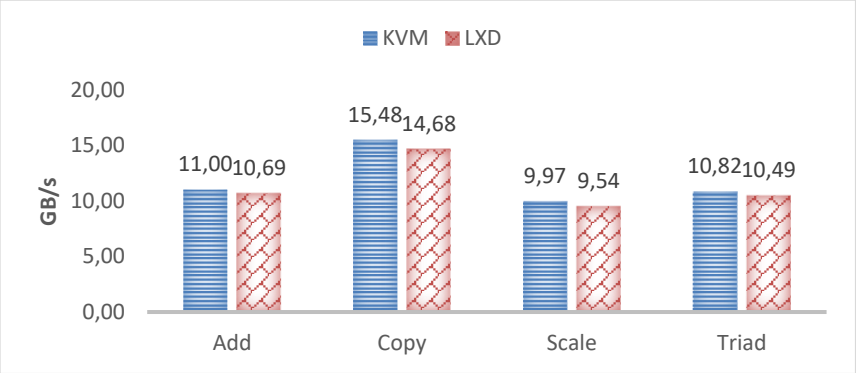


Şekil 6:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin HPL kıyaslama aracı ile yapılan işlemci performans ölçümlerinin karşılaştırma sonuçları verilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).

7.3 Bellek performans test sonuçları ve yorumlanması

Bellek performansı için HPCC kıyaslama aracı kullanılmıştır. HPCC kıyaslama aracında bellek performansını ölçen Stream kıyaslama yazılımı kullanılmıştır. Stream, sürdürülebilir bellek bant genişliğini ve dört basit vektör çekirdekleri olan "copy", "scale", "add" ve "triad" için karşılık gelen hesaplama oranını ölçen basit bir sentetik kıyaslama aracıdır. Stream kıyaslama aracıyla yapılan testlerde "copy", "scale", "add" ve "triad" vektörel işlemleri yapılıp bellek bant genişliği hesabı yapılmıştır. Stream testleri PerfKit Benchmarker aracı ile LXD ve KVM hesaplama sunucuları üzerinde 15 defa çalıştırılıp "copy", "scale", "add" ve "triad" vektörel işlemleri için ayrı ayrı saniyedeki Gigabyte cinsinden bellek bant genişliği hesaplanarak uygulanmıştır. PerfKit Benchmarker aracı otomatik olarak hesaplama sunucularında HPCC sanal makinesini oluşturup bu makine üzerinde Stream testini otomatik gerçekleştirmiştir. Stream testi işlemci çekirdeklerini maksimum sınırlarında kullanarak bellek bant genişliğini ölçmektedir. Şekil 7'de elde edilen sonuçlar belleğin bir bölgesinden başka bir bölgesine "copy", "scale", "add" ve "triad" vektör işlemleri uygulanarak saniyede gigabyte cinsinden taşınan veri miktarını ifade etmektedir. Yüksek değerler daha iyi bellek performansını göstermektedir. Stream testi için gerçekleştirilen 15 test sonucunda Şekil 7'de görüleceği üzere LXD hesaplama sunucusu üzerinde "copy" vektörel işlemi için ortalama 14,68 GB/s, "scale" vektörel işlemi için ortalama 9,54 GB/s, "add" vektörel işlemi için ortalama 10,69 GB/s ve "triad" vektörel işlemi için ortalama 10,49 GB/s değerleri elde edilirken KVM hesaplama sunucusu üzerinde "copy" vektörel işlemi için ortalama 15,48 GB/s, "scale" vektörel işlemi için ortalama 9,97 GB/s, "add" vektörel işlemi için ortalama 11,00 GB/s ve "triad" vektörel işlemi için ortalama 10,82 GB/s değerleri elde edilmiştir.

Bellek performansını Stream kıyaslama aracı ile değerlendirdiğimizde Şekil 7'deki gibi KVM hipervizör tipi sanallaştırma çözümünün biraz daha iyi performans gösterdiği ortaya çıkmıştır. Fakat performans farkı göz ardı edilebilecek kadar düşüktür. LXD ve KVM sanallaştırma teknolojilerinin benzer sonuçlar vermesi her iki sanallaştırma teknolojisinin de Linux çekirdeği seviyesinde sanallaştırma sağlamalarıdır. Ayrıca Şekil 7'deki değerler hem LXD'deki nova-lxd hem de KVM'deki hipervizör yükünün eşit seviyede ek yük getirdiğini göstermiştir. Elde edilen sonuçlar daha yeni olan LXD konteyner sanallaştırma çözümünün bellek performansı açısından geliştirmeye ihtiyacı olduğunu göstermektedir.



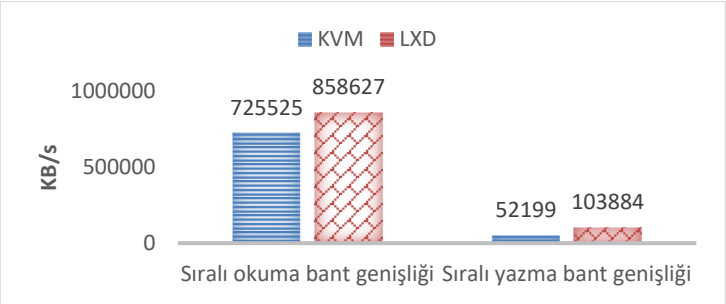
Şekil 7:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin Stream kıyaslama aracı ile yapılan "add", "copy", "scale" ve "triad" vektörel işlemleriyle bellek performansı ölçümlerinin karşılaştırma sonuçları verilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).

7.4 Disk okuma/yazma performans test sonuçları ve yorumlanması

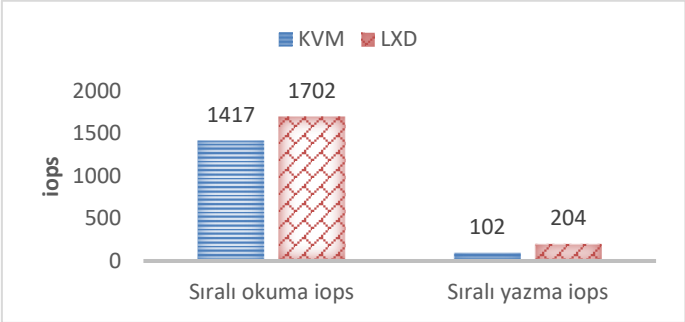
Disk okuma/yazma performansı için Fio kıyaslama aracı kullanılmıştır. Giriş çıkış zamanlayıcısı olarak Ubuntu işletim sisteminin varsayılan giriş/çıkış zamanlayıcısı olan deadline giriş/çıkış zamanlayıcısı kullanılmıştır. Depolama cihazlarında diske sıralı okuma, sıralı yazma, rasgele okuma ve rasgele yazma gibi işlemlerle disk performansları bant genişliği ve iops değerleri açısından değerlendirilmektedir. Fio testleri hem PerfKit Benchmarker hem de Cloudbench araçları ile LXD ve KVM hesaplama sunucuları üzerinde 15 defa çalıştırılıp diske sıralı okuma, sıralı yazma, rasgele okuma ve rasgele yazma işlemleri için ayrı ayrı bant genişliği ve iops değerleri elde edilerek uygulanmıştır. PerfKit Benchmarker ve Cloudbench araçları otomatik olarak hesaplama sunucularında Fio sanal makinesini oluşturup bu makine üzerinde Fio testlerini otomatik gerçekleştirmiştir. Şekil 8'de gösterilen değerler saniyede ortalama kilobyte cinsinden diskten sıralı okunan ve diske sıralı yazılan veri miktarını ifade etmektedir. Yüksek değerler diskten sıralı okuma ve diske sıralı yazmada bant genişliği bakımından daha iyi disk performansını göstermektedir. Şekil 9'da gösterilen değerler saniyede ortalama giriş/çıkış işlemi (iops) cinsinden diske sıralı okumada ve diskten sıralı yazmada gerçekleştirilen giriş/çıkış sayısını ifade etmektedir. Yüksek değerler diskten sıralı okuma ve diske sıralı yazmada giriş/çıkış sayısı bakımından daha iyi disk performansını göstermektedir. Fio kıyaslama aracı ile sıralı okuma işlemi için gerçekleştirilen 15 test sonucunda Şekil 8 ve Şekil 9 üzerinde görüleceği üzere LXD hesaplama sunucusu üzerinde bant genişliği ortalama 858627 KB/s ve ortalama saniyede 1702 giriş/çıkış sayısı değerleri elde edilmiş iken sıralı yazma işlemi için bant genişliği ortalama 103884 KB/s ve ortalama saniyede 204 giriş/çıkış sayısı değerleri elde edilmiştir. KVM hesaplama sunucusu üzerinde ise sıralı okuma işlemi için bant genişliği ortalama 725525 KB/s ve saniyede ortalama 1417 giriş/çıkış sayısı değerleri elde edilmiş iken sıralı yazma işlemi için bant genişliği ortalama 52199 KB/s ve ortalama saniyede 102 giriş/çıkış sayısı değerleri elde edilmiştir.

Disk performansını disk üzerinde sıralı okuma ve sıralı yazmada, bant genişliği ve giriş/çıkış sayısı açısından değerlendirdiğimizde sırasıyla Şekil 8 ve Şekil 9'da görüldüğü üzere LXD sanallaştırma teknolojisinin KVM'den çok daha iyi performans gösterdiği görülmüştür. Sıralı okuma ve sıralı yazma teknikleri video, müzik ve yüksek çözünürlüklü görüntüler gibi büyük dosyalar okurken ve yazarken kullanılan tekniklerdir. Burada LXD konteyner tipi sanallaştırma çözümünün gücü ortaya çıkmaktadır. Bu sonucun ortaya çıkmasındaki temel sebep arada KVM hipervizörünün olmasına bağlı olarak Linux işletim sisteminin gerçekleştirilen okuma yazma işleminin sıralı olduğunu fark edememesi ve çekirdek seviyesinde gerekli giriş/çıkış

tamponlamasını yapamamasıdır. Ayrıca KVM hipervizörü LXD'ye göre disk üzerinde sıralı okuma ve yazma işlemlerinde daha fazla ek yük getirmiştir.



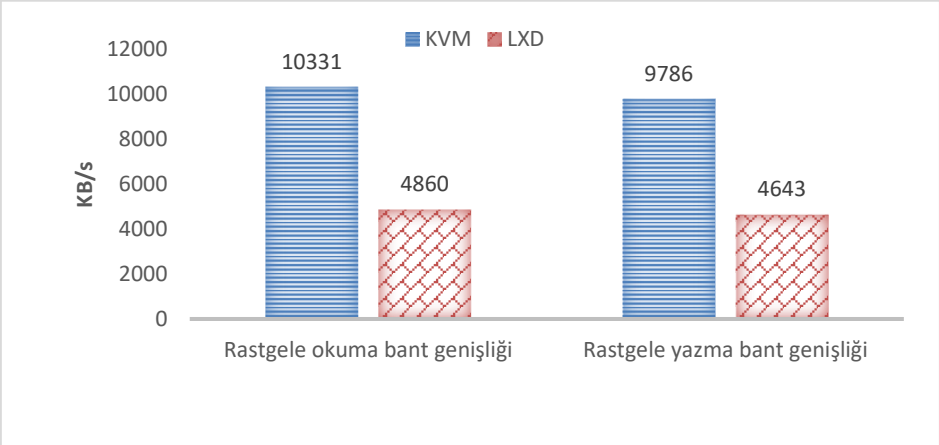
Şekil 8:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin Fio kıyaslama aracı ile yapılan sıralı okuma ve sıralı yazma işlemleri için bant genişliği performans ölçümlerinin karşılaştırma sonuçları verilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).



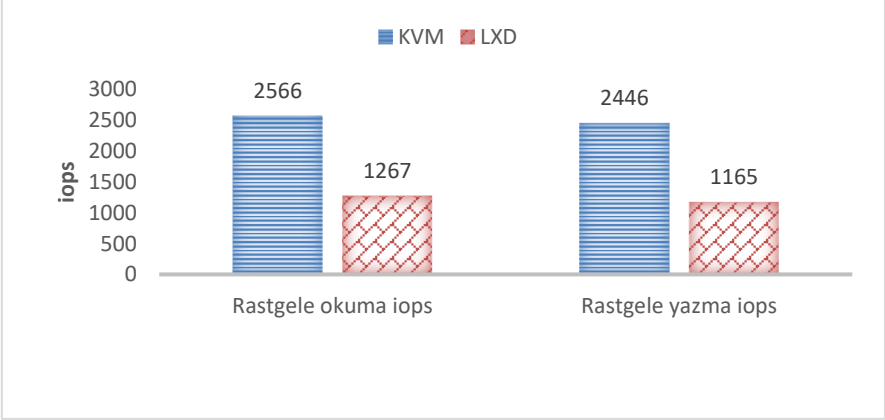
Şekil 9:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin Fio kıyaslama aracı ile yapılan sıralı okuma ve sıralı yazma işlemleri için iops performans ölçümlerinin karşılaştırma sonuçları gösterilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).

Şekil 10'da gösterilen değerler saniyede ortalama kilobyte cinsinden diskten rasgele okunan ve diske rasgele yazılan veri miktarını ifade etmektedir. Yüksek değerler diskten rasgele okuma ve diske rasgele yazmada bant genişliği bakımından daha iyi disk performansını göstermektedir. Şekil 11'de gösterilen değerler saniyede ortalama giriş/çıkış işlemi (iops) cinsinden diske rasgele okumada ve diskten rasgele yazmada gerçekleştirilen giriş/çıkış sayısını ifade etmektedir. Yüksek değerler diskten rasgele okuma ve diske rasgele yazmada giriş/çıkış sayısı bakımından daha iyi disk performansını göstermektedir. Fio kıyaslama aracı ile yapılan rasgele okuma ve yazma işlemleri için yapılan 15 test sonucunda Şekil 10 ve Şekil 11 üzerinde görüleceği üzere LXD hesaplama sunucusu üzerinde rasgele okuma işlemi için bant genişliği ortalama 4860 KB/s ve saniyede ortalama 1267 giriş/çıkış sayısı değerleri elde edilmiş iken rasgele yazma işlemi için bant genişliği ortalama 4643 KB/s ve saniyede ortalama 1165 giriş/çıkış sayısı değerleri elde edilmiştir. KVM hesaplama sunucusu üzerinde ise rasgele okuma işlemi için bant genişliği ortalama 10331 KB/s ve saniyede ortalama 2566 giriş/çıkış değerleri elde edilmiş iken rasgele yazma işlemi için bant genişliği 9786 KB/s ve saniyede ortalama 2446 giriş/çıkış değerleri elde edilmiştir.

Disk performansını disk üzerinde rasgele okuma ve rasgele yazmada, bant genişliği ve giriş/çıkış sayısı açısından değerlendirdiğimizde sırasıyla Şekil 8 ve Şekil 9'da görüldüğü üzere sıralı okuma ve sıralı yazmanın aksine KVM sanallaştırma teknolojisinin LXD'den çok daha iyi performans gösterdiği görülmüştür. Rasgele okuma ve rasgele yazma teknikleri küçük dosyalar okurken ve yazarken kullanılan tekniklerdir. Veri tabanları bu tekniği kullanmaktadır. Burada da KVM hipervizör tipi sanallaştırma çözümünün gücü ortaya çıkmaktadır. Giriş çıkış zamanlayıcısı için hem KVM hem de LXD sanallaştırma çözümü üzerinde bulundukları Ubuntu 16.04 LTS işletim sisteminin varsayılan giriş çıkış zamanlayıcısı olan deadline zamanlayıcısını kullanmaktadır. KVM'de hipervizör yükü olmasına rağmen LXD'den daha iyi sonuç vermiştir. Bu sonuç OpenStack bulut altyapısını LXD ile bütünleşik olmasını sağlayan nova-lxd eklentisinin küçük boyuttaki dosya işlemlerinde etkili performans göstermediğini ortaya çıkarmıştır.



Şekil 10:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin Fio kıyaslama yazılı ile yapılan rasgele okuma ve rasgele yazma işlemleri için bant genişliği performans karşılaştırma sonuçları verilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).



Şekil 11:
Şekilde LXD ve KVM sanallaştırma teknolojilerinin Fio kıyaslama aracı ile yapılan rasgele okuma ve rasgele yazma işlemleri için iops performans karşılaştırma sonuçları verilmiştir (Yüksek değerler daha iyi performansı göstermektedir.).

8. SONUÇ VE ÖNERİLER

Bu çalışma kapsamında OpenStack bulut altyapısında KVM ve LXD sanallaştırma çözümleri ile bir bulut ortamı tasarlanmıştır. Bu bulut ortamında Alan Bilgisine Dayalı Metodolojisi uygulanarak KVM ve LXD sanallaştırma çözümlerinin ağ, işlemci, bellek ve disk yoğunluklu iş yüklerinden oluşan uygulama tipleriyle performans değerlendirilmesi yapılmıştır. Bu çalışma OpenStack bulut altyapısında sanallaştırma teknolojilerinin farklı tipteki iş yükleriyle performans değerlendirmesinin yapıldığı ilk çalışma olarak literatürde gerçekleşen çalışmalardan ayrılmaktadır.

Bu çalışma Cloudbench ve PerfKit Benchmarker araçları ile OpenStack bulut ortamında gerçekleştirilmiştir. Testler her iki kıyaslama aracı kullanılarak doğrulanmıştır. Ağ yoğunluklu iş yükü olarak Iperf, işlemci yoğunluklu iş yükü olarak HPL, bellek yoğunluklu iş yükü olarak Stream ve disk yoğunluklu iş yükü olarak Fio kıyaslama araçları kullanılmıştır.

Ağ performansını değerlendirdiğimizde LXD konteyner tipi sanallaştırma çözümü KVM hipervizör tipi sanallaştırma çözümünden %17,77 daha iyi performans göstermiştir. Bu sonucun ortaya çıkmasında LXD'nin ağ oluşumunda ve yalıtımında Linux ağ isim uzaylarını kullanmasıdır. KVM de Linux çekirdeğinin özelliklerini kullanmasına rağmen bir hipervizör yükü bulunmaktadır. Bu hipervizör yükü ağ üzerinde gecikmelere sebep olmaktadır.

İşlemci performansını değerlendirdiğimizde LXD konteyner tipi sanallaştırma çözümü KVM hipervizör tipi sanallaştırma çözümünden %0,6 daha iyi performans göstermiştir. Fakat performans farkı göz ardı edilebilir seviyededir. LXD ve KVM sanallaştırma teknolojilerinin benzer sonuçlar vermesi her iki sanallaştırma teknolojisinin Linux çekirdeği seviyesinde sanallaştırma sağlamalarıdır. Her iki sanallaştırma çözümü de işlemci zamanlaması için üzerinde bulundukları Linux çekirdeğinin işlemci zamanı algoritmasını kullanmaktadırlar.

Bellek performansını Stream kıyaslama aracı ile değerlendirdiğimizde KVM hipervizör tipi sanallaştırma çözümünün "add" işlemi için %2,8, "copy" işlemi için %5,17, "scale" işlemi için %4,3, "triad" işlemi için %3,05 LXD'den daha iyi performans gösterdiği ortaya çıkmıştır. Fakat performans farkı göz ardı edilebilecek seviyededir. LXD ve KVM sanallaştırma teknolojilerinin benzer sonuçlar vermesi her iki sanallaştırma teknolojisinin Linux çekirdeği seviyesinde sanallaştırma sağlamalarıdır. Bellek yönetiminde her iki sanallaştırma çözümü de Linux çekirdeğinin bellek yönetimini kullanmaktadır.

Disk performansını Fio kıyaslama aracı ile sıralı okuma ve sıralı yazma açısından değerlendirdiğimizde LXD sanallaştırma teknolojisinin sıralı okuma iş yükünde bant genişliğinde %15,5, saniyedeki giriş çıkış sayısında %16,74 ve sıralı yazma iş yükünde bant genişliğinde %49,75 ve saniyedeki giriş çıkış sayısında %50 KVM'den çok daha iyi performans sergilemiştir. Sıralı okuma ve sıralı yazma teknikleri yüksek boyutlu ve yüksek çözünürlüklü büyük dosyalar okurken ve yazarken kullanılan tekniklerdir. Burada LXD konteyner tipi sanallaştırma çözümünün gücü ortaya çıkmaktadır. Giriş çıkış zamanlayıcısı için hem KVM hem de LXD sanallaştırma çözümü üzerinde bulundukları işletim sisteminin varsayılan giriş çıkış zamanlayıcısını kullanmaktadır. Bu sonucun ortaya çıkmasında KVM üzerindeki hipervizör yükü sebep olmuştur.

Disk performansını Fio kıyaslama aracı ile rasgele okuma ve rasgele yazma açısından değerlendirdiğimizde KVM sanallaştırma teknolojisinin rasgele okuma iş yükünde bant genişliğinde %52,96, saniyedeki giriş çıkış sayısında %50,62 ve rasgele yazma iş yükünde bant genişliğinde %52,55 ve saniyedeki giriş çıkış sayısında %52,37 LXD'den çok daha iyi performans gösterdiği görülmüştür. Rasgele okuma ve rasgele yazma teknikleri küçük boyutlu dosyalar okurken ve yazarken kullanılan tekniklerdir. Veri tabanları bu tekniği kullanan sistemlerdendir. Burada da KVM hipervizör tipi sanallaştırma çözümünün gücü ortaya çıkmaktadır. Bu sonuç OpenStack bulut altyapısını LXD ile bütünleşik olmasını sağlayan nova-lxd eklentisinin sisteme ek yük getirmesinden kaynaklanmaktadır.

Bu sonuçlar bulut altyapısında sadece bir sanallaştırma çözümü değil de karma sanallaştırma çözümlerinin gerekliliğini ortaya koymuştur. Ayrıca LXD daha yeni bir sanallaştırma çözümü

olmasına rağmen çok iyi performans sonuçları vermiştir. LXD işletim sistemi seviyesinde sanallaştırma sağladığı için disk alanından da KVM hipervizörüne göre daha avantajlı konumdadır. Gelecekte kullanıcıların uygulamalarını taşıyacağı bulut altyapısını seçerken bulut altyapısının çoklu sanallaştırma teknolojilerini desteklemesi özelliği büyük bir etken olacaktır.

KAYNAKLAR

1. Awasthi, S., Pathak, A. ve Kapoor, L. (2016) Openstack-paradigm shift to open source cloud computing & its integration, 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 112-119. doi:10.1109/IC3I.2016.7917944

2. Barik, R. K., Lenka, R. K., Rao , K. R. ve Ghose, D. (2016) Performance analysis of virtual machines and containers in cloud computing, 2016 International Conference on Computing, Communication and Automation (ICCCA), Noida, 1204–1210. doi:10.1109/CCAA.2016.7813925

3. Casalicchio, E. ve Perciballi, V. (2017) Measuring Docker Performance: What a mess!!!, ICPE '17 Companion: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion, L'Aquila, Italy, 11-16. doi:10.1145/3053600.3053605

4. DeMuro, J. (2018). Konu: What is container technology?. Erişim Adresi: https://www.techradar.com/news/what-is-container-technology (Erişim Tarihi: 17.05.2019).

5. Feitelson, D. G. (2006) Experimental Computer Science:The Need for a Cultural Change, Academic Paper, School of Computer Science and Engineering The Hebrew University of Jerusalem, Israel

6. Felter, W., Ferreira, A., Rajamony, R. ve Rubio, J. (2015) An updated performance comparison of virtual machines and Linux containers, 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, 171-172. doi:10.1109/ISPASS.2015.7095802

7. Graber, S. (2016). Konu: LXD 2.0: Introduction to LXD [1/12]. Erişim Adresi: https://stgraber.org/2016/03/11/lxd-2-0-introduction-to-lxd-112/ (Erişim Tarihi: 17.05.2019).

8. Gupta, S. ve Gera, D. (2016) A Comparison of LXD, Docker and Virtual Machine, International Journal of Scientific & Engineering Research, 7(9), 1414-1417.

9. IBM Developer, (2018). Konu: Cloudbench (CBTOOL). Erişim Adresi: https://developer.ibm.com/open/projects/cloudbench-cbtool/ (Erişim Tarihi: 17.05.2019).

10. Kang, M., Kang, D., Walters, J. P. ve Crago, S. P. (2017) A Comparison of System Performance on a Private OpenStack Cloud and Amazon EC2, 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, 310–317. doi:10.1109/CLOUD.2017.47

11. Kozhirbayev, Z. ve Sinnott, R. O. (2017) A performance comparison of container-based technologies for the Cloud, Future Generation Computer Systems, 68, 175-182. doi: 10.1016/j.future.2016.08.025

12. Li, Z., O'Brien, L., Cai, R. ve Zhang, H. (2012) Towards a Taxonomy of Performance Evaluation of Commercial Cloud Services, 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI , 344–351. doi:10.1109/CLOUD.2012.74

13. Li, Z., O'Brien, L. ve Kihl, M. (2016) DoKnowMe: Towards a Domain Knowledge-driven Methodology for Performance Evaluation, ACM SIGMETRICS Performance Evaluation Review, 43(4), 23–32. doi: 10.1145/2897356.2897360

14. Linux Containers, (2019). Konu: LXC Introduction. Erişim Adresi: https://linuxcontainers.org/lxc/introduction/ (Erişim Tarihi: 17.05.2019).

15. Mell, P. ve Grance, T. (2011) The NIST Definition of Cloud Computing, National Institute of Standards and Technology Information Technology Laboratory, doi:10.6028/NIST.SP.800-145

16. Morabito, R., Kjällman, J. ve Komu, M. (2015) Hypervisors vs. lightweight virtualization: A performance comparison, 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, 386-393. doi:10.1109/IC2E.2015.74

17. OpenStack Docs, (2019). Konu: Welcome to OpenStack Documentation. Erişim Adresi: https://docs.openstack.org/pike/ (Erişim Tarihi: 17.05.2019).

18. PerfKitBenchmarker, (2015). Konu: PerfKit Benchmarker. Erişim Adresi: http://googlecloudplatform.github.io/PerfKitBenchmarker/ (Erişim Tarihi: 17.05.2019).

19. Popek, G. J. ve Goldberg, R. P. (1974) Formal requirements for virtualizable third generation architectures, Communications of the ACM, 17(7), 412–421. doi:10.1145/361011.361073

20. Qian, L., Luo, Z., Du, Y. ve Guo, L. (2009) Cloud Computing: An Overview, IEEE International Conference on Cloud Computing, Springer, Berlin, Heidelberg, 5931, 626–631. doi:10.1007/978-3-642-10665-1_63

21. Ruan, B., Huang, H., Wu, S. ve Jin, H. (2016) A Performance Study of Containers in Cloud Environment, Asia-Pacific Services Computing Conference, Springer, Cham, 10065, 343-356. doi:10.1007/978-3-319-49178-3_27

22. Sampathkumar, A. (2013). TigerPrints Virtualizing Intelligent River®: A Comparative Study of Alternative Virtualization Technologies, M.Sc. thesis, Clemson University, Computer Science, Clemson, South Carolina, United States

23. Wang, B., Song, Y., Cui, X. ve Cao, J. (2017) Performance comparison between hypervisor- and container-based virtualizations for cloud users, 2017 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, 684–689. doi: 10.1109/ICSAI.2017.8248375

24. Yemelianov, A. (2017). Konu: Containerization Mechanisms: Cgroups. Erişim Adresi: https://blog.selectel.com/containerization-mechanisms-cgroups/ (Erişim Tarihi: 17.05.2019).