SUNUCU SANALLAŞTIRMA VE UYGULAMA SANALLAŞTIRMA TEKNOLOJİLERİ PERFORMANS KARŞILAŞTIRMASI

Ali DOĞRU

17 14 02 121

YÜKSEK LİSANS TEZİ

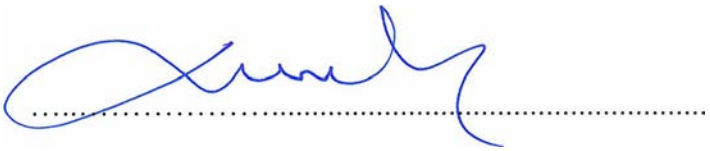
Bilgisayar Mühendisliği Anabilim Dalı
Tezli Yüksek Lisans Programı
Danışman: Prof. Dr. Emin Murat Esin

İstanbul
T.C. Maltepe Üniversitesi
Fen Bilimleri Enstitüsü
Şubat,2019

JÜRİ VE ENSTİTÜ ONAYI

19.02.2019 tarihinde tezinin savunmasını yapan Ali DOĞRU' ya ait "Sunucu Sanallaştırma Ve Uygulama Sanallaştırma Teknolojileri Performans Karşılaştırması" başlıklı çalışma, Jürimiz tarafından Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Tezli Yüksek Lisans Programında Yüksek Lisans Tezi Olarak Oy Birliğiyle kabul Edilmiştir.

Prof. Dr. Emin Murat ESİN (Başkan)
(Danışman)



Prof. Dr. Mesut RAZBONYALI
(Üye)

Prof. Dr. oruç Raif ÖNVURAL
(Üye)

Prof. Dr. İlter BÜYÜKDIĞAN

Enstitü Müdürü

	ŞEKİL ONAY SAYFASI	Doküman No	FR-105
		İlk Yayın Tarihi	20.12.2017
		Revizyon Tarihi	10.12.2018
		Revizyon No	01
		Sayfa	1/2

ŞEKİL ONAY SAYFASI

28/02/2019

MALTEPE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE,

Aşağıda bilgileri bulunan lisansüstü öğrencinin tezi şekil yönünden tarafımca incelenmiş ve Enstitüye teslim edilmesi uygun bulunmuştur.

Anabilim Dalı Başkanı
Dr. Öğr. Üyesi Ali AKMAN

ÖĞRENCİ BİLGİLERİ	
ADI SOYADI	ALİ DOĞRU
ÖĞRENCİ NUMARASI	17 14 02 121
ANABİLİM DALI	BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
PROGRAMI	(X) YÜKSEK LİSANS ( ) DOKTORA ( ) SANATTA YETERLİK
DANIŞMANI	Prof. Dr. Emin Murat Esin
TEZ BAŞLIĞI	SUNUCU SANALLAŞTIRMA VE UYGULAMA SANALLAŞTIRMA TEKNOLOJİLERİ PERFORMANS KARŞILAŞTIRMASI
SAVUNMA TARİHİ	19/02/2019
e-posta	dogru.ali@outlook.com

İç Kapak	☒ Var ☐ Yok
Jüri Onay Sayfası	☒ Var ☐ Yok
Etik İlke ve Kurallara Uyum Beyanı	☒ Var ☐ Yok
İntihal Raporu	☒ Var ☐ Yok
Teşekkür Sayfası	☒ Var ☐ Yok

Hazırlayan İlgili Birim	Kalite Koordinatörü Dr. Öğr. Üyesi Şafak GÜNDÜZ	Kurumsal Yetkili Prof. Dr. Belma AKŞİT

(Doküman No: FR-105; Yayın Tarihi 20.12.2017; Revizyon Tarihi: ; Revizyon No:00)

	ŞEKİL ONAY SAYFASI	Doküman No	FR-105
		İlk Yayın Tarihi	20.12.2017
		Revizyon Tarihi	10.12.2018
		Revizyon No	01
		Sayfa	2/2

Öz (Başlık-Öz-Anahtar Sözcükler)	☒ Var ☐ Yok
Abstract (Title-Abstract-Key Words)	☒ Var ☐ Yok
İçindekiler	☒ Var ☐ Yok
Çizelgeler Listesi	☐ Var ☒ Yok
Şekiller Listesi (varsa)	☐ Şekil yok ☒ Uygundur ☐ Uygun Değildir
Kısaltmalar Listesi	☒ Var ☐ Yok
Tablolar Listesi (varsa)	☐ Tablo yok ☒ Uygundur ☐ Uygun Değildir
Ekler Listesi (varsa)	☐ Ek yok ☒ Uygundur ☐ Uygun Değildir
Özgeçmiş	☒ Var ☐ Yok
Sayfa Genişliği	☒ Uygundur ☐ Uygun Değildir
Yazı Tipi	☒ Uygundur ☐ Uygun Değildir
Referans Kullanımı	☒ Uygundur ☐ Uygun Değildir
Kaynakça Yazımı	☒ Uygundur ☐ Uygun Değildir
Ekler (varsa)	☐ Ek yok ☒ Uygundur ☐ Uygun Değildir

Dr. Öğr. Üyesi Erdal GÜVENOĞLU
İmza

Hazırlayan İlgili Birim	Kalite Koordinatörü Dr. Öğr. Üyesi Şafak GÜNDÜZ	Kurumsal Yetkili Prof. Dr. Belma AKŞİT

(Doküman No: FR-105; Yayın Tarihi 20.12.2017; Revizyon Tarihi: ; Revizyon No:00)

İLKE VE KURALLARA UYUM BEYANI

maltepe üniversitesi	ETİK İLKE VE KURALLARA UYUM BEYANI	Doküman No	FR-178
		İlk Yayın Tarihi	01.03.2018
		Revizyon Tarihi	
		Revizyon No	00
		Sayfa	1/1

Revizyon Takip Tablosu

REVİZYON NO	TARİH	AÇIKLAMA
00	01.03.2018	İlk yayın.

ETİK İLKE VE KURALLARA UYUM BEYANI

22./02./2019

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarından bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilmeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; çalışmanın Maltepe Üniversitesinde kullanılan "bilimsel intihal tespit programı" ile tarandığını ve öngörülen standartları karşıladığını beyan ederim.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

Ali DOĞRU

Hazırlayan	Kalite Koordinatörü	Kurumsal Yetkili
İlgili Birim	Dr. Öğr. Üyesi Şafak GÜNDÜZ	Prof. Dr. Belma AKŞİT

(Doküman No: FR-178; Yayın Tarihi: 01.03.2018; Revizyon Tarihi: ; Revizyon No:00)

İNTİHAL RAPORU

Uygulama ve Sunucu Sanallaştırma Teknolojileri Vmware ve Docker Teknolojilerinin Karşılaştırılması

ORIJINALLIK RAPORU

%8	%7	%5	%
BENZERLIK ENDEKSI	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCI ÖDEVLERI

BIRINCIL KAYNAKLAR

1	sbe.maltepe.edu.tr İnternet Kaynağı	%1
2	www.maltepe.edu.tr İnternet Kaynağı	<%1
3	ab.org.tr İnternet Kaynağı	<%1
4	lib.dr.iastate.edu İnternet Kaynağı	<%1
5	home.austin.rr.com İnternet Kaynağı	<%1
6	www.duo.uio.no İnternet Kaynağı	<%1
7	digitalcommons.unf.edu İnternet Kaynağı	<%1
8	tel.archives-ouvertes.fr İnternet Kaynağı	<%1

Prof. Dr. Emin Murat ESİN
Elektronik & Bilgisayar Müh.

TEŞEKKÜR

Öncelikle yapmış olduğum bu çalışmada bilgi ve tecrübeleri ile çalışmama yön veren ve iş hayatımdan kalan kısıtlı zamanlar için bana zaman ayıran danışmanın Prof. Dr. Emin Murat Esin'e teşekkür etmek istiyorum.

Aynı şekilde çalışmalarım boyunca oluşturduğum test ortamları için gerekli kaynak ve olanakları sağlayan Pegasus Hava Yolları operasyon Müdürü Haluk Yüzücü 'ye teşekkürü borç bilirim.

Bütün bir eğitim ve tez çalışmaları süresince zaman zaman ihmal ettiğim eşime ve kızıma anlayışları için ayrıca teşekkür ederim.

Ali Doğru

Şubat 2019

ÖZ

SUNUCU SANALLAŞTIRMA VE UYGULAMA SANALLAŞTIRMA TEKNOLOJİLERİ PERFORMANS KARŞILAŞTIRMASI

Hazırlayan: Ali Doğru
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Maltepe Üniversitesi Fen Bilimleri Enstitüsü, 2019
Danışman: Prof. Dr. Emin Murat Esin

Bilgisayar dünyasında uzun süredir kullanılan hipervizör tabanlı sanallaştırma teknolojileri ve bulut bilişim ile adı çok daha fazla duyulan konteyner teknolojileri gerek kurumsal veri merkezlerinde gerekse bulut bilişim altyapılarında sıklık ile kullanılan iki teknolojidir. Bu teknolojiler günümüzde kullanılan veri merkezlerinin altyapılarını oluşturmaktadır.

Bu iki teknoloji, verimlilik, yönetim kolaylığı, güvenlik gibi birçok faydalar nedeniyle tercih edilmektedir. Bununla birlikte bu faydaların yanında fiziksel kaynaklar ile işletim sistemleri ve uygulamalar arasında giren sanallaştırma katmanları ve yazılımlar nedeni ile performans kayıpları yaşanmaktadır. Teknolojilerin arasındaki farklılıklar, farklı donanım katmanlarında farklı performans değerlerinin alınmasına neden olmaktadır.

Sanallaştırama teknolojisi seçimi yapılırken uygulama özellikleri ve kullanılacağı iş yüküne göre seçim yapmak elde edilecek faydayı artırmaktadır. Bu nedenle performans kayıp oranlarının donanım kaynakları için ölçülmesi ve karşılaştırılması önemli bir konu olarak karşımıza çıkmaktadır.

Bu çalışmada, sektörde en çok kullanılan Vmware ve Docker teknolojileri üzerine çalışılmıştır. Fiziksel donanımlar ayrı bir etki alanı olarak ele alınıp, her bir etki alanı için ayrı testler yapılmış ve elde edilen sonuçlar karşılaştırılmıştır. Docker konteyner teknolojisinin testlerin büyük çoğunluğunda Vmware çözümünden daha iyi performans gösterdiği görülmüştür.

Anahtar Sözcükler: Sanallaştırma, Linux Konteynerleri, Docker, Hipervizör, Performans,

ABSTRACT

SERVER VIRTUALIZATION AND APPLICATION VIRTUALIZATION TECHNOLOGIES PERFORMANCE COMPARISION

Ali Doğru
Master Thesis
Department Of Computer Engineering
Master Of Computer Engineering Programme
Maltepe University Institute of Science and Technology, 2019
Thesis Advisor: Prof. Dr. Emin Murat Esin

Hypervisor-based virtualization technologies, which have been used for a long time in the IT World and the containerization technologies that mostly used in Cloud computing are two Technologies thats commonly used in enterprise datacenters and Cloud compting datacenters. These two Technologies constitute the infrastructures of the all datacenters today.

These two technologies are preferred because of many benefits such as efficiency, ease of management and security. With these benefits brings performance overhead because of the virtualization layer and software. Differences between technologies lead to different performance values in different hardware layers.

Choosing virtualization technology according to application features and workload that will expected, will increase the benefit to be achieved. For this reason, measurement and comparison of performance overhead rates for each virtualized hardware resources is an important issue.

In this study, we use Vmware and Docker technologies that widely used in the sector. Physical equipment was considered as a separate domain and separate tests were performed for each domain and the results were compared. Docker container technology has shown better performance than most Vmware solutions.

Keywords: Virtualization**,** Containerization, Linux Containers, Docker, Hypervisor, Performance

İÇİNDEKİLER

JÜRİ VE ENSTİTÜ ONAYI i
İLKE VE KURALLARA UYUM BEYANI iv
İNTİHAL RAPORU v
TEŞEKKÜR vi
ÖZ vii
ABSTRACT viii
İÇİNDEKİLER ix
TABLOLAR LİSTESİ xii
ŞEKİLLER LİSTESİ xiii
KISALTMALAR xv
ÖZGEÇMİŞ xvii
BÖLÜM 1. GİRİŞ 1
1.1. Problem 3
1.2. Amaç 5
1.3. Tanımlar 6
BÖLÜM 2. BULUT BİLİŞİM 8
2.1. Bulut Bilişim Tanım 8
2.2. Bulut Bilişimin Tarihi 9
2.3. Bulut Bilişim Hizmet Modelleri 10
2.3.1. Altyapı Hizmeti (IaaS) 11
2.3.2. Platform Hizmeti (PaaS) 11
2.3.3. Yazılım Hizmeti (SaaS) 11
2.4. Bulut Bilişim Yerleşim Modelleri 11
2.4.1. Özel Bulut Modeli 12
2.4.2. Genel Bulut Modeli 12
2.4.3. Hibrit Bulut Modeli 12
2.5. Bulut Bilişim Mimarisi 13
2.5.1. Donanım Katmanı 13
2.5.2. Altyapı Katmanı 13
2.5.3. Platform Katmanı 14
2.5.4. Uygulama Katmanı 14

2.6. Bulut Bilişimin Faydaları 14
BÖLÜM 3. SANALLAŞTIRMA 16
3.1. Sanallaştırma Tarihçesi 16
3.2. Sanallaştırma Teknolojilerinin Sınıflandırılması 19
3.2.1. Tip 1 hipervizör Tabanlı Sanallaştırma 19
3.2.2. Tip 2 hipervizör Tabanlı Sanallaştırma 19
3.3. Konteyner Tabanlı Sanallaştırma 20
3.4. Sunucu Sanallaştırma Teknolojisi 21
3.5. Vmware Sanallaştırma Çözümü 22
3.6. Docker Sanallaştırma Çözümü 23
3.7. Docker ile Vmware Mimari Karşılaştırılması 26
3.8. Docker ve Vmware Avantaj ve Dezavantajları 27
3.9. Sanallaştırmanın Faydaları 27
3.10. Sanallaştırmanın Getirdiği Performans Kayıpları 28
3.10.1. İşlemci Performans Kayıpları 29
3.10.2. Bellek Performans Kayıpları 30
3.10.3. Depolama Alanı Performans Kayıpları 31
3.10.4. Network Performans Kayıpları 31
BÖLÜM 4. PERFORMANS KARŞILAŞTIRMA VE KULLANILAN ARAÇLAR 32
4.1. Performans Karşılaştırma Araçları 32
4.1.1. İşlemci Karşılaştırma Araçları 32
4.1.2. Bellek Karşılaştırma Araçları 34
4.1.3. Sabit Disk G/Ç Karşılaştırma Araçları 36
4.1.4. Network G/Ç Karşılaştırma Araçları 36
BÖLÜM 5. YÖNTEM 38
5.1. Araştırma Modeli 38
5.1.1. Deney Öncesi Hazırlık Aktiviteleri 38
5.1.2. Deney Sonrası Veriler Ve Analizler 41
BÖLÜM 6. BULGULAR VE YORUMLAR 43
6.1. İşlemci Performansı Testleri 43
6.1.1. Veri Sıkıştırma Ölçümleri 43
6.1.2. Veri Sıkıştırma Sonuçlar ve Analiz 44
6.1.3. Kayan Nokta, Şifreleme ve Tam Sayı İşlemleri 45
6.1.4. Kayan Nokta, Şifreleme ve Tam Sayı İşlemleri Sonuçlar ve Analiz 45
6.2. Bellek Performans Testleri 47
6.2.1. Stream Bellek Ölçümleri 47

6.2.2. Stream Bellek Ölçüm Sonuçları 48
6.3. Sabit Disk G/Ç Performans Testleri 49
6.3.1. Bonnie++ Sabit Disk G/Ç Ölçümleri 49
6.3.2. Bonnie++ Sabit Disk G/Ç Sonuçlar ve Analiz 54
6.4. Network Performans Testleri 55
6.4.1. Network Bant Genişliği Ölçümleri 55
6.4.2. Network Bant Genişliği Ölçümleri Sonuçlar ve Analiz 56
6.4.3. Network Gecikme Süreleri Ölçümleri 56
6.4.4. Network Gecikme Süreleri Ölçümleri ve Analizi 58
6.5. LAMP Kümesi Performans Testleri 58
6.5.1. Veri tabanı INSERT İşlemleri 59
6.5.2. Veri tabanı INSERT İşlemleri Sonuçlar ve Analizi 60
6.5.3. Veri tabanı Okuma İşlemleri 61
6.5.4. Veri tabanı Okuma İşlemleri Sonuçlar ve Analizi 64
BÖLÜM 7. SONUÇ 66
EK'LER 69
KAYNAKÇA 122

TABLOLAR LİSTESİ

Tablo 1. Avantajlar ve Dezavantajlar 27
Tablo 2. UnixBench Test Açıklamaları (Google Code Arşiv, 2018) 34
Tablo 3. Performans Test Araçları 41
Tablo 4. Stream Vektör işlemleri. (McCalpin, The STREAM Benchmark, 1996) 47
Tablo 5. Sanal Sunucu Network Bant Genişliği Değerleri 55
Tablo 6. Fiziksel Sunucu Bant Genişliği Değerleri 56
Tablo 7. Docker Konteyneri Bant Genişliği Değerleri 56
Tablo 8. Network Gecikme Değerleri 57
Tablo 9. İnsert İşlemleri Sonuçları 59
Tablo 10. İşlem Sayıları 60
Tablo 11. Ortalama Cevap Süreleri 61
Tablo 12. Veri tabanı Kayıt Sayıları 61
Tablo 13. İşlem Sayısı 62
Tablo 14. Cevap Süresi 62
Tablo 15. Vmware Bellek Ham Sonuçları 69
Tablo 16. Fiziksel Bellek Ham Sonuçları 69
Tablo 17. Docker Konteyner Bellek Ham Sonuçları 69
Tablo 18. PXZ Sıkıştırma Sonuçları Ham Verileri 10^9 bayt 71
Tablo 19. PXZ Sıkıştırma Sonuçları Ham Verileri 10^{10} bayt 71
Tablo 20. Vmware Iperf Ham Verileri 111
Tablo 21. Fiziksel Iperf Ham Verileri 111
Tablo 22. Docker Iperf Ham Verileri 112

ŞEKİLLER LİSTESİ

Şekil 1. Bulut Bilişim Modelleri ve Sorumluluk Dağılımı (Chou, 2010)..... 10
Şekil 2. Bulut Bilişim Yerleşim Modelleri 12
Şekil 3. Bulut Bilişim Mimarisi (Armbrust & diğerleri., 2010) 13
Şekil 4. Hipervizör Tabanlı Sanallaştırma Tipleri (Shujaa, Gania, & Madani, 2016).... 19
Şekil 5. Sunucu Sanallaştırma (Oguchi & Yamamoto, 2008) 22
Şekil 6. Docker İstemci Sunucu Mimarisi (docs.docker.com, 2018) 23
Şekil 7. Docker Uygulama Mimarisi ve Katmanları (Dergan, 2014) 24
Şekil 8. Docker İmaj Oluşturma ve Konteyner Süreci (docs.docker.com, 2018)..... 25
Şekil 9. Sanallaştırma Teknolojileri Mimarileri (de la Fé Herrero & Gutiérrez, 2016) . 26
Şekil 10. Sanal Makine İşlemci Zamanlayıcısı (Hwanju, 2015) 29
Şekil 11. Vmware Bellek Adresleme (Vmware vSphere 5.1 Documentation Center, 2015) 30
Şekil 12. Deney Akış Şeması..... 41
Şekil 13. Veri Sıkıştırma Süreleri 44
Şekil 14. Sıkıştırma Performans Kayıp Oranları 44
Şekil 15. GeekBench Skor 46
Şekil 16. Performans Kayıp Oranı..... 46
Şekil 17. Stream Veri Transfer Hızı Sonuçları 48
Şekil 18. Bellek Performans Kayıpları 48
Şekil 19. 16 GB Veri Okuma Yazma Değerleri 50
Şekil 20. 16 GB Veri Seti İşlemci kullanım Oranları 50
Şekil 21. 32 GB Veri Okuma Yazma Değerleri 51
Şekil 22. 32 GB Veri Seti İşlemci kullanım Oranları 51
Şekil 23. 32 GB Veri Seti İşlemci kullanım Oranları 52
Şekil 24. 64 GB Veri Okuma Yazma Değerleri 52
Şekil 25. 64 GB Veri Seti İşlemci Kullanım Oranları 53
Şekil 26. Performans Kayıp Oranları..... 54
Şekil 27. Performans Kayıpları..... 57
Şekil 28. Lamp Stack (Hsu, 2017)..... 59
Şekil 29. Thread Sayısı ile Cevap Süresi Değişimi 60

Şekil 30. Thread Sayısı ile Yapılan İşlem Sayısı 60
Şekil 31. Docker Cevap Süresi Dağılımı – 100 Thread 62
Şekil 32. Sanal Makine Cevap Süresi Dağılımı – 100 Thread 62
Şekil 33. Docker Cevap Süresi Dağılımı – 200 Thread 63
Şekil 34. Sanal Makine Cevap Süresi Dağılımı – 150 Thread 63
Şekil 35. Docker Cevap Süresi Dağılımı – 200 Thread 63
Şekil 36. Sanal Makine Cevap Süresi Dağılımı – 200 Thread 63
Şekil 37. Docker Cevap Süresi Dağılımı – 250 Thread 64
Şekil 38. Sanal Makine Cevap Süresi Dağılımı – 250 Thread 64
Şekil 39. Docker Aktif Thread Sayısı ve Cevap Süresi 64
Şekil 40. Sanal Makine Aktif Thread Sayısı ve Cevap Süresi 64

KISALTMALAR

KVM	**:** Kernel-based Virtual Machine
LXC	: Linux Containers
AWS	: Amazon Web Services
GAE	: Google App Engine
HPC	: High-Performance Computing
LAMP	: Linux Apache Mysql PHP, PERL, Python
NIST	: National Institute of Standards and Technology
IP	: Internet Protocol
API	: Application Program Interface
REST	: Representational State Transfer
G/Ç	: Giriş/Çıkış
VM	: Virtual Machine
VMM	: Virtual Machine Manager
MMU	: Memory Management Unit
TLB	: Translation Lookaside Buffer
vNIC	: Virtual Network Interface Card
vSwitch	: Virtual Switch
ALU	: Arithmetic Logic Unit
FPU	: Floating-point Unit
LZMA	: Lempel–Ziv–Markov Chain Algorithm
PXZ	: Parallel XZ
SAS	: Serial Attached SCSI
AES	: Advanced Encryption Standar
ECS	: Experimental Computer Science
DoKnowMe	: Domain Knowledge-driven Methodology
Mbps	: Megabits Per Secoınd
GB	: Gigabyte

MB/s	: Megabyte Per Second
Rpm	: Revolutions Per Minute
MTU	: Maximum transmission unit
ms	: Milisaniye
µs	: Microsaniye

ÖZGEÇMİŞ

Ali Doğru

Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Eğitim

Derece	*Yıl*	*Üniversite, Enstitü, Anabilim/Anasanat Dalı*
Y.Ls.	2019	Maltepe Üniversitesi, Fen Bilimler Enstitüsü Bilgisayar Müh. Anabilim Dalı
Ls.	2015	Anadolu Üniversitesi, İşletme ve İktisat Fakültesi İşletme Bölümü
Ls.	2005	Ege Üniversitesi, Mühendislik Bilimler Enstitüsü Tekstil Mühendisliği Anabilim Dalı
Lise	2000	Kdz. Ereğli Anadolu Lisesi

İş/İstihdam

Yıl	*Görev*
2011-	Linux Sistem Mühendisi. Pegasus Hava Taşımacılığı A.Ş
2009-2011	Uygulama ve Sunucu Destek Müh. CITIBANK A.Ş
2007-2009	Uygulama ve Yazılım Destek Müh. Avea İletişim Hizmetleri A.Ş
2005-2007	Teknik Destek Uzmanı. Bosch Sanayi ve Ticaret A.Ş.

Kişisel Bilgiler

Doğum yeri ve yılı : Kdz. Ereğli 1982 Cinsiyet: Erkek

Yabancı diller : İngilizce

GSM / e-posta : 0553 543 5857 / dogru.ali@outlook.com

BÖLÜM 1. GİRİŞ

Günümüzde bilişim alanındaki teknolojik gelişmeler, internet hızlarındaki artış ve işlemci, bellek hızlarında ve güçlerindeki artışlar, bilgisayarların ve akıllı telefonların günlük hayatın içine çok daha fazla girmesine neden olmuştur. Akıllı telefonlar yüksek internet hızı ile her an ve her yerden ihtiyaç duyduğumuz verilere erişim sağlamamıza olanak sunmaktadır. Otobüs ile yolculuk yaparken sinema bileti alabilmekte veya eve doğru yürürken evimize yiyecek siparişi verebilmekteyiz. Günlük hayattaki bu kullanımlar ve endüstride işlenecek olan verilerin boyutlarının büyümesi, daha yüksek kaynak ihtiyacı olan yazımların ve uygulamaların geliştirilmesine ve yaygınlaşmasına neden olmuştur. Bu talepleri karşılamak için gerek bilişim altyapılarında kullanılan sunucu sayılarında, gerekse sunucularda kullanılan fiziksel kaynaklarda büyük artışlar olmuştur.

Altyapı kaynaklarındaki bu gelişme ve büyüme beraberinde yeni problemler getirmiştir. Bunların başında sunucuların yönetimi, güncellenmesi, yedeklenmesi, çeşitli bakım işlemleri, veri merkezlerinde kapladıkları alanlar, elektrik tüketim maliyetleri gibi birçok yönetimsel ve finansal problemler ortaya çıkmıştır. Aynı zamanda altyapılara yapılan yatırımların verimli kullanılması ve mevcut kapasiteden maksimum verim nasıl alınabilir sorularını da ortaya çıkarmıştır. Bu problemlerin ve sorunların çözümü için, endüstride çeşitli çözümler geliştirilmiş ve beraberinde yeni teknolojiler ortaya çıkmıştır. Bunların en önemlileri sanallaştırma çözümleri ve Bulut Bilişimdir.

Bulut Bilişim günümüzde en çok konuşulan ve büyük küçük birçok firma tarafından tercih edilen bir teknoloji olarak karşımıza çıkmaktadır. Bulut bilişim uygulamaların, verilerin, sunucuların internet üzerinden servis olarak sunulmasını sağlayan teknolojidir.

Bulut bilişim, arka planda bulunan sunucular, depolama cihazları, network cihazları gibi kaynakların yönetimi, kurulumu ve konfigürasyonu gibi kompleks işlerden kullanıcıları soyutlayarak, internet üzerinden erişebilecekleri basit bir ara yüz üzerinden istekleri doğrultusunda ihtiyaçlarını kendi kendilerine tanımlayıp yönetebilecekleri bir ortam sağlamaktadır. (Voorsluys, Broberg, &Buyya, 2011)

Bulut bilişimin tercih edilmesinin birçok nedenleri vardır. Bunlardan biride işletim maliyetleridir. Klasik veri merkezi kullanımlarında bir donanım yatırım yapılıp satın alındığında, o kaynak genel olarak günün belirli saatlerinde tam, yarım ya da çok daha az bir kapasite ile çalışmaktadır, kalan saatlerde ise kullanılmamakta ve âtıl kapasite olarak kalmaktadır. Bulut bilişimde ise ihtiyaç duyulan servislerin, sunucuların, uygulamaların sadece kullanıldıkları süre, kaynak büyüklüğü saniyede yapılan işlem adeti gibi parametrelere bağlı olarak ücretlendirilmektedir. Bu da uygulamaların ya da sunucuların âtıl saatlerde çok daha az bir maliyet ile çalıştırılmasına olanak sağlamaktadır.

Bulut bilişim, kaynakların, kullanım oranı, cevap süresi, günün belirli saatleri ve kaynak kullanım oranları gibi ölçülebilir ve tanımlanabilir değerlere göre, herhangi bir kesinti ve kullanıcı müdahalesi olmadan dinamik olarak yönetilebilmesini sağlamaktadır. Aynı zamanda veriye internet üzerinden istenildiği zaman ulaşılabilmektedir. Bulut teknolojisinin sunduğu bu kolaylıkların temelinde ise yine sanallaştırma teknolojisi bulunmaktadır. (Herrod, 2010)

Sanallaştırma teknolojisi (Server Virtualization) tek bir sunucu üzerinden birden fazla işletim sistemini çalıştırmaya olanak sağlayan teknolojidir. (Vaughan-Nichols, 2006) Yine aynı şekilde farklı servisleri ve uygulamaları tek bir sunucu üzerinde çalıştırılmasını sağlayarak yönetimsel masrafları azaltır ve kaynakların daha etkili şekilde kullanılmasını sağlar. Birden fazla, az kaynak kullanan sunucuları bir ya da daha fazla fiziksel sunucu üzerinde güvenli bir şekilde birleştirerek veri merkezlerinde kullanılan alanlardan da tasarruf sağlar. (Smith & Nair, 2005)

Sanallaştırma fiziksel kaynakların maksimum verim ile kullanılmasına olanak sağlamaktadır. Farklı iş yüklerinin tek bir platform üzerinde çalıştırılabilmesi, yönetimsel olarak, kaynak ataması açısından ve maliyetler açısından büyük faydalar sağlamaktadır. (Apparao P., 2007)

Sanallaştırmanın birçok çeşidi vardır. Gerek bulut bilişim sağlayıcıları gerekse kurumsal veri merkezlerinde farklı sanallaştırma teknolojileri kullanılmaktadır. Bu teknolojiler bulut bilişimde daha çok konteyner tabanlı sanallaştırma ya da sunucu sanallaştırma olarak karşımıza çıkmaktadır.

1.1. Problem

Sanallaştırmanın sağladığı faydalar ve kolaylıklar ve günümüzde bu kadar çok kullanılması teknoloji üreticileri tarafından üzerine daha fazla düşülmesine neden olmuştur. Birçok teknoloji üreticisi endüstride doğan bu talebe cevap verebilmek ve daha iyi çözümler üretebilmek için sanallaştırma üzerine olan çalışmalarını ağırlık vermişlerdir. Üreticiler sanallaştırma ile birlikte gelen performans kayıplarını en aza indirgemek, daha basit, daha stabil ve daha az yer tutacak çözümler üzerinde çalışıp, kaynaklardan maksimum verimi alan teknolojiler üretmeye çalışmaktadırlar.

Bulut bilişimde çok sık kullanılan konteyner tabanlı sanallaştırma teknolojisi ve kurumsal ve bulut servis sağlayıcıları veri merkezlerinde daha çok kullanılan hipervizör tabanlı sanallaştırma teknolojisi bu çözümlerden ikisidir. Sunucu sanallaştırma alanında Vmware (datanyze.com, 2017), konteyner tabanlı sanallaştırma alanında Docker endüstride öne çıkan firmalardır. Docker aynı zamanda konteynerleştirme teknolojilerinde bir standart olarak kabul edilmektedir.

Bunların yanında Xen, Oracle VM, Microsoft Hyper-V, QEMU gibi hipervizör tabanlı sanallaştırma çözümleri ve KVM, LXC, OpenVZ gibi Konteyner tabanlı sanallaştırma çözümleri de bulunmaktadır.

Bu kadar farklı sanallaştırma çözümünün bulunduğu ortamda gerek bulut bilişimde gerekse kurumsal veri merkezilerinde sanallaştırma teknolojisi seçimi önemli bir problem olarak karşımıza çıkmaktadır. Sanallaştırma teknolojisi tercihi yaparken dikkate alınması gereken birçok faktör vardır. Tercih edilen teknolojinin her uygulamayı desteklemesi, sanallaştırma katmanından kaynaklanan performans kayıplarının oranı, sanallaştırma işleminin güvenliği, yönetim kolaylığı ve teknolojinin bulut bilişime ya da fiziksel sunuculara genişletilebilme özelliği gibi kriterler bunlardan birkaçıdır. (VMware White Papers, 2018).

Firmalar kendi ortamlarında birden fazla sanallaştırma teknolojisi kullanabilmektedirler. Tercihler daha önce belirtilen kriterlere ve kullanılacak olan uygulamaya göre değişebilmektedir. Bu kriterler arasında performans kayıpları birçok firma tarafından ağırlıklı olarak göz önüne alınmaktadır. Performans denildiğinde, fiziksel kaynağın farklı birimlerinin performansı ayrı ayrı ve farklı sonuçlar verebileceği karşımıza çıkmaktadır. Bunun sonucu olarak tercih yapmak daha da zor hale gelmektedir.

Sanallaştırma teknolojileri performans karşılaştırmaları ile ilgili hem akademik hem de endüstrideki teknoloji firmaları tarafından yapılmış birçok araştırma ve raporlar bulunmaktadır.

İsveç Lund Üniversitesinden Zheng Li ve arkadaşları 2017 yılında hipervizör tabanlı bir sanallaştırma teknolojisi ile konteyner tabanlı bir teknolojiyi karşılaştırmış ve hipervizör tabanlı sanallaştırmanın performans verilerinin sadece yükün büyüklüğüne bağlı olarak değişmediğini aynı zamanda yükün cinsine göre de değiştiğini ortaya koymuştur.

Benzer şekilde Wes Felter ve arkadaşları, Amerika IBM Araştırma Laboratuvarlarında, 2014 yılında sıklıkla kullanılan KVM ve Docker teknolojilerini karşılaştırmışlardır. Her bir kaynak için, ayrı yükler oluşturup bunların analizini yapmışlar ve genel olarak Docker teknolojisinin, KVM'den daha iyi performans elde ettiğini göz önüne sermişlerdir.

Delft Üniversitesi Teknoloji bölümünden Alexandru Iosup ve arkadaşları, bulut bilişim dünyasında sıkça kullanılan Amazon Web Services (AWS) ve Google App Engine ürünlerinin performans karşılaştırmalarını yapmışlardır. Bu iki ürün üzerinde çalışan uygulamaları, uzun bir zaman boyunca 3 ay kadar izlemişler önemli veriler elde etmişlerdir. Bu iki ürün arka planda hipervizör ve konteyner tabanlı sanallaştırma teknolojilerini ya da bunların hibrit halini kullanmaktadırlar. Yaptıkları araştırmada, sanallaştırılmış ortam üzerinde çalıştırılacak uygulamanın türünün tercih yapılırken önemli bir etkiye sahip olduğunu göstermişlerdir. Örnek olarak GAE üzerinde çalıştırılan veri tabanı hizmeti, AWS üzerinde çalışan veri tabını hizmetinden daha iyi performans göstermiştir.

2015 yılında IMS Mühendislik Koleji Bilgisayar Bilimleri departmanından Prof. Ann Mary Joy, yaptığı çalışmada Linux Konteynerleri (LXC) ile sanal makine performanslarını karşılaştırmış ve LXC'nin geleneksel sanallaştırma teknolojilerine göre daha az performans kayıpları olduğunu ortaya koymuştur.

PUCRS Üniversitesinden Xavier M. ve arkadaşları, hipervizör tabanlı sanallaştırma ile konteyner tabanlı sanallaştırma teknolojilerini Yüksek başarımlı bilgisayar sistemleri (HPC) üzerinde karşılaştırmışlardır.

Konteyner tabanlı sanallaştırma teknolojisinin, Yüksek Başarımlı Sistemler üzerinde performansını derinlemesine ölçmüşler ve konteyner tabanlı sanallaştırma çözümünün fiziksel sunucular kadar iyi performans gösterdiğini görmüşlerdir. Fakat güvenlik açısından ve kaynakların yönetimi açısından eksiklikler görmüşlerdir.

2007 yılında HP Laboratuvarlarında Padala Pradeep ve arkadaşları, sunucu konsolidasyonu amacı ile Xen sanallaştırma çözümü ile OpenVZ çözümünü karşılaştırmışladır. Yaptıkları çalışmalar göstermiştir ki ortamda sanallaştırılan uygulama sayısı 4 kat arttığında, hipervizör tabanlı XEN için yanıt süreleri %400 artarken, OpenVZ için bu rakam %100 olarak ölçümlenmiştir.

Yine 2007 yılında Clarkson Üniversitesinden, Jeae Neefe Matthews ve arkadaşları Xen, Vmware İş istasyonu ile Solaris konteynerleri, OpenVZ teknolojilerini karşılaştırmıştır. Bunun için, işlemci yoğun, bellek yoğun, disk yoğun ve network gönderme-alma yoğun olmak üzere 6 farklı stres testi uygulamışlardır. Çalışma sonucunda çok yüksek yükler altında Vmware'in daha kararlı sanal makineler çalıştırabildiğini tespit etmişlerdir. Kaynak izolasyonu açısından konteyner tabanlı sanallaştırma daha iyi sonuçlar vermiştir.

2012 yılında Oslo Üniversitesi Koleji'nden Naveed Yaqub Vmware ve KVM teknolojilerini performans kayıpları açısından karşılatırmış ve 6 farklı stres testi uygulamıştır. Testleri sonucunda Vmware ve KVM arasındaki performans kayıp oranları test süresince kullanılan veri, paket boyutlarına ve çalışma süresine göre değişiklik göstermektedir. Vmware bazı durumlarda %200, bazı durumlarda %15 daha iyi performans göstermiştir.

1.2. Amaç

İşlemci, bellek, network, sabit disk gibi sunucu donanımlarındaki yeni gelişmeler ve bunların sanallaştırma teknolojileri ile uyumluluğu her geçen gün değişmektedir. Bundan dolayı sanallaştırma teknolojileri arasında performans kayıp oranları değişmektedir.

Bu araştırmada hedeflenen sunucu sanallaştırma market lideri Vmware ve Konteyner tabanlı sanallaştırma market lideri Docker teknolojilerini güncel donanımlar ve yazılımlar üzerinde çalıştırarak işlemci, bellek, sabit disk gibi fiziksel kaynaklar, üzerinde farklı senaryolar ile performans verilerini karşılaştırıp, karar verme aşamasında değerlendirilmek üzere güncel bir analiz sunmaktır. Endüstride sıkça kullanılan LAMP (Linux, Apache, MySQL, PHP) kurulumu üzerinde farklı büyüklüklerdeki yük altında nasıl davrandığı değerlendirilip, analiz sunulmuştur.

1.3. Tanımlar

Hipervizör (hypervisor): Türkçe çevirisi hiper yönetici olan bir yazılımdır. Bir fiziksel sunucu üzerindeki kaynakları sanallaştırarak, birden fazla işletim sisteminin bu kaynaklar ile çalıştırılmasına olanak sağlayan ve kaynak paylaşımını yöneten yazlımdır.

Docker Konteynerleri (Docker Containers): Bir yazılımın tek başına çalışabilmesi için gerekli olan kaynak kod, yürütme kütüphaneleri, sistem araçları, sistem kütüphaneleri ve sistem araçlarının bir paket haline getirilmiş ve bir standart oluşturulmuş halidir.

Sanal makine (Virtual Machine): Bir bilgisayarın programlarının yürütüldüğü fiziksel makine yerine işletim sistemi tarafından kullanıcıya sunulan mantıksal makine gösterimi.

Sanal makine monitörü (Virtual Machine Monitor): hipervizör tabanlı sanallaştırma teknolojilerinde, hipervizör yazılımının bir parçası olarak fiziksel makine üzerinde sanal makinelerinin oluşturulmasını, yönetimini yapan yazılımdır.

İsim Uzayı (Namespace): Ad uzayı olarak Türkçe çevirisi yapılmaktadır. Tanım olarak bir bilgisayar ağında olduğu gibi, paylaşılmış bilgi işleme ortamında belirli nesneleri tanımlamakta kullanılan bir ya da daha fazla adın oluşturduğu bir varlık. 2. C++ programlama dilinde bir değişkenin geçerli olduğu kaplam.

(Kontrol Grupları) Cgroups: Türkçe çevirisi Kontrol Grupları olan bir terimdir. Tanım olarak, bir sistem üzerinde çalışan işlemlerin o işlemlerin sahiplerine göre hiyerarşik bir yapı oluşturan ve bu yapı üzerinde işlemci, giriş-çıkış üniteleri, bellek gibi kaynakların kullanımının sınırlandırılabilmesini, yönetilebilmesini sağlayan çekirdek sistem özelliğidir.

Kökü Değiştir (Change Root-Chroot): Çekirdek sistemin bir özelliği olan Chroot, sistem üzerinde çalışan herhangi bir servis ya da uygulama için, ayrı ve izole bir (/) kök dizin ve alt dizinlerini oluşturarak uygulamanın ya da yazılımın yürütme sırasında ihtiyaç duyduğu kütüphaneleri, sistem programlarını bu oluşturulan yeni dizinlerin altına kopyalar ve buradan çalıştırılmasını sağlar.

Linux Konteynerleri (Linux Containers): Linux işletim sistemi seviyesinde bir sanallaştırma yöntemidir. Kontrol grupları(cgroups), Chroot, SELinux gibi çekirdek sistem özelliklerini kullanarak bir işletim sistemi üzerinde birden fazla Linux sistemi çalıştırılmasını sağlamaktadır.

Güvenli Linux (Security-Enhanced Linux-SELinux): Tam açılımı Security-Enhanced Linux olan bu yazılım Linux Çekirdeğinin güvenlik modülüdür.

Güvenli Bilişim Modu (Secure computing mode-seccomp): Linux Çekirdeğinin bir güvenlik filtresidir. Çalışmakta olan işlemlerden gelen Sistem çağrılarını, adet argüman gibi değerlere göre filtre uygulanmasını sağlamaktadır.

Maksimum İletim Birimi (Maximum Transmission Unit-MTU): Bilgisayar ağlarında çalışan yönlendirici, sunucu network ara yüz kartları gibi cihazlara tanımlanan, o cihazlar üzerinden tek seferde iletilebilecek maksimum network paket boyutunu ifade eder. Bu boyutun üzerindeki paket MTU boyutu ölçüsünde bölünerek iletilir.

İş Parçacığı (Thread): Thread iş parçacığı olarak Türkçeye çevrilmektedir. Performans testlerinde kullanılan thread, senaryosu hazırlanmış ve yapılacak işleri sırası ile yapan iş parçacığı olarak kullanılmaktadır.

BÖLÜM 2. BULUT BİLİŞİM

Bu Bölümde Bulut Bilişim hakkında genel bilgi, tarihi, Bulut Bilişimin mimari özellikleri ve Bulut Bilişim modelleri tanımlanmış ve bunların özellikleri anlatılmıştır.

Evlerimizde elektrikli bir aleti kullanırken yaptığımız iş sadece cihazı duvardaki prize takmak ve açma kapama düğmesine basmaktır. Bunları yaparken, o elektrik enerjisinin nasıl üretildiği, nasıl dağıtıldığı, nasıl evimize kadar getirildiği ile ilgilenmeyiz. Elektrik; bize sanallaştırılmış bir hizmet olarak geldiğinden, arka planda yürütülen üretim, dağıtım, bakım, yenilenme gibi birçok hizmetten haberdar olmayız. Arka plandaki kompleks işler sanallaştırılıp bize bu kaynağı kullanmak için sadece bir duvar prizi sunulmuştur. (Voorsluys, Broberg & Buyya, 2011)

İşte Bulut bilişimin temelinde yatan da bu düşüncedir. Bilişim kaynaklarını, elektrik, su hizmetlerinde olduğu gibi arka plandaki kompleks işlerden soyutlayarak kullanıcılara bilişim hizmeti sunmaktır.

Bu düşünceyi gerçekleştirmek için temel olarak fiziksel kaynakların sanallaştırılması ile mümkün olabilmektedir. Bu nedenle gerek bulut bilişim gerekse büyük veri merkezlerinde sanallaştırma en önemli teknoloji olarak karşımıza çıkmaktadır.

2.1. Bulut Bilişim Tanım

İşlemci ve depolama teknolojilerindeki hızlı gelişmeler ile internet hızlarındaki artış, eskiye göre bilişim kaynaklarının ucuzlamasına, daha güçlü sunucuların kullanılmasına olanak sağlamıştır. Bu gelişmeler beraberinde bulut adı verilen yeni bir bilişim modelini ortaya çıkarmıştır.

Literatürde bulut bilişim için birçok tanım verilmektedir. Bu tanımların karşılaştırıldığı bir çalışma Vaquero L. M ve arkadaşları tarafından yapılmıştır. (Vaquero & diğerleri., 2009). Bu araştırmada 20'den fazla tanım karşılaştırılmıştır

Genel olarak bulut bilişim, bilgisayarlar ve diğer cihazlar için, ihtiyaç duyulduğu zaman ve faklı kullanıcılar arasında paylaştırılan fiziksel kaynakların, internet üzerinden erişilen ve kaynakların basit bir ara yüz kullanılarak tanımlandığı ve talebe göre hızlı bir şekilde artırılıp azaltılabildiği bir teknolojidir. Bu yönü ile bulut bir ürün değil sunulan bir servistir. Bulut bilişimin en çok kabul gören tanımı ise NIST tarafından yapılmıştır.

Bu tanıma göre, Bulut Bilişim; Çok az yönetim çabası ya da hizmet sağlayıcı etkileşimi ile hızlı bir şekilde hazırlanabilen ve kullanıma sunulabilen, paylaştırılmış ve yapılandırılabilen hesaplama kaynakları (Örneğin, network, sunucu, depolama, uygulama ve servis) havuzuna, her yerden, uygun ve isteğe bağlı olarak network üzerinden erişim sağlayan bir modeldir. (Mell & Grance, 2011)

2.2. Bulut Bilişimin Tarihi

Bulut bilişim temelleri bilgisayar tarihi kadar eskidir diyebiliriz. 1950'li yıllarda üniversite ve şirketlerde kullanılan büyük ana bilgisayarlar zayıf istemci ve terminallere aynı anda hizmet verebilmekteydi. Bunun altıda yatan teknoloji ise zaman paylaşımı teknolojisidir. Ana bilgisayar kaynaklarını nanosaniye seviyesinde istemciler arasında paylaştırmaktadır. Buda her istemcinin aynı anda çalışmasına olanak sağlamaktadır. Diğer bir deyiş ile istemciler nanosaniye seviyesindeki süreleri hissedemedikleri için aynı anda çalışılıyor hissi oluşturmaktadır. Bu sayede kaynakların daha verimli kullanılması sağlanmaktadır.

Bulut bilişimin altında yatan ana düşünce ise ilk olarak 1960 yılında John McCarthy tarafından "The Challenge of The Computer Utility" kitabında bahsedilmiştir. Kitapta bilgi işlemin elektrik su gibi kamuya açık ve her yerden erişilebilen bir servis olarak sunulacağından bahsetmiştir.

Kelime olarak bulut telekomünikasyon firmalarının networklerini belirtmekte, geniş ATM networklerini tanımlamada, kullanılmıştır. 2006' da Google CEO su Eric Schmid tarafından internet üzerinden servislerin sunulduğu iş modeli olarak kullanılmıştır. (Zhang , Cheng & Boutaba, 2010)

Günümüzde bildiğimiz kavram olarak bulut bilişim ilk olarak 2002 yılında Amazon Web Servisleri olarak ortaya çıkmıştır. Amazon firması veri merkezlerini taşıma kararı aldığında, mevcut kapasitesinin sadece %10 nu kullandığını ve geri kalan kapasitenin ise yoğun ya da kampanya dönemleri için hazır bekletildiğini görmüşler. Bu kullanılmayan kapasiteyi önce şirket içindeki diğer bölümlerin hizmetine sunmuşlar. Daha sonra diğer firmalara servis olarak kullandırma yoluna giderek bulut bilişimi kavramını endüstriye sunmuş oldular.

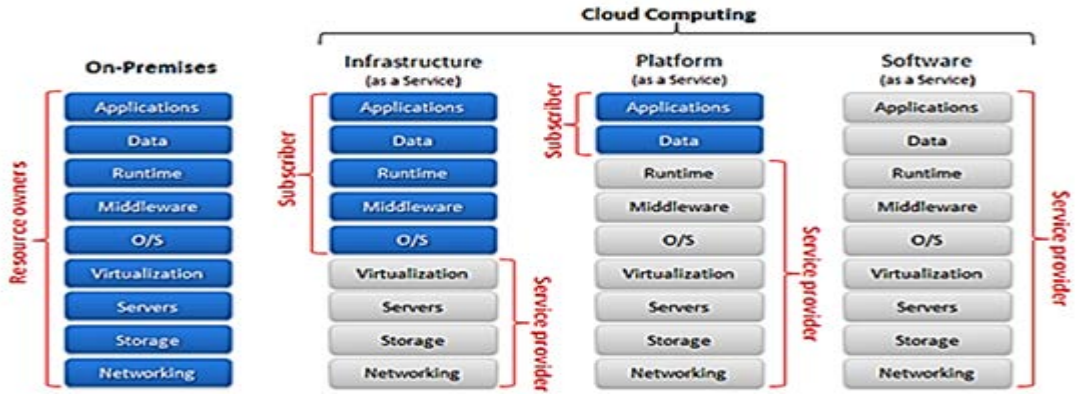
Amazonun getirdiği diğer bir yenilik ise bu kaynakları kullanımı fiyatlamak için, ortaya attığı "kullandıkça ödeme" modeli olmuştur. Bu model günümüzde bulut bilişim için standart haline gelmiştir.

Bulut bilişimin yaygınlaşması ile teknoloji firmaları bu alanda yatırımlarını artırmıştır. 2010 yılında Microsoft Azure servisini, 2011 yılında IBM SmartCloud servisini ve 2012 yılında Oracle Cloud sevisini hizmete sunmuştur.

Ticari amaçlı servislerin yanında Openstack, OpenNebula, Eucalyptus, Apache CloudStack açık kaynak kodlu bulutlaştırma ürünleri de geliştirilmiştir. (Cordeiro & diğerleri., 2011)

2.3. Bulut Bilişim Hizmet Modelleri

Bulut bilişim servis sağlayıcıları, bulut hizmetlerini üç ana modele göre sunmaktadırlar. (Voorsluys, Broberg & Buyya, 2011) Altyapı Hizmeti (IaaS), Platform Hizmeti (PaaS) ve Yazılım Hizmeti (SaaS). Sırası ile her model, bulut hizmeti alan şirketlere ya da kişilere daha az iş ve sorumluluk bırakmaktadır. Hizmet alan tarafın ve servis sağlayıcıların sorumluluk dağılımları aşağıda şekilde gösterilmiştir.



Şekil 1. Bulut Bilişim Modelleri ve Sorumluluk Dağılımı (Chou, 2010)

Şekil 1'de görüldüğü üzere sunucuların bulut servis sağlayıcıları yerine, kaynaklara sahip taraflar tarafından kendi veri merkezlerinde (On-Premises) işletildiği durumlarda uygulamalar, veri, ara katman yazılımları, İşletim sistemleri, sanallaştırma katmanları, fiziksel sunucular, depolama alanları ve network altyapısının sorumluğu tamamen kendi sorumluluklarında olmaktadır. Bunların işletilmesi, bakımı, güncellemesi tamamen kendilerine ait olmaktadır.

Bulut bilişime geçildiği takdirde, her bir bulut modelinde bu sorumlukların bir kısmı ya da tamamı servis sağlayıcıya devredilmektedir. Bu sözü edilen modeller aşağıda açıklanmıştır.

2.3.1. Altyapı Hizmeti (IaaS)

Bulut bilişimin en temel modelidir. Bulut servis sağlayıcıları, kullanıcılara kendi veri merkezlerinden fiziksel ya da sanal makinalar tahsis ederler. Bu sunucuların ya da aygıtların yönetimi, bakımı yedeklenmesi gibi operasyonel görevler hizmeti alan tarafa aittir. Yine altyapı hizmeti içinde, sanal makine imajları, çeşitli depolama alanları, IP adresleri, güvenlik duvarı, yük dengeleyiciler gibi altyapısal ürünler de sunulmaktadır. (Amies, Tong & Sluiman, 2012)

2.3.2. Platform Hizmeti (PaaS)

Platform hizmetinde, bulut servis sağlayıcı işletim sistemini, çalıştırılmak istenen program için gerekli uygulama ve işleme platformunu sunmaktadır. Hizmet alan taraf uygulama ve veriyi sağlamak açısından sorumludur.

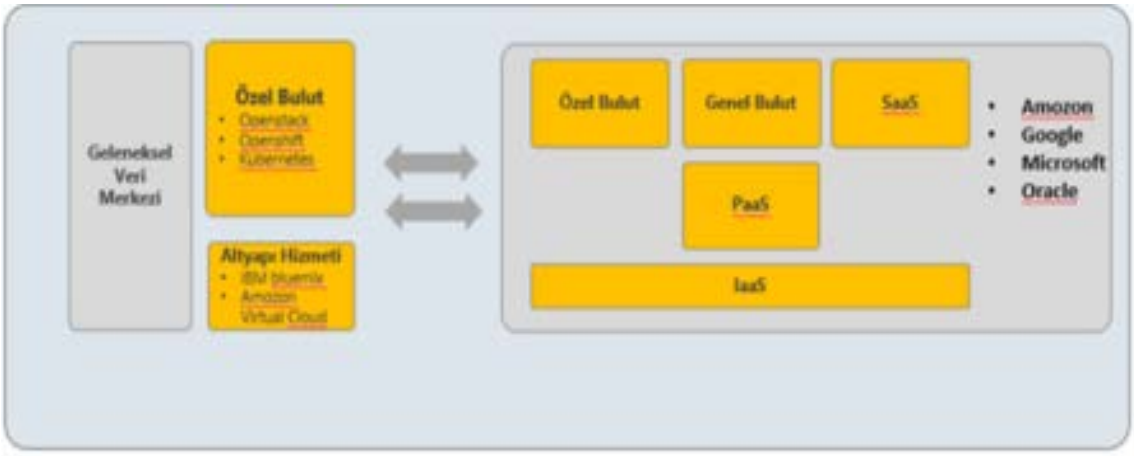
2.3.3. Yazılım Hizmeti (SaaS)

Yazılım hizmetinde bulut servis sağlayıcı uygulamaların yüklenmesi, işletilmesi görevlerini yapar. Böylelikle kullanıcının herhangi bir kurulum yapmasına gerek kalmamaktadır.

2.4. Bulut Bilişim Yerleşim Modelleri

Bulut bilişimde şirketlerin kullanımı ve bulut hizmetinin alındığı veri merkezine bağlı olarak üç çeşit yerleşim tipi vardır. Özel Bulut, Genel Bulut ve Hibrit Bulut. Şirketler ihtiyaçları doğrultusunda kendileri için, en uygun olan yerleşim modelini seçerek bulut bilişimden maksimum faydayı sağlamaya çalışmaktadırlar.

Bunların dışında Topluluk Bulutu, Yüksek Performans Hesaplama Bulutu, Büyük Veri Bulutu gibi belirli amaç ve toplulukları için oluşturulmuş modeller de vardır. Şekil 2'de Bulut Yerleşim Modelleri gösterilmiştir.



Şekil 2. Bulut Bilişim Yerleşim Modelleri

2.4.1. Özel Bulut Modeli

Özel bulut modeli, tekbir şirkete hizmet eden bulut modelidir. (Mell & Grance, 2011) Şirketlerin veri merkezlerinin ya da altyapı sağlayıcılarından kiralanan ya da satın alınan kaynakların bulut yazımları ile sanallaştırılarak bulut esnekliğinin kazandırılmasıdır. Bu modelde altyapı yönetim masrafları şirket tarafından karşılanmakta, bulut bilişimin ana faydalarından olan altyapı maliyetleri azaltma hedefinden uzaklaşılmaktadır.

2.4.2. Genel Bulut Modeli

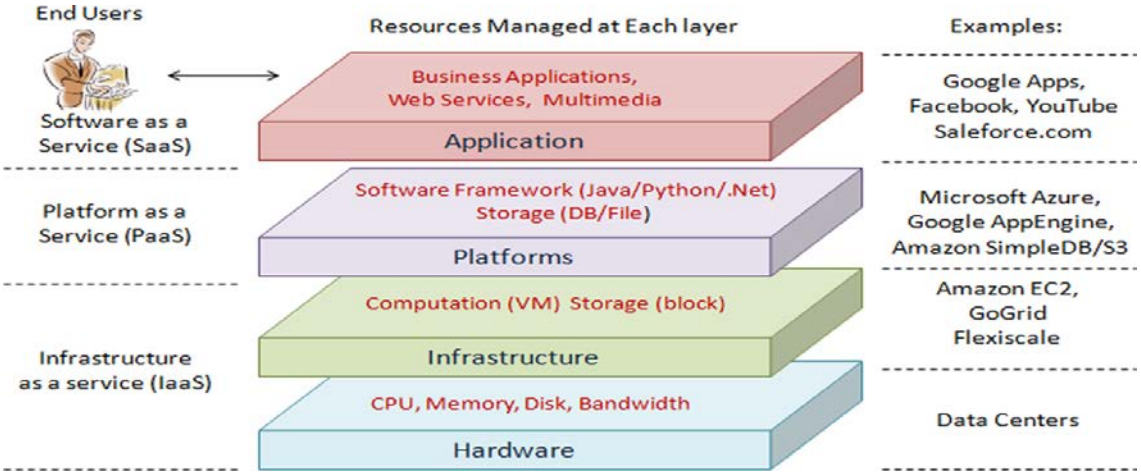
Genel bulut Modelinde, bulut hizmet sağlayıcıları, depolama, sunucu ve diğer kaynaklarını bütün kullanıcılar ile paylaşmaktadır. Teknolojik olarak özel bulut modelinden çok büyük farkları olmasa da ortak kullanım olduğu için gizlilik ve güvenlik açısından farklılıklar barındırmaktadır.

2.4.3. Hibrit Bulut Modeli

Bu model, özel ve genel bulut modellerinin aynı anda şirketler tarafından kullanıldığı modeldir. (Stevens, 2011) Şirketler güvenlik, işleme gücü gibi ihtiyaçları için kendi özel bulutunu kullanırken, dönemsel artışların yaşandığı, şirket dışında, dünyanın farklı yerlerinde bulunan müşterilerine hizmet verdiği servislerini genel bulutta konumlandırabilmektedir.

2.5. Bulut Bilişim Mimarisi

Bulut bilişim servis modelleri tanımlanırken bulut bilişimin mimari katmanları baz alınarak yapılmıştır. Donanım katmanı, Altyapı Katmanı, Platform Katmanı ve Uygulama Katmanı olmak üzere dört adet katmandan söz edebiliriz. Bu katmanlar örnekleri ile birlikte Şekil 3'de gösterilmiştir.



Şekil 3. Bulut Bilişim Mimarisi (Armbrust & diğerleri., 2010)

Mimari olarak bu katmanların hangi bulut servis modelinde kullanıldığı endüstride sık kullanılan örnekleri ile Şekil 3'de gösterilmiştir. Bu sözü edilen katmanlar aşağıda anlatılmıştır.

2.5.1. Donanım Katmanı

Bu katman bulut bilişim veri merkezlerinde bulunan fiziksel sunucular, depolama alanları, network cihazları, güvenlik cihazları, soğutma üniteleri gibi bütün fiziksel cihazları kapsamaktadır. Donanım yenileme, ekleme, çıkarma, bakım işleri ve konfigürasyonları işlemleri bu katmanda yapılan ve kontrol edilen işlerdir.

2.5.2. Altyapı Katmanı

Bu katman bulut bilişimin otomatik kaynak atama, yük dengeleme, elastik ve hızlı bir şekilde yatayda ve dikeyde kaynak artırma gibi en önemli ve anahtar özelliklerinin kazandırıldığı katmandır. Sanallaştırma katmanı olarak da isimlendirilen bu katmanda, KVM, Xen ve Vmware gibi sanallaştırma teknolojileri kullanılmaktadır. (Zhang , Cheng & Boutaba, 2010) Fiziksel kaynaklardan, kaynak havuzları ve fiziksel kaynakların paylaştırılması bu katmanda gerçekleşmektedir.

2.5.3. Platform Katmanı

Altyapı katmanın üzerinde oluşturulan bu katmanda hazır işletim sistemleri ve uygulama sanallaştırma ile Konteynerler içerisinde programların çalıştırılmasını sağlayan çatı uygulamalar sunulmaktadır.

2.5.4. Uygulama Katmanı

Bulut bilişim mimarisinin en üst katmanıdır. Yazılım hizmeti modelinin işletildiği katmandır. Firmalar kendi geliştirdikleri uygulamaları bu katmanda kullanıma açmaktadırlar. Bu katmanda servis tamamen sağlayıcı tarafından verilmekte kullanıcı tarafından herhangi bir operasyonel işlem yapılmamaktadır.

2.6. Bulut Bilişimin Faydaları

Bulut bilişiminin birçok faydası bulunmaktadır. Bu faydalar şirketlerin kullanmak istedikleri hizmet modeline ya da yerleşim modeline göre değişebilmektedir. Ortak faydaları sıralarsak;

Self servis: Bulut bilişim, kaynaklara istediğimiz zaman, istediğimiz şekilde özelleştirerek erişmemize ve bunların ödemelerini hesaplarken ve yaparken herhangi bir insan müdahalesine ya da yardımına ihtiyaç duymadan tamamen self servis olarak yapmamıza olanak sağlar. (Mell & Grance, 2011)

Kullandığın kadar ödeme: Bulut bilişim, verilen servislerin ve kaynakların masraflarını kullanım miktarlarına göre hesaplamaktadır. Kullanıcıya bu kaynakları hızlı ve kolay bir şekilde artırıp azaltmasına olanak sağladığından ihtiyaç olmayan ya da kullanmadığı kaynakların ücretini ödemek zorunda kalmamaktadır. (Voorsluys, Broberg & Buyya, 2011)

Esneklik: Bulut bilişim kullanıcıya sanallaştırma teknolojisi sayesinde sınırsız kaynak izlenimi verir. Kullanıcı kendi isteği doğrultusunda web ara yüzü aracılığı ile kaynakları kendisi artırıp azaltabilir. Kaynakların yük altında dinamik bir şekilde artırılabilmesi ya da azaltılabilmesi olanak sağlamaktadır. (Mell & Grance, 2011).

Bulut bilişimin anahtar faydalarına bakıldığı zaman bu faydalara olanak sağlayan teknolojinin sanallaştırma teknolojisi olduğunu görüyoruz. Bulut hizmetlerinin alt yapısı çok büyük veri merkezlerinde binlerce fiziksel sunucu ve cihazların sanallaştırma teknolojisi ile kolaylıkla yönetilebilir duruma getirilmesi ile mümkün olmaktadır.

Bulut hizmet sağlayıcılarından, Platform, Yazılım ya da Altyapı hizmeti alırken sağlayıcıların kullandığı sanallaştırma teknolojileri farklılık gösterebilmektedir. Yazılım ve Platform servislerinde genellikle Konteyner tabanlı sanallaştırma, altyapı hizmetlerinde ise hipervizör tabanlı sanallaştırma kullanılmaktadır.

BÖLÜM 3. SANALLAŞTIRMA

Bu Bölümde Bulut bilişimin temelinde yatan sanallaştırma teknolojisinin, tarihi, gelişimi, sanallaştırma tipleri ve farkları anlatılmıştır.

Bulut bilişimin altında yatan anahtar teknoloji sanallaştırma teknolojisidir. (Li & diğerleri., 2017) Bulut bilişimin sağladığı birçok fayda bu teknoloji sayesinde mümkün olmaktadır. Sanallaştırma için birçok tanım kullanılmıştır. En çok kullanılan ve kabul gören tanımlar şunlardır;

Sanallaştırma, fiziksel kaynakların farklı yöntemler kullanılarak, işletim sistemleri ve kullanıcılardan soyutlanması demektir. Sanallaştırma fiziksel kaynakların işlevlerini bir yazılım (hipervizör) ya da Linux çekirdeği seviyesinde benzetim yaparak sanal bilgisayar sistemi oluşturmaktır diyebiliriz. Sanallaştırma ile esneklik ve kaynakları daha verimli kullanılması sağlanmakta aynı zamanda aynı fiziksel kaynak üzerinde farklı işletim sistemleri (Linux, Windows, Solaris, vs.) çalıştırılmasına olanak sağlanmaktadır.

Hipervizör ya da diğer adı ile sanal makine monitörü, fiziksel donanım üzerinde sanal makinelerin ve sanal kaynakların oluşturulmasını ve yönetilmesini sağlayan yazılımdır. Hipervizör tabanlı sanallaştırma bu yazılım sayesinde mümkün olmaktadır.

Diğer bir sanallaştırma yöntemi ise konteyner tabanlı sanallaştırmadır. Bu yöntemde Linux çekirdeği içinde bulunan “namespace”, “cgroups” gibi fonksiyonlar kullanılarak aynı çekirdek sistem üzerinde sanal kaynakların oluşturulmasıdır.

3.1. Sanallaştırma Tarihçesi

Sanallaştırma kavramı en az bilgisayar kavramı kadar eskidir. İlk olarak 1950’lerde işlemci kaynaklarını daha verimli kullanmak fikri üzerinden konuşulmaya başlanmıştır. 1960’ın başlarında işletim sistemlerinde toplu işlem (batch processing) teknolojisi ve ardından 1967 yıllarından sonra zaman paylaşımı (time sharing) teknolojisi ile sanallaştırmanın temelleri atılmış oldu. Zaman paylaşımı teknolojisi birden fazla uygulamanın ve birden fazla kullanıcı tarafından aynı anda kullanılmasına olanak sağlamaktadır. Her kullanıcı o fiziksel kaynağın tamamen kendine ya da kullandığı uygulamaya ait olduğunu düşünmektedir.

Bunun nedeni fiziksel kaynağın nanosaniye süresinde kullanıcılar arasında paylaştırılması ve bu sürelerin kullanıcılar tarafından anlaşılmasının mümkün olmamasından kaynaklanmaktadır.

Yalnız bu teknoloji ile beraberinde bazı problemler getirmiştir. Bunlardan biri, bir uygulamanın hata alması bütün sistemi ve bütün kullanıcıların işlemlerini etkilemesidir. Sistemin kararlılığını artırmak için uygulamaların birbirinden yalıtılması gerekmektedir.

Uygulamaların birbirinden yalıtılması, o günlerin teknolojisi ile her uygulama için ayrı bir fiziksel sistem kullanılması ile olmuştur. Bu çok pahalı bir çözüm olmasının yanında zaman paylaşım teknolojisi olmadığı için sistem kaynaklarının boşa harcanmasına neden olmaktadır. (Marinescu & Kröger, 2007) Bu problem, yeni fikirler ve teknolojilerin geliştirilmesine neden olmuştur. Yazılım geliştirmeleri sonucunda talimat ile öğretim simülasyonu (instruction-by-instruction simulation) geliştirilmiştir. Bu teknoloji her uygulama için, mevcut sistemin donanım ve yazılım kopyasını oluşturarak, kullanıcıların kullanımına sunmaktadır. Hatta bu teknoloji üzerinde kullanıcılar sadece uygulamalarını değil komple bir işletim sistemini bile kullanabilmektedir. Bu sanal makine düşüncesinin ortaya çıkış ana fikridir.

Marinescu ve arkadaşlarına göre (Marinescu & Kröger, 2007) sanal makine ile uygulamaların birbirinden yalıtılması 20 ye 1 oranında performans kaybına neden olmaktadır. Simülasyon yazılımlarının performansının izlenmesi ve sistemlerin verimliliğini artırmak için 1960'larda IBM tarafından IBM virtual monitör/370 sanal makine monitörü tanıtılmıştır.

1970ler boyunca sanallaştırma teknolojisi maliyet etkinliği nedeni ile kullanımı genişledi ve geliştirildi. 1980 ve 1990 yılları arasında donanım maliyetlerinin ucuzlaması ve kişisel bilgisayarların yaygınlaşması sanallaştırma teknolojisinin eskisi kadar geliştirilmemesine ve ilgilenilmemesine neden olmuştur. (Soltesz & diğerleri., 2007)

1990'lı yıllarda Stanford Üniversitesinde bir araştırma projesi olarak ele alınan sanal makine teknolojisi, Vmware şirketi ile birlikte yeniden şekillendirilip, geliştirilmeye başlanmıştır.

Tekrar popüler olmaya başlayan bu teknoloji, XenSource, SWsoft, Microsoft, Oracle gibi birçok teknoloji firmasının bu alana yönelmesine neden olmuştur. Sadece yazılım firmaları değil Intel ve AMD gibi donanım üreticileri ürünlerinde sanallaştırma desteği vermişlerdir.

Sanal makine ve sanallaştırma teknolojisi hipervizör tabanlı sanallaştırma olarak bilinmektedir. Sanal makine tarihsel gelişimi içinde, günümüzde Konteyner tabanlı sanallaştırma olarak bilinen farklı bir teknolojide geliştirilmiştir.

Bu teknolojinin temelinde çekirdek sistem seviyesinde işlemlerin ve uygulamaların birbirinden yalıtılması yatmaktadır. İlk olarak 1979 yılında Unix V7'de "Chroot" ile birlikte tanıtılmıştır. 1982'de, her işlemi ve onların dosya erişimlerini birbirinden yalıtan "Chroot" sürümü Linux BSD sürümündeki çekirdek sisteme eklenmiştir. 2000'li yıllarda FreeBSD içinde güvenliği artırılmış sürümleri geliştirilmiş ve 2001 yılında ile dosya sistemi, network adresi ve ana belleği ayrı ayrı tanımlanmasına olanak sağlayan Linux VServer geliştirilmiştir. 2006 yılına kadar geliştirilmesi ve desteği devam etmiştir.

2004 yılında klon alınmasını kolaylaştırılmış ve Chroot teknolojisinin bazı problemlerinin çözülmüş hali olan Solaris Konteynerleri (Biederman, 2006), 2005 Yılında Linux çekirdek sistemine yama olarak eklenen Open VZ geliştirilmiştir.

2006 yılında Google tarafından bir grup işlemin kaynak kullanımlarını sınırlamak, hesaplamak ve yalıtmak için Kontrol grupları(cgroups) geliştirilmiş ve 2.6.24 versiyonlu Linux çekirdek sistemine eklenmiştir.

2008 yılında LXC (Linux Konteyner) ile tam olarak işletim sistemi seviyesinde sanallaştırma yapılmıştır. Linux kontrol grupları teknolojisine dayanan bu sistem ile işlemci, bellek, dosya sistemi, network gibi kaynak kullanımları yalıtılmış ve stabil hale getirilmiştir.

2013 yılında Dotcloud tarafından Docker açık kaynak kodlu olarak geliştirilmeye başlanmıştır. Linux Konteyner (LXC) tabanlı olarak başlayan bu teknoloji 2014 yılında firmanın kendi geliştirdiği kütüphane Konteyner teknolojisi tabanlı olarak geliştirilmeye başlanmıştır. Docker Konteyner tabanlı sanallaştırma dünyasında en çok tercih edilen üründür.

3.2. Sanallaştırma Teknolojilerinin Sınıflandırılması

Sanallaştırma teknolojileri, sanallaştırmanın yapıldığı yani, sanallaştırma katmanın kullanıldığı katmana ve kullanılan teknolojiye göre üçe ayrılmaktadır.

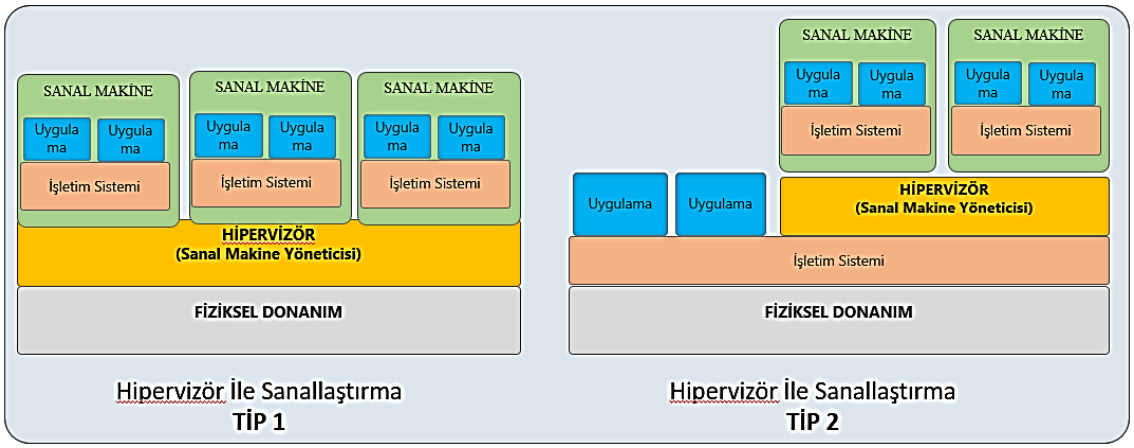
- Hipervizör Tabanlı Sanallaştırma
  - Hipervizör Tip 1
  - Hipervizör Tip 2
- Konteyner tabanlı Sanallaştırma (İşletim Sistemi Seviyesi)

3.2.1. Tip 1 hipervizör Tabanlı Sanallaştırma

Bu teknolojide hipervizör yazılımı arada herhangi bir işletim sistemi olmadan fiziksel makinenin üzerine kurulmaktadır. Fiziksel kaynakların sanallaştırılması bu katmanda yapılıp, sanal makinelere bu katmandan verilmektedir. Örnek olarak ESX Server, Xen verilebilir.

3.2.2. Tip 2 hipervizör Tabanlı Sanallaştırma

Bu yöntemde hipervizör bir uygulama gibi işletim sistemi üzerine kurulur. Hipervizör fiziksel kaynaklara işletim sistemi üzerinden erişir. Kurulumu kolay olan bu yöntem daha çok kişisel kullanım ve test amaçlı kullanılmaktadır. Marinescu ve arkadaşlarına göre bu yöntem %30 oranında performans kaybına neden olmaktadır. (Marinescu & Kröger, 2007) Hyper-V, Oracle Virtualbox, Vmware Workstation Tip 2 sanallaştırma örnekleridir. Sanallaştırma tipleri Şekil 4'te gösterilmiştir.



Şekil 4. Hipervizör Tabanlı Sanallaştırma Tipleri (Shujaa, Gania, & Madani, 2016)

Şekil 4'te görüldüğü üzere Tip 1 ve Tip 2 sanallaştırma tiplerinde en büyük fark sanallaştırma yazılımının yani hipervizör katmanının nerede olduğudur. Tip 1 'de sanallaştırma yazılımı fiziksel kaynaklara direk ulaşırken, Tip 2'de direk ulaşamamaktadır. Tip 2'de çalışmakta olan bir işletim sisteminin fiziksel kaynak erişimi sırası ile Sanal işletim sistemi, Tip 2 hipervizör, fiziksel sunucunun işletim sistemi, Fiziksel kaynak şeklinde olmaktadır.

3.3. Konteyner Tabanlı Sanallaştırma

Konteyner tabanlı sanallaştırma işletim sistemi seviyesinde yapılan, Linux çekirdek sistem içinde oluşturulan Linux konteynerleri ile yapılmaktadır. Hipervizör tabanlı sanallaştırmaya göre daha hafif bir sanallaştırma tekniğidir. (Xavier & diğerleri., 2013) . Konteyner içinde çalışan uygulamalar ve işletim sistemleri kurulu olduğu sistemin çekirdek sistemini ortak kullanmaktadır. Fiziksel makinenin kaynaklarının sanal ortamlara paylaştırılması, her sanal ortam örneği için oluşturulan soyutlanmış kullanıcı isim uzaylarında (user-namespace) yapılmaktadır.

Konteyner tabanlı sanallaştırmada sanal makineler, kurulu olduğu sistem üzerinde bir işlem olarak çalışmakta ve fiziksel kaynaklara hipervizör tabanlı sanallaştırmadan farklı olarak sistem çağrıları ile erişmektedir. Bu teknolojide işlemlerin birbirinden soyutlanması çekirdek sistem isim uzayları sayesinde yapılmaktadır. (Biederman, 2006)

Çekirdek sistem isim uzayları, çalışmakta olan her bir işlemin sistem kaynaklarını farklı olarak görmesini sağlamaktadır. Her bir işlem kendine tanımlanan, atanan kadar kaynakları görmekte ve diğer işlemlerin ve global sistemin kaynaklarına, çevresel değişkenlerine erişememektedir. Konteyner tabanlı sanallaştırmada her bir sanal ortam ayrı bir işlem olarak çalıştığı için bu sistemler birbirinden bu sayede yalıtılmış olmaktadır.

Konteyner tabanlı sanallaştırmada kaynak yönetimi yine Linux çekirdek sistemin bir özelliği olan kontrol grupları (cgroups) ile yapılmaktadır. (Menage , 2006) Kontrol grupları çalışmakta olan işlemler ve işlem grupları için kaynak kullanımı sınırlandırılmasına olanak sağlamaktadır. Bu sayede Linux çekirdeği içerisinde bir işlem olarak çalışan konteynerlerin kaynak kullanımları, atamaları kontrol edilebilmektedir.

Linux konteynerleri temel olarak Linux çekirdek sisteminin aşağıdaki özelliklerini kullanmaktadır. (Linux Containers, 2018)

- İsim Uzayları (ipc, utc, mount, pid, network and user),
- Kontrol grupları (cgroups)
- Chroot
- Güvenli Linux (SELinux) ve Güvenli Bilişim Modu (seccomp)

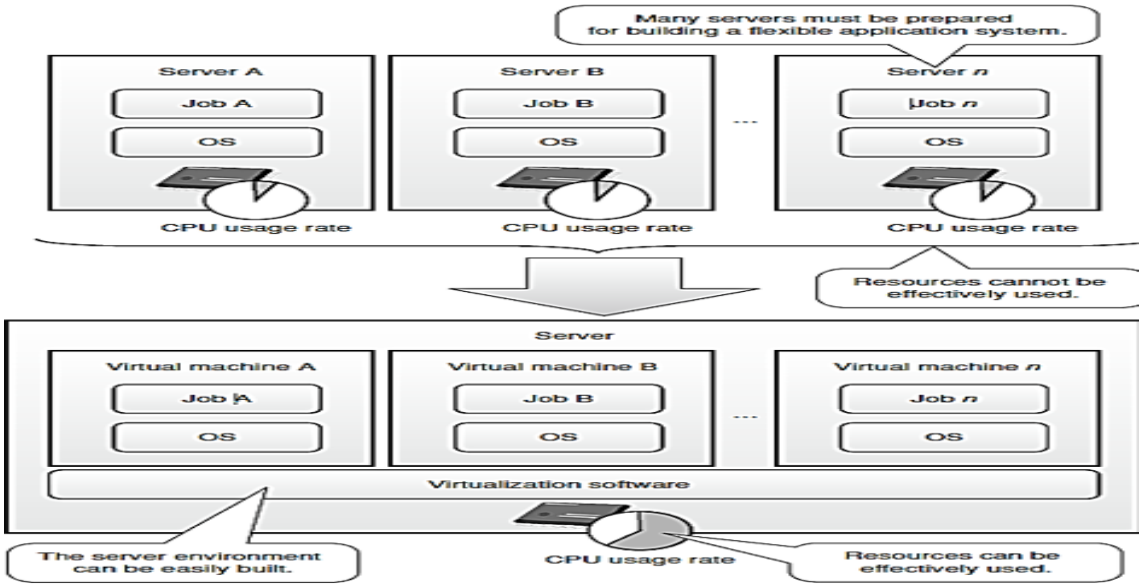
Endüstride en çok kullanılan sanallaştırma teknolojilerine örnek olarak Linux-VServer, OpenVZ, LXC, Solaris Konteynerleri, FreeBSD Jails ve Docker örnek olarak verilebilir.

3.4. Sunucu Sanallaştırma Teknolojisi

Sunucu sanallaştırma düşüncesi Stanford Üniversitesi Bilgisayar Bilimleri profesörü Dr. Mendel Rosenblum ‘un sunucularda ve sistem servislerinde verimliliğin ve kullanım oranının artırılması için bir teknik olarak makine sanallaştırma konusu araştırması ile başlanmıştır.

Sunucu sanallaştırma teknolojisinin sunucu platformlarının kurulum maliyetlerini düşürmesi hedeflenmiştir. Bunlar, yer, zaman, insan gücü, barındırma, enerji gibi maliyetler olabilir. Bunun yanında fiziksel kaynakların kullanım oranlarını maksimize ederek mevcut fiziksel kaynakların çok daha verimli kullanılmasını sağlamaktadır. (Oguchi & Yamamoto, 2008)

Farklı özelliklerde ve gereksinimleri olan sunucular kurmak yerine, bir fiziksel sunucu üzerine sanal makineler kurulması kurulum maliyetlerini düşürmekte ve İşlemci, bellek ve network kartı gibi fiziksel kaynakların kullanım oranlarını artırmaktadır. Şekil 5’te sunucu sanallaştırma ve fiziksel kurulum arasındaki fark gösterilmiştir.



Şekil 5. Sunucu Sanallaştırma (Oguchi & Yamamoto, 2008)

Şekil 5'te görüldüğü üzere, esnek bir uygulama sistemi oluşturulabilmesi ve bunların birbirinden en az şekilde etkilenmesini sağlamak için n adet uygulama ya da iş için n adet fiziksel sunucu ve işletim sistemi kurulumu gerekmektedir. Bu şartlar sağlandığında ise bu sunucuların yönetimi, bakımı zorlaştığı gibi işlemci ve diğer kaynakların kullanım oranları gerçek kapasitelerinin çok çok altında kalabilmektedir.

Sunucu sanallaştırma ile bir fiziksel sunucu üzerinde, birden fazla işi ya da uygulamayı çalıştıracak sunucular kurulabilir. Bu sayede tek bir fiziksel kaynak üzerinden yönetim yapılıp, fiziksel kaynakları daha verimli şekilde kullanılabilmektedir.

3.5. Vmware Sanallaştırma Çözümü

Sanallaştırma teknolojisinin, daha önce belirttiğimiz, yüksek kullanım, yönetim kolaylığı, izole edilmiş güvenli çalışma alanı gibi faydaları sağlaması sektörde bu teknolojiye olan talebin artmasına neden olmuş ve birçok teknoloji şirketinin bu alanda çalışmalar yapmasına yol açmıştır. Dünyanın en büyük sanallaştırma yazılımı, platformu ve aracı sağlayıcısı durumunda olan Vmware bu şirketlerin en önde gelenidir.

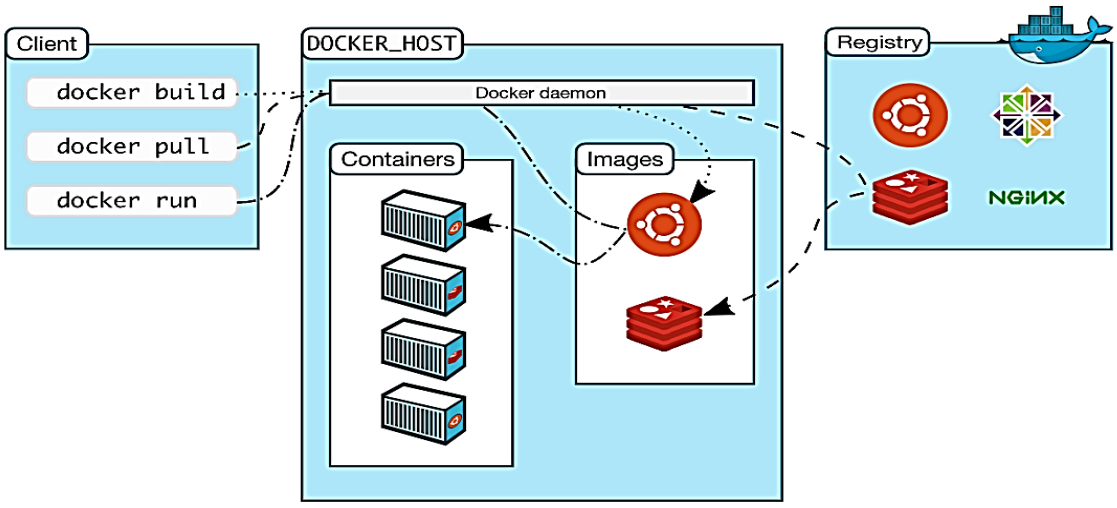
Vmware genel olarak Tip 1 ve Tip 2 sanallaştırma çözümlerini sunmaktadır. Endüstride profesyonel çözüm olarak Tip 1 sanallaştırma kullanılmaktadır. Vmware bu sanallaştırma sınıfı için VMWARE ESXi ürünü bulunmaktadır. ESXi Direk olarak fiziksel sunucunun üzerine kurulup, sanal makinelere işlemci, bellek, network gibi fiziksel kaynakları sanallaştırarak ve belirlediğimiz oranlarda vermektedir.

3.6. Docker Sanallaştırma Çözümü

Docker konteyner teknolojisi, işletim seviyesi sanallaştırma sınıfı içerisinde bulunan bir üründür. Diğer işletim sistemi seviyesi sanallaştırma çözümlerinde olduğu gibi LXC yani Linux konteyner teknolojisini kullanmaktadır. Endüstride konteyner pazarının %43 ü Docker çözümünü kullanmaktadır. (Datanyze, 2017) En yakın rakibi olarak Kubernetes %19 ve %9 ile Vagrant çözümü gelmektedir.

Konteynerleri uygulama konteynerleri ve işletim sistemi konteynerleri olarak ayırmak gerekir. Yaygın olarak uygulama konteynerleri kullanılmaktadır. Konteyner teknolojisi 10 yıldır konuşulmasına rağmen Docker bu alanda son yılların en başarılı çözümü olmuştur. (Rad, Bhatti & Ahmadi, Mart 2017) Docker dört ana bileşenden oluşmaktadır. Docker istemcisi ve sunucusu, Docker imajları, Docker kayıt defteri ve Docker konteynerleri.

Docker istemcisi ve sunucusu; Docker istemci sunucu mimarisinde çalışmaktadır. Docker istemcisi ile Docker sunucusu RESTFul API üzerinden konuşur. İstemci ve sunucu aynı makine üzerinde olabileceği gibi başka bir makinedeki sunucuya da istek gönderebilir. Şekil 6'da Docker istemci ve sunucu mimarisi gösterilmiştir.

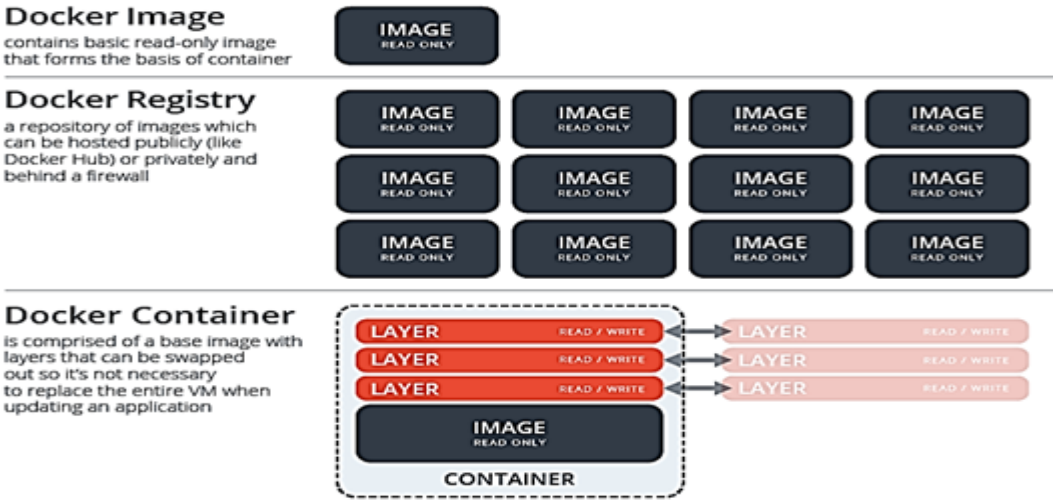


Şekil 6. Docker İstemci Sunucu Mimarisi (docs.docker.com, 2018)

Şekil 6'da görüldüğü üzere, Docker İstemcileri ilk olarak sunucu üzerinde kurulu olan Docker servisi ile konuşmaktadır. Docker istemcisi ile gönderilen komutlar, ilk olarak Docker servisi tarafından karşılanıp, daha sonra Docker konteynerlerine uygulanmaktadır.

Docker İmajları: Docker konteynerleri üzerinde bir uygulama çalıştırmak için ve çalışan bu uygulamanın konteyner haline getirilip dağıtılabilmesi için, uygulamanın temel gereksinimlerinin o konteyner içinde paketlenmiş olması gerekir. Bu noktada Docker imajlarına ihtiyaç duyulur. Docker imajı oluşturmak için iki yol bulunmaktadır.

Bunlardan ilki sadece okuma yetkisi bulunan temel imajları kaynak alarak uygulama için gerekli paketlerin kurulumunu yaparak imaj oluşturmaktır. İşletim sistemi üreticilerinin temel Docker imajları Docker kayıt defterinde ya da kendi sitelerinde indirilebilir içerik olarak bulunmaktadır. Diğer bir yol ise Docker servisinin Docker imajı oluşturmak için okuduğu ve uygulamanın ihtiyaç gereksinimlerin kurulması için gerekli komutları içeren dosya kullanarak oluşturma yoludur.



Şekil 7. Docker Uygulama Mimarisi ve Katmanları (Dergan, 2014)

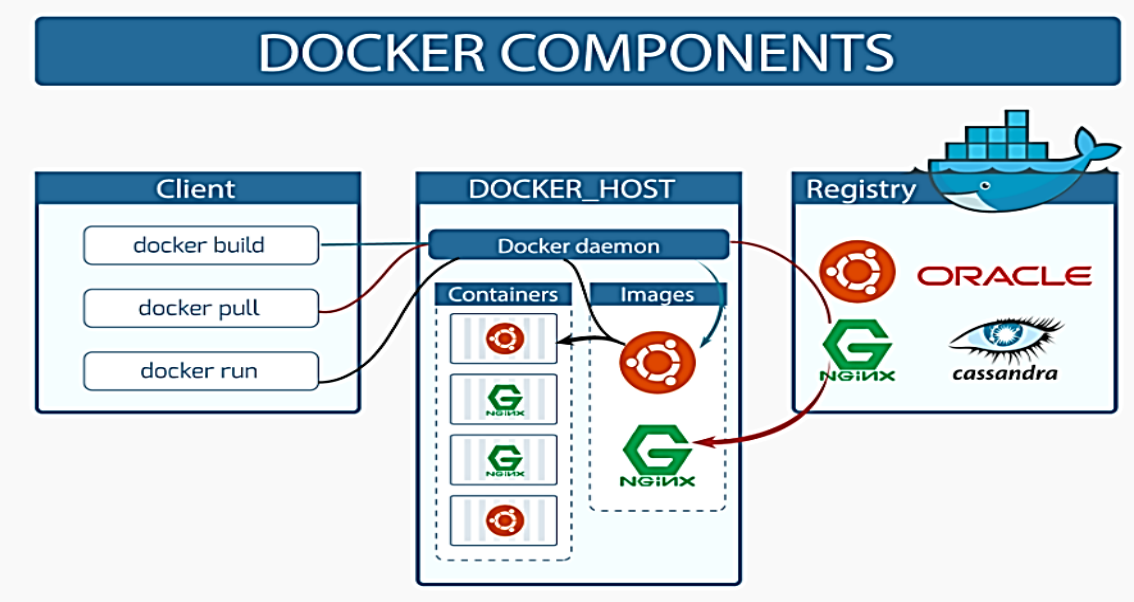
Şekil 7'de görüldüğü üzere Docker imajları, bir konteynerin ve/veya konteyner içinde çalışacak olan uygulamanın en basit, en temel kütüphanelerini içeren katmandır. Bu imajlar tamamen yazma korumalı sadece okuma izni olan katmandır.

Bu imajların gerek genele açık internet kütüphanelerinde gerekse şirket içi özel kütüphanelerde barındırıldığı yerlere Docker kayıt defteri denilmektedir. Bu kayıt defterlerinde farklı işletim sistemleri tabanlı, farklı özelliklerde, versiyonlarda birçok uygulama konteynerleri temel imaj olarak tutulmaktadır.

Docker konteynerleri ise, bu kayıt defterlerinde bulunan temel imajlardan, ihtiyaca göre oluşturulan ve temel imajı değiştirmeden, üzerine yeni yazılabilir katmanların oluşturulduğu yapıdır.

Docker Kayıt defteri: Docker kayıt defteri Docker temel imajlarının ve bunların versiyon, geliştirici bilgisi, yayınlanma tarihleri, indirme sayıları vs. gibi bilgilerinin saklandığı ve herkes ya da belirli kişiler ile paylaşılabildiği depolama alanıdır. Bu kayıt de oluşturulan kişiselleştirilmiş imajların yüklenip, geri çağrıla bilinmektedir. Herkese internet üzerinden erişim izni verilmiş olan ve şirket için ya da özel izinli erişim izinli olarak iki tipi bulunmaktadır. Docker Hub, herkese açık ve doğrulanmış üreticilerin ya da bireysel kullanıcıların oluşturduğu imajların yayınlandığı bir bulut hizmetidir.

Docker Konteynerleri: Docker konteynerleri Docker imajları temel alınarak, bu imajların üzerine oluşturulur. Bir uygulamanın çalışması için gerekli tüm paket ve programları içerir. Örnek olarak bir Apache Tomcat 8 uygulaması için, CentOS 6 işletim sistemi, Java 8 ihtiyacı var ise, bu şekilde oluşturulmuş konteyner çalıştırıldığında, CentOS 6 işletim sistemi üzerinde Java 8 ile çalışan Tomcat Servisi çalışmaya başlamaktadır.



Şekil 8. Docker İmaj Oluşturma ve Konteyner Süreci (docs.docker.com, 2018)

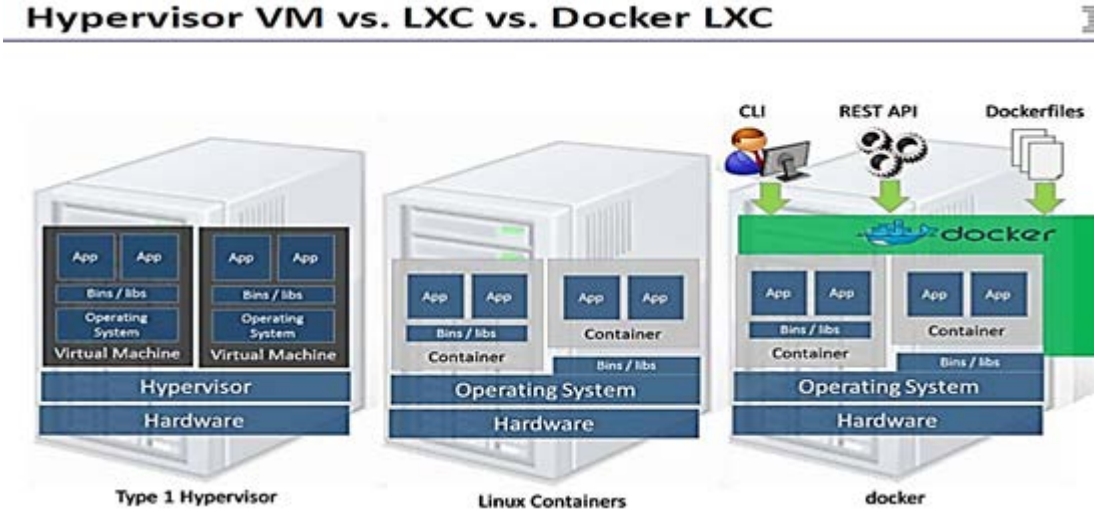
Şekil 8'de Docker üzerinde imaj oluşturma ve konteyner çalıştırma süreci sırası ile gösterilmiştir. Bu süreç aşağıda anlatılmıştır.

- İstemcisi üzerinden "Docker build" komutu, imaj oluşturma komutudur. Bu komutlar ilk olarak Docker sunucusu üzerindeki Docker servisi tarafından alınır.
- Komut içerisinde istenilen imajın, temel imajının olup olmadığı kontrolü için ilk olarak kendi üzerindeki kayıt defterine bakar.

- Kayıt kendi üzerinde yok ise internet üzerinde erişime açık olan Docker kayıt defterine bakar ve bu kayıt defterlerinden “Docker pull” komutu ile temel imajı indirir.
- Daha sonra indirilen imaj üzerinden, istemci tarafından gönderilen “Docker run” ve istenilen parametreler (port, isim, bellek vs. gibi) ile konteyner çalıştırılır.

3.7. Docker ile Vmware Mimari Karşılaştırılması

Görüldüğü üzere Linux konteynerleri, Docker konteynerleri ve hipervizör tabanlı sanallaştırma çözümleri arasında teknolojik, sanallaştırmanın yapıldığı katman ve kullanılan kullanıcı erişim ara yüzleri açısından farklılar görünmektedir. Yapısal olarak farkları Şekil 9’da görülmektedir.



Şekil 9. Sanallaştırma Teknolojileri Mimarileri (de la Fé Herrero & Gutiérrez, 2016)

Şekil 9’da görüldüğü üzere sanallaştırma yapılan seviye ve sanallaştırılan kaynaklar her teknolojide farklılık göstermektedir. Tip 1 hipervizör tabanlı sanallaştırma kullanılarak çalıştırılan bir uygulamanın çalışması için uygulamanın ikili dosyaları ve kütüphaneleri, bütün bir işletim sistemi ve sanal makineye ihtiyaç duyar. Sözü edilen bu sanal makinenin fiziksel kaynak ihtiyacı ise hipervizör üzerinden sağlanır.

Bir Linux konteyneri ya da Docker konteyneri üzerinde çalışan bir uygulamanın çalışması için uygulamanın ikili dosyaları, kütüphaneleri ve fiziksel kaynak üzerindeki işletim sistemi yeterli olmaktadır.

3.8. Docker ve Vmware Avantaj ve Dezavantajları

Docker ve Vmware çözümlerinin gerek sanallaştırmanın yapıldığı katman gerekse yazılım farklılıklarından dolayı avantaj ve dezavantajları bulunmaktadır. Firmaların bilgi işlem stratejilerine ve organizasyon yapılarına göre bu özellikler farklı firmalar için farklı önemler arz etmektedir.

Bu iki çözümün birbirlerinden en çok fark yaratan özellikleri Tablo 2'de gösterilmiştir.

Tablo 1. Avantajlar ve Dezavantajlar

Özellik	**Docker**	**Vmware**
Açılıp Kapanma hızı	Çok Hızlı (milisaniye, saniye)	Görece daha yavaş (saniye, dakika)
Kopya oluşturulması	Çok Hızlı (saniye, dakika)	Görece daha yavaş (dakika, saat)
Fiziksel Kaynakların Kullanımı	Yüksek	Görece daha düşük
Gereksinim Duyduğu Boyut	Az (Mb)	Görece daha fazla (Mb, GB)
Taşınabilirlik	Çok kolay	Görece daha zor.
Lisans Maliyeti	Açık kaynak	Lisans Satın alınmalı
Tam Sanallaştırma Desteği	Yok (Linux Çekirdek Sistem sağlamamaktadır.)	Var (hipervizör sağlamaktadır.)
32 bit Desteği	Yok (Sadece 64 bit)	Var (32 bit ve 64 bit destekliyor)
Güvenlik	Görece daha düşük (İşlem izolasyonu ile)	Yüksek (Sanal ortamlar tamamen izole)

3.9. Sanallaştırmanın Faydaları

Mevcut veri merkezlerinde ve hatta kişisel bilgisayarlarımızda, sanallaştırma teknolojilerini kullanmamızın birçok faydaları bulunmaktadır. Bu faydalar şirket gereksinimlerinin ve kişisel çalışmalarımızın farklılıklarına göre çeşitlilik göstermektedir. Ortak sayılabilecek genel faydaları aşağıdaki gibi sıralanabilir.

- İş yükünü, daha az sayıda fiziksel sunucu üzerinde birleştirerek, sunucu sayısından ve buna bağlı olarak, veri merkezlerinde kullanılan alandan, elektrik ve soğutma gibi giderlerden tasarruf etmemizi sağlar.
- Daha az fiziksel sunucu kullanıldığı için, bakım, onarım ve yönetimsel masraflar azalmaktadır.
- Güvenli olmayan, ya da risk taşıyan uygulamaları oluşturacağımız sanal makineler üzerinde birbirinden izole olarak çalıştırarak daha güvenli bir ortam oluşturulabilir.

- Bir fiziksel sunucu üzerinde sınırlı kaynakları sanallaştırma ile sanal makinelere sınırsız kaynak varmış gibi gösterebiliriz. Buda bazı simülasyonları yapmamıza olanak sağlar. Örnek olarak sınırsız network kartı ile çok büyük ağların simülasyonu yapılabilir.
- Fiziksel makinelerin, donanım sürücü versiyonu ve işletim sistemleri bağımlılığından kurtularak, aynı fiziksel donanım üzerinde çok farklı ve farklı versiyonlardaki işletim sistemlerini çalıştırılabilir.
- İşletim sistemi seviyesinde ya da uygulamalar seviyesinde yapılacak olan değişiklikler veya güncellemeler üretim ortamına uygulanmadan önce test ortamlarında kolaylıkla test edilebilir. Uygulama öncesinde sistemin sanallaştırma araçları ile geri dönüş noktası oluşturularak problem durumunda hızlı bir şekilde önceki durumuna geri döndürülebilir.
- Yedekleme, kaynak artırımı, taşınma işlemleri sanallaştırma araçları ile herhangi bir kesinti olmadan, ya da çok kısa bir süre içerisinde gerçekleştirilebilir.

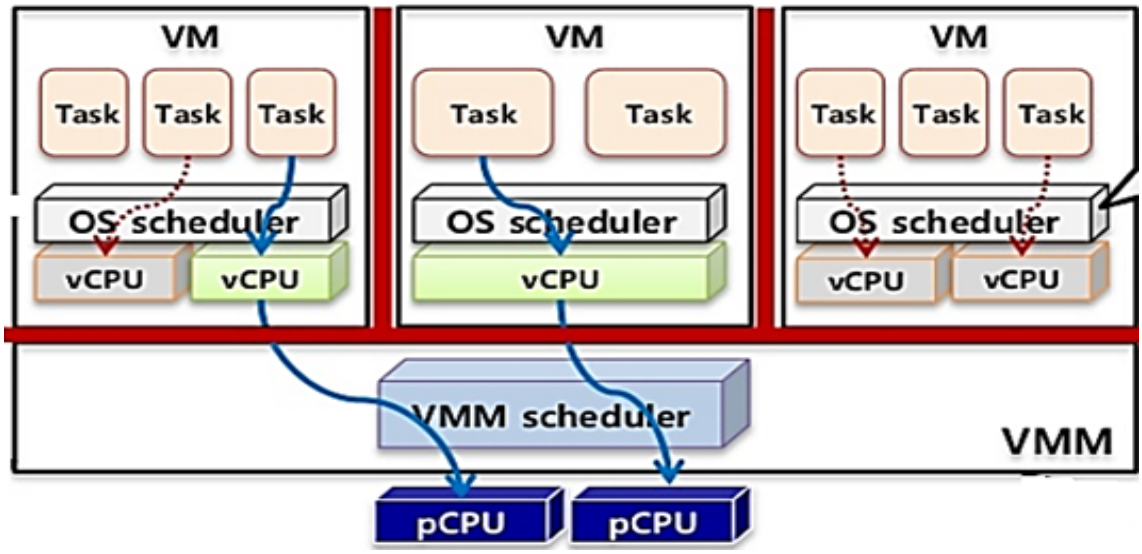
3.10. Sanallaştırmanın Getirdiği Performans Kayıpları

Sözü edilen iki teknoloji fiziksel bir donanım üzerinde çalışmaktadır. Uygulamaların ya da işletim sistemlerinin bu fiziksel kaynakları kullanabilmesi için ara bir katman oluşturmakta ve bu katman üzerinden kaynak yönetimi yapmaktadırlar. Fiziksel kaynaklar ile uygulamalar arasına giren bu katmanın diğer bir değişle sanallaştırma yazılımının fiziksel makineye göre performans kayıpları olmaktadır. (Fuertes, 2012) Kaynaklara erişimde iki farklı teknoloji, iki farklı yöntem kullanıldığı için performans kayıpları da birbirlerinden farklılık gösterebilmektedir.

Performans kayıpları İşlemci, G/Ç üniteleri (Sabit Disk, Ağ kartı) ve Bellek kaynaklarına erişimde yaşanmaktadır.

3.10.1. İşlemci Performans Kayıpları

Fiziksel donanım ve sanal makineler arasındaki sanallaştırma katmanı mecburen bir performans kaybına neden olmaktadır. Sanallaştırma ile işlemcilerin kullanım oranları ve verimlilikleri artırılıyor olsa da tek bir kaynak birden fazla sanal makine ile paylaştırıldığından sanal makine başına erişilebilir olan fiziksel işlemci kaynağı azalmaktadır. (Tong & diğerleri., 2011) Şekil 10'da sanallaştırılmış bir sistemde İşlemci zamanlayıcısı ve sanal makine monitörü zamanlayıcısının çalışması gösterilmiştir.



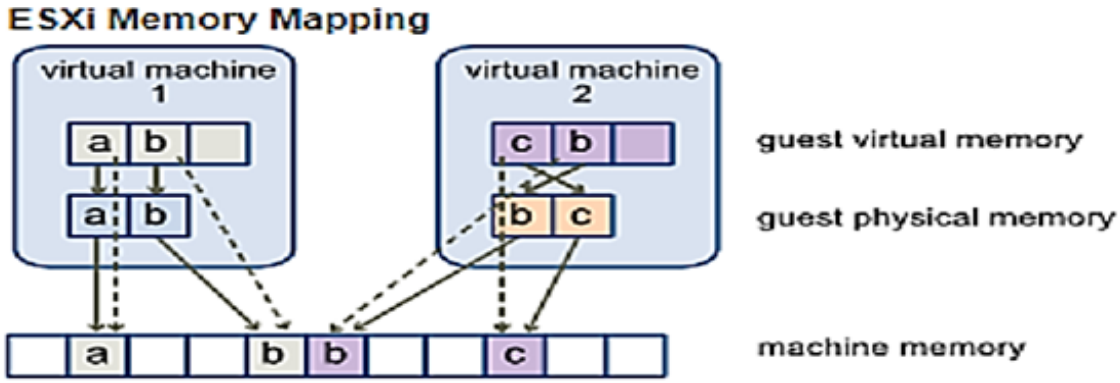
Şekil 10. Sanal Makine İşlemci Zamanlayıcısı (Hwanju, 2015)

Şekil 10'da görüldüğü üzere her VM (Sanal makine) de çalışmakta olan işletim sistemi kendi sanal işlemcileri biliyor ve işlemci isteklerini bunlara yapıyor. İşletim sistemindeki işlemci zamanlayıcısı, sanal makine içerisindeki işlemlerin isteklerini düzenliyor ve bu sanal işlemcilerden talep ediyor. Bu sanal işlemciler, fiziksel işlemci isteklerini sanallaştırma katmanındaki Sanal Makine Yöneticisi (VMM)'e iletiyor. VMM sanal makinelerden gelen bu istekleri kendi zamanlayıcısı ile düzenleyip fiziksel işlemci gücünü karşılıyor.

VMM bu istekleri kendi belirlediği ya da sanal makine özelliklerinde parametrik olarak verilen değerlere göre sıraya koyar. Sanal makine içerisinden gelen işlemci kaynağı talepleri bu akış içerisinde gerçekleştirildiği için performans kayıpları kaçınılmaz olmaktadır.

3.10.2. Bellek Performans Kayıpları

Fiziksel sistemin belleği dinamik olarak, sanal makineler ve konteynerler arasında paylaştırılmaktadır. Sanallaştırma ile birlikte işlemci üzerindeki bellek yönetim birimi (MMU) ve etkin sayfalar önbelleğini (TLB) de sanallaştırılmaktadır. Bu sayede sanal makineler ya da konteynerler fiziksel belleğe direk olarak erişememektedirler. Sanallaştırma yazılımları ile sanal ortamdaki bellek ile fiziksel bellek arasında adreslemeyi sağlamaktadır. Bu ara katman bellek kullanımı ve erişimi açısından performans kayıplarına neden olmaktadır. (VmWare Inc., 2008) Şekil 11'de Vmware ESXi ürünün bellek adresleme işlemi gösterilmiştir.



Şekil 11. Vmware Bellek Adresleme (Vmware vSphere 5.1 Documentation Center, 2015)

- Şekil 11'de kutucuklar bellek sayfalarını, oklar farklı adresleme yönlerini göstermektedir.
- Sanal makine 1 ve 2 içerisindeki konuk işletim sistemleri çalışmaktadır. Konuk sanal bellekten, konuk fiziksel belleğe olan adresleme okları, konuk işletim sistemi tarafından yapılmaktadır.
- Konuk fiziksel belleğinden, fiziksel makine belleğine doğru olan adresleme okları hipervizör katmanında Sanal Makine Yöneticisi (VMM) tarafından yapılmaktadır.
- Sanal makine işlemcisi Gölge sayfalar tablo adreslemesini kullanır. Bu adresleme şekilde çizgili oklar ile gösterilmiştir ve sanal makine monitörü tarafından yapılmaktadır.

Bellek adresleme işlemlerindeki bu ekstra katman bulunması, sanallaştırma ortamında bellek performanslarında ve erişiminde performans kayıplarına neden olmaktadır.

3.10.3. Depolama Alanı Performans Kayıpları

Sanallaştırma katmanın performans kaybı yaşadığı diğer bir kaynak ise dosya yazma okuma işlemlerinin yapıldığı disk kaynağıdır. Özellikle hipervizör tabanlı sanallaştırmada bir fiziksel kaynak üzerinde birden fazla ve farklı işletim sistemine sahip sanal makineler çalıştırıldığı için, bu makinelerin disk erişim istekleri sanallaştırma üzerinde daha fazla yük binmesine neden olmaktadır. Genel olarak bulut bilişimde ve sanallaştırma ortamlarında kullanılan asenkron G/Ç aktivitesi performans kayıplarına neden olmaktadır. (Li & diğerleri., 2013) . Daha iyi disk performansı elde etmek için araştırmalar yapılmaktadır. Bununla ilgili yayınlanmış en son çalışmalardan biri sanallaştırılmış bir networke bağlı olarak çalışan depolama aygıtının G/Ç performansının üzerine yapılmıştır. (Li & diğerleri., 2017)

3.10.4. Network Performans Kayıpları

Hipervizör tabanlı sanallaştırma çözümü olarak ele aldığımız Vmware ESXi çözümünde ağ sanallaştırama işlemi için birkaç sanal ve fiziksel kaynaklar kullanılmaktadır. Bunlar sanal ağ ara yüz kartı (vNIC), sanal dönüştürücü (vSwitch), sanallaştırma çekirdek sistemi (VMKernel) ve fiziksel ağ ara yüz kartıdır. Transfer edilecek olan paket ilk olarak vNIC tarafından vSwitch ‘e gönderilir. VSwitch ağ paketinin içeriğine fiziksel adresine ve hedefine bakarak ilgili G/Ç port kuyruğuna sokar. VMKernel içindeki ağ fonksiyonu kuyruktaki bu ağ paketini alarak fiziksel ağ ara yüz kartı sürücüsünü kuyruğuna sokar. Bu sürücü paketi alarak iletim kuyruğuna ekler. Bütün bu süreç içerisinde beklemeler ve gecikmeler olmaktadır. Sanallaştırma ortamında ağ performans kayıpları ile ilgili birçok araştırmalar yapılmıştır. (Shea & diğerleri., 2014), (Wang & Ng, 2010), (Felter & diğerleri., 2015), (Hwang, Zeng & Wu, 2013), (Oljira, Brunstrom & Taheri, 2016).

Yine aynı şekilde Konteyner tabanlı teknolojilerde ağ G/Ç aktivitesi için benzer sanal yönlendiriciler, sanal ağ ara yüz kartları kullanılmaktadır. Bu teknolojide de fiziksel ağ ara yüz kartlarına erişim için ara bir katman kullanıldığı için performans kayıpları yaşanmaktadır. Bu kayıplar için de çeşitli araştırmalar yapılmaktadır. (Zeng & diğerleri., 2017), (Claassen, Koning & Grosso, 2016), (Liu & Hao, 2011)

BÖLÜM 4. PERFORMANS KARŞILAŞTIRMA VE KULLANILAN ARAÇLAR

Hipervizör tabanlı sanallaştırma ile konteyner tabanlı sanallaştırma teknolojilerini karşılaştırma çalışması birçok farklı yönden ele alınabilir. Örnek olarak fiyat, ürünün destek seviyesi, internette ya da firmanın sağladığı dokümantasyon ve ürünün yaşam döngüsü boyunca alacağı güncellemeler ve hata düzeltmeleri verilebilir. Akademik ve sistem yöneticisi gözü ile bakıldığında akla ilk gelen, bu iki teknoloji üzerinde çalıştırılan sunucu, uygulama ya da servislerin sanallaştırma katmanın dezavantajları ile birlikte gelen performans kayıplarının ölçülmesi ve karşılaştırılmasıdır.

4.1. Performans Karşılaştırma Araçları

Endüstride sunucu kaynaklarının performanslarını karşılaştırmak ve ilgili kaynaklar üzerinde yük oluşturup skorlar oluşturmak ve için birçok araçlar bulunmaktadır. Bütün bir sistem üzerinde birden fazla araç ile sistemin genel toplam skorunu oluşturan araçlar olduğu gibi, tek tek kaynaklar üzerinde skor oluşturan araçlar da vardır. Bu araçlardan birkaçı genel olarak anlatılmıştır.

4.1.1. İşlemci Karşılaştırma Araçları

İşlemci performans karşılaştırma araçları, işlemciler üzerindeki kayıt edici üniteleri (register), aritmetik mantık ünite (ALU) ve matematik işlem ünitesi (FPU) gibi alanlar üzerinde yük oluşturarak çıkan sonuçları puanladıkları gibi, bazı sıkıştırma algoritmaları, matematiksel hesaplamaları ya da bunları içeren matematik problemlerini çözümünü sağlayarak, zaman, kullanım oranı değerlerine göre puanlama yaparlar. Bu araçlardan bazıları aşağıda anlatılmıştır.

PXZ sıkıştırma aracı:

Veri sıkıştırma işlemleri bilişim alanında sıklıkla kullanılmaktadır. PXZ aracı, paralel çalışan, LZMA (Lempel–Ziv–Markov Zincir Algoritması) algoritmasını kullanan veri sıkıştırma aracıdır. (Novy, 2009). LZMA Algoritması kayıpsız veri sıkıştırması yapmakta kullanılan, sözlük sıkıştırma algoritmasıdır.

PXZ çalıştırıldığı sistemin işlemci kaynaklarını otomatik algılayarak, kullanabildiği kadar işlemci iş parçacığını (thread) paralel olarak çalıştırır**.** Sonuçlar belirtilen zaman içinde sıkıştırılan dosya boyutu olarak hesaplanmaktadır. Birim olarak MB/s olarak ifade edilmektedir.

GeekBench

GeekBench (GeekBench, 2017), sistem üzerinde 27 farklı program çalıştırarak sistemin farklı alanlardaki skorlarını hesaplamaktadır. Bu alanlar Şifreleme skoru, Tam sayı işlemleri skoru, Kayan nokta işlemleri skoru ve bellek skoru olmak üzere 4 adettir. (GeekBench, 2018)

SysBench

Açık kaynak kodlu çoklu iş parçacığı desteği olan ve işlemci, bellek, disk G/Ç ve karşılıklı dışlama (MUTEX) performansını ölçmektedir. Çoğunlukla veri tabanı performans ölçümleri için kullanılmaktadır. (Gentoo Foundation, 2017) İçerisinde kurulum ile birlikte gelen dosya sistemi, işlemci, bellek, mutex kıyaslama araçları bulunmaktadır.

UnixBench

UnixBench ilk olarak 1983 yılında Unix sistemlerin çeşitli kaynaklarının ve özellikle işlemci performanslarını değerlendirmek üzere Monash Üniversitesinde açık kaynak kodlu olarak geliştirilmiştir. İlk çalıştırıldığında ilk olarak tek işlemci kullanarak sonuçlar üretmektedir. Sonrasında sistem üzerinde N adet işlemci varsa bunların birlikte kullanıldığı sonuçlar üretir. UnixBench aracının yaptığı testler Tablo 2'de verilmiştir. (UnixBench, 2018)

Tablo 2. UnixBench Test Açıklamaları (Google Code Arşiv, 2018)

Dhrystone	1984 Yılında Reinhold P. Weicker tarafından geliştirilmiş, işlemci üzerinde dizgi operasyonları yapmaya odaklanmış bir performans ölçme aracıdır. Kayan noktalı (floating-point) işlemler yapmaz. Sonuçları MIPS cinsinden vermektedir. MIPS (Saniyede yapılan milyon adet yönerge sayısı). Örnek olarak 1977 yılında kullanılan VAX 11/780 sisteminin skoru 1MIPS dir.
Whetstone	İşlemci üzerinde özellikle Kayan noktalı işlemler yaparak performans ölçümü yapmaktadır. Trigonometrik işlemler, dizi erişimleri ve prosedür çağrılarını test etmektedir. Sonuçlar MWIPS birimi olarak verilmektedir. Saniyede yapılan milyon adet whetstone yönerge sayısı.
Execl Çıktı	Saniyede yapılan execl çağrı sayısıdır. Execl, exec sistem çağrı fonksiyonun bir türüdür. Unix ve türevi sistemlerde çok daha sık kullanılır.
Dosya kopyalama	Farklı tampon ve blok büyüklükleri (1024/2000 blok, 256/500 blok, 4096/8000 blok) kullanılarak saniyede yapılan okuma, yazma ve kopyalama değerlerini büyüklük cinsinden raporlar. Birin olarak Kbps kullanır
Yöneltme Çıktısı (PIPE)	Yöneltme çıktısı performansı, saniyede yapılan 512 bayt verinin, Yöneltme sırasına yazılıp, buradan okunması sayısıdır.
Yöneltme Tabanlı İçerik Değiştirme	Yöneltme sırası kullanılarak bir dizginin iki işlem arasında değiştirilmesidir. Saniyede yapılan değiştirme işlemi olarak ölçülür.
İşlem Yaratma	İşletim sisteminin işlem yaratma çağrılarının sayısını ölçmektedir. Saniyede yaratılan ve bellekten okunabilen işlem sayısıdır.
Kabuk Betikleri	Bir veri dosyası üzerinde değişiklik yapan bir kabuk betiğinin çalıştırılması ve sonlandırılması işlemini ölçmektedir. 2 farklı ölçüm yapar, 1 eş zamanlı betik ve 8 eş zamanlı betik için ölçümler yapar.
Sistem Çağrısı Ölçümü	Çekirdek sisteme giriş ve çıkışın etkisini ölçmektedir. Örnek olarak, bir programın sürekli çekirdek sistemden bir işlemin tekil numarasını istemesi gibi. Saniyede yapılan çağrı sayısı olarak sonuç üretmektedir.

4.1.2. Bellek Karşılaştırma Araçları

Tek başına bellek performansını ölçmek mümkün değildir. Bellek performansı denildiği zaman ölçümler içerisine işlemci ve işlemci önbellekleri de girmektedir. Bellek karşılaştırma araçları sistem belleğini ve bellek okuma, yazma değiştirme işlem hızları ve saniyede yapılan işlemin büyüklükleri ve bellek altyapısı bant genişliği gibi farklı açılardan test etmektedir. Araştırma ve deneysel bilişim biliminde kullanılan araçların bir kısmı;

Bandwidth

Bandwidth aracı, bellek bant genişliği ölçmekte kullanılan ve sunucunun bellek alt sisteminin performansı, önbellek yapısı veri yolu hızlarının ve işlemci performansı ölçümlerinde kullanılmaktadır. Memst, memcpy ve bzero fonksiyonlarını kullanarak bellek üzerinde farklı İşlemci önbellek seviyelerinde sıralı okuma, yazma işlemleri yapmaktadır. (Smith, 2017)

RAMSpeed

RAMSpeed ya da diğer adı ile RAMSMP C programlama dilinde yazılmış, önbellek ve bellek performansını ölçen bir araçtır. Çoğunlukla kayan nokta işlemleri ve dizgi işlemleri yaparak sonuçlar üretmektedir. Büyük veri blokları kullanarak kayan nokta ya da dizgi verilerini bellek üzerindeki bir yerden başka bir yere kopyalama işlemleri yapmaktadır. Daha sonra bu veriler üzerinde değiştirme, ekleme gibi işlemler yaparak ölçümler sağlamaktadır. (Coffey & diğerleri., 2011)

STREAM

STREAM ilk olarak Fortran 77 dilinde daha sonra C dilinde yazılmış bellek test aracıdır. Bilgisayar işlemci hızları, bellek hızlarına oranla daha yüksek hızlara sahip olduğundan birçok uygulama ve programlar bellek performansına bağlı kalmaktadırlar. Bu araç sistemlerin özellikle önbellek boyutlarından daha büyük veri setleri ile çalışarak işlemci önbellek tekrar kullanımın önüne geçilmesi sağlanmıştır. (cs.virginia.edu, 2018) Bundan dolayı çok büyük ve vektör tipindeki uygulamaların performans ölçümünü yapmak için daha uygun görülmektedir. Aracın yaptığı vektör işlemleri aşağıdaki gibidir.

Name	Per iteration: kernel	bytes	FLOPS
COPY:	a(i) = b(i)	16	0
SCALE:	a(i) = q*b(i)	16	1
SUM:	a(i) = b(i) + c(i)	24	1
TRIAD:	a(i) = b(i) + q*c(i)	24	2

Bu işlemler uzun vektör işlemleri ile blokların yeniden oluşturulmasıdır. Dizi boyutları önbellek büyüklüğünden daha büyük olacak şekilde ayarlanarak, önbellek kullanımının önüne geçilmektedir. (McCalpin, 1996)

Mbw (Memory Bandwidth benchmark)

Mbw uygulaması halen geliştirilmekte olan ve işletim sistemleri yazılım kütüphanelerinde bulunup kolayca kurulumu yapılan bilen oldukça basit bir bellek hız ölçüm aracıdır.

Mbw aracı, farklı boyutlarda verilebilen büyük veri dizileri, belleğe kopyalama, yığın alma ve farklı blok boyutları kullanarak kopyalama işlemleri yapmaktadır. C programlama dilinin memcpy () ve mempcpy () konutlarını kullanmaktadır.

4.1.3. Sabit Disk G/Ç Karşılaştırma Araçları

Bu bölümde, endüstride performans ölçümlerinde sıklıkla kullanılan depolama alanı ve sabit disk performans ölçüm araçları anlatılacaktır. Bu araçlar sabit diskler üzerinde okuma, yazma işlem hızlarını ve dosya sistemi performansını ölçmektedir.

Bonnie++

Bu araç sabit disk ve dosya sistemi performansını ölçmek için bir dizi basit test senaryolarını çalıştırmaktadır. Veri okuma, yazma, silme işlem hızlarını ölçmekte aynı zamanda saniyede yapılabilen arama adetini hesaplamaktadır. Rusel Coker tarafından yapılan geliştirmeler ile birlikte sabit diskin fiziksel sektörlerine direk erişim testi yapabilmekte ve 2G dan daha büyük veri setleri ile çalışabilmektedir. (Coker, 2018)

Iozone

Iozone çeşitli dosya işlemlerini ölçerek ve çeşitli dosya işlemleri oluşturarak sonuçlar üretmektedir. Okuma, yazma, tekrar okuma, üzerine yazma, terse okuma, atlayarak okuma, rastgele okuma yazma gibi birçok işlem yaparak dosya G/Ç performansını ölçmektedir. (IOZone.org, 2018)

Flexible IO (FIO)

Jens Axboe tarafında yazılmış bu araç birden fazla iş parçacığı kullanarak kullanıcı tarafından belirlenen G/Ç işlemleri yapar. Tipik kullanımı, kullanıcı tarafından görülmek istenen G/Ç yükünü gerçekleştirecek iş dosyası hazırlanıp çalıştırılması şeklinde olmaktadır. (Axboe, 2018)

4.1.4. Network G/Ç Karşılaştırma Araçları

Network performans değerlendirmesi, network bant genişliği kullanımı, network gecikme süresi, paket kayıpları gibi network üzerinde gerçekleşen işlemlerin değerlerine göre sonuçlar üretmektedir.

Iperf3

Iperf3 aracı IP networkünde ulaşılabilecek en yüksek bant genişliğine ulaşmaya çalışan ve bu açıdan ölçümler yapan ve az sistem kaynağı kullanan bir araçtır. IP Network ayarlarında tampon büyüklüğü, çeşitli network protokolleri gibi parametrelerin seçilmesine olanak sağlayarak çeşitli senaryolarda testlerin çalıştırılmasına izin verir.

Sunucu istemci mimarisinde çalışan bu araç test süresince transfer edilen veri miktarı, test süresince ulaşılan bant genişliği cwd değerlerini hesaplamaktadır. Test edilecek sistem sunucu, ilgili yükün oluşturulacağı taraf ise istemci olarak ayarlanır. (IPerf, 2018)

Nuttcp

Bu araç network çıktı değerini, network kartı ve network bant genişliğini ölçmektedir. Sunucu istemci mimarisinde çalışmaktadır. Burada test edilecek olan sistem istemci, dışarıdaki sistem ise sunucu rolündedir. Testlerde çıktı değeri değişken olarak verilmekte ve 650-1100 Mbps değerleri arasında testler yapabilmekte ve network giriş ve çıkış değerlerini ölçebilmektedir. (Nuttcp , 2016)

Netperf

Netperf aracı network seviyesinde birçok farklı testler yapabilmesine rağmen daha çok network gecikmelerini ölçmek için kullanılmaktadır. İstek ve cevap süresi çevrimi ölçümlemektedir. Sunucu istemci mimarisinde çalışan bu araçta test edilecek olan sistem sunucu, dışarıdaki sistem ise istemci rolündedir.

Netperf ’in en sık kullanılan özelliklerinden bir tanesi toplu veri transferidir. Akış, ya da tekyönlü akış performansı olarak da bilinen bu test senaryosunda bir sistemin bir veriyi ne kadar hızlı gönderebileceği ve diğer bir sistem bir veriyi ne kadar hızlı alabileceği ölçülmektedir.

Qperf

Qperf aracı istemci sunucu mimarisinde, TCP/IP üzerinden çalışan network bant genişliği, network gecikmeleri ve işlemci kullanım oranlarını hesaplayan network performans ölçme aracıdır. (George, 2018)

BÖLÜM 5. YÖNTEM

Bu çalışmada bulut ortamında ve kurumsal veri merkezlerinin birçoğunda sıklık ile kullanılan iki teknolojinin performans değerlerinin karşılaştırılması ve uygun bir teknoloji seçimi yapılabilmesi için analizler üretilmesi amaçlanmıştır. Sonuç olarak yapılacak olan çalışmada bir takım kıyaslama araçları ile sentetik yükler üretilip sistemlerin üreteceği değerlerin yorumlanmasıdır. Bu nedenle bu çalışmayı Deneysel Bilgisayar Bilimi (ECS) alanında bir çalışma olarak değerlendirebiliriz. (Feitelson, 2006)

5.1. Araştırma Modeli

Test ortamların hazırlanması ve uygulanması boyunca metodoloji olarak Etki Alanı Bilgisi Bilgi Teknolojisi Metodolojisi (DoKnowMe) (Li, O'Brien & Kihl, 2016) takip edilmiştir. Bu metodoloji kullanılarak çalışmayı, deney öncesi hazırlık aktiviteleri ve deney sonuçları ve analizleri olarak iki alana ayırdık. Buna göre,

5.1.1. Deney Öncesi Hazırlık Aktiviteleri

Deney öncesi hazırlık aşamasında, bu araştırmanın bulmaya çalıştığı sorular, ölçümü yapılacak olan hizmetlerin tanımı, kullanılabilecek olan kıyaslama araçları ve bunların sonuçlarının kullandığı ölçü birimleri, deneyleri etkileyecek kaynak tipi, kaynak büyüklüğü gibi faktörlerin listelenmesi ve seçimi ve deneyin akış şemasının çıkartılması gibi deneyin tutarlı ve tekrarlanabilir olmasını sağlayan ve deneyler boyunca izlenmesi gereken yol haritası belirlenmiştir.

Gereklilik Tanımlama

Deneyler boyunca izleyeceğimiz metodolojiye göre bu deneyin neden yapılacağını, neden gereklilik duyulduğunun tanımlanması gerekir. Bunun net ifadeler ile yazılması beklenir. Gereklilik tanımlama bizim çalışmamız açısından, endüstride sıklıkla kullanılan hipervizör ve Konteyner tabanlı sanallaştırma teknolojilerinin seçimi ve tercihi konusunda performansa dayalı analizler elde etmektir. Bu doğrultuda bu iki teknolojinin niceliksel olarak karşılaştırılması gerekmektedir.

Bu teknolojilerin çıktıları olan Vmware sanal makinesi ile Docker Konteynerleri aynı kaynaklar üzerinde ayrı yarı hazırlanıp aşağıdaki soruları cevap bulunması beklenmektedir.

Soru 1: Tek başına çalışan bir Docker Konteyneri, temel alınan fiziksel sunucuya göre ne kadar performans kaybı yaşıyor.

Soru 2: Tek başına çalışan Vmware sanal makinesi, temel alınan fiziksel sunucuya göre ne kadar performans kaybı yaşıyor.

Soru 3: Tek başına çalışan bir Docker Konteyneri, temel alınan fiziksel sunucuya göre belirli bir zaman aralığı boyunca performansı nasıl değişiyor.

Soru 4: Tek başına çalışan bir Vmware sanal makinesi, temel alınan fiziksel sunucuya göre belirli bir zaman aralığı boyunca performansı nasıl değişiyor.

Hizmet Özellikleri Tanımlama

Bulut ortamında ve kurumsal veri merkezlerinde verilen servislerin kalitesi sınıflandırıldığında network, hesaplama, bellek ve depolama değerleri üzerinden değerlendirilmektedir. (Li & diğerleri., 2012) Bu nedenle araştırma 2 teknolojiyi karşılaştırmak için hesaplama hızı, bellek erişim ve bant genişliği hızı, depolama alanı hızı ve network bant genişliği ve gecikmesi değerleri ölçülüp analiz edilmiştir.

Deney Faktör Listesi Ve Seçimler

Deney öncesinde, deney faktörlerinin belirlenmesi ve tanımlanması deney sonuçlarının tekrarlanabilmesi ve deney sonuçlarının güvenilirliği için çok önemli yer tutmaktadır. Bu nedenle bu faktörleri kaynak tipi ile ve yük tipi ile alakalı faktörler olarak sınıflandırılmıştır.

Kaynak Tipi İle İlişkili Faktörler

Aşağıda belirtilen İşlemci, Disk, Bellek ve Network kaynakları ve özellikleri testleri yapılacak olan sistemler için tamamen aynıdır. Deneylerin ve analizlerin günümüzde geçerliliğinin olması için, mümkün olduğunca yeni sürümler, yeni donanımlar ve yeni işletim sistemi sürümleri tercih edilmiştir.

- Kaynak Tipleri: Deney ortamında aynı fiziksel özelliklere sahip 3 farklı kurulum içeren sunucular kullanılmıştır. Fiziksel Sunucu olarak ProLiant BL460c Gen8 kullanılmıştır. (HP ProLiant BL460c Gen8 Server Blade)
- Network : Deney ortamında kullanılan sunucular üzerinde 10 GB hızında “HP FlexFabric 10Gb 2-Port 534FLB” adaptörü kullanılmıştır. Network trafiğinin geçtiği

Network ağı en fazla 1 GB hızı desteklediği için, görülebilecek en yüksek hız 1 GB olacaktır.

- İşlemci: Deney ortamında kullanılan sunucular üzerinde 2 adet 10 çekirdekli Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz işlemcisi bulunmaktadır. Testler boyunca yanıltıcı olmaması için Intel Hyperthreading özelliği kapatılmıştır.
- Bellek : Deney ortamında kullanılan sunucular üzerinde 1866 MHz Hızında 16Gb kapasitesi olan 16 adet DIMM DDR3 Bellek bulunmaktadır. Deney süresince sunucu bellekleri 32 GB olarak kullanılmıştır.
- Depolama Alanı : Deney ortamında kullanılan sunucular üzerinde 256Gb boyutunda 2 adet 15K Rpm hızında SAS sabit diskler bulunmaktadır. Bu diskler Smart Array P220i Controller ile RAID 0 olarak kullanılmıştır.
- İşletim Sistemi : Deneyler boyunca test edilen sistemlerde Ubuntu 18.04 Bionic Beaver x64 sürümü kullanılmıştır. Docker konteynerleri için de aynı şekilde Ubuntu 18.04 Bionic Beaver sürümü temel imaj olarak alınmıştır.
- Sanallaştırma Yazılımı: Deney ortamında hipervizör tabanlı sanallaştırma için Vmware vSphere 6.5 (VMware vSphere 6.5 Documentation, 2017) Docker yazılımı olarak 17.12.1-ce, build 7390fc6 (Docker CE, 2018) kullanılmıştır.

Yük Tipi İle İlişkili Faktörler

- Süre: Deneyler sırasında kullanılacak olan yük araçları önceki bölümlerde belirtilmiştir. Süre değerini değişken olarak alabilen araçlar için yük testleri 60 saniye, 600 Saniye, 1800 Saniye süreleri boyunca çalıştırılıp, zamana bağlı değişiklikler gözlemlenecektir.
- Yük Büyüklüğü: Kullanılacak olan araçlar tarafından otomatik olarak tanımlanmaktadır.

Kıyaslama Araçlarının Seçimi ve Birimler

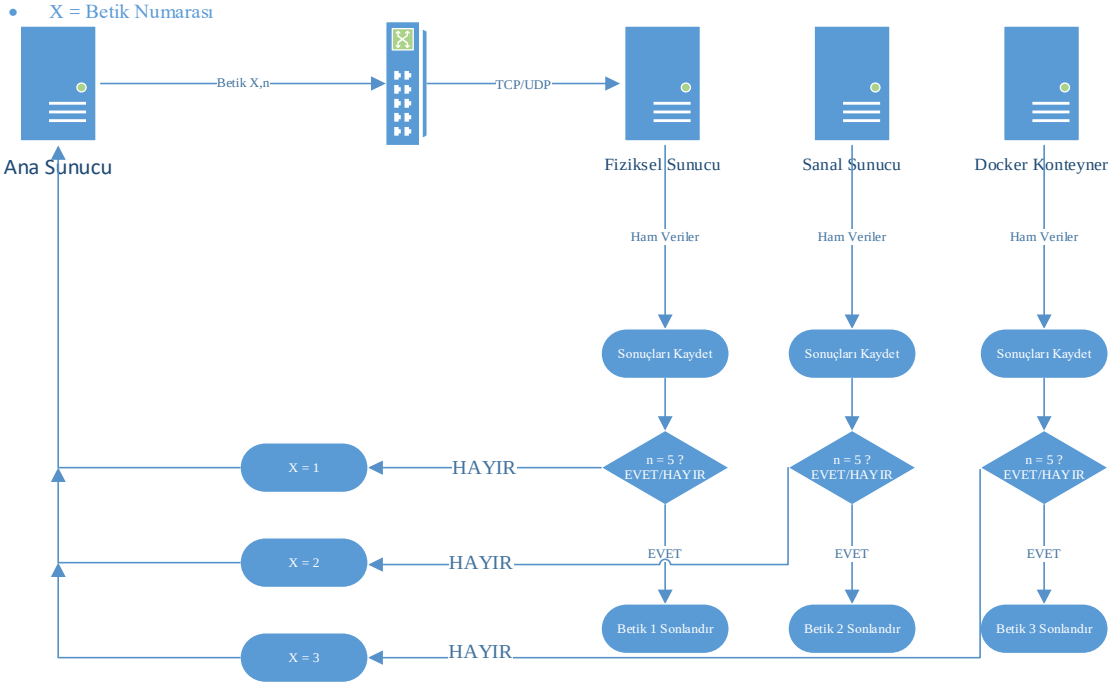
Kıyaslama araçlarının seçimi daha çok o araçların deney ortamı için uygunluklarına dayanır. Ayrıca bu araçlar seçilirken çalıştıkları sisteme, kendilerinin çalışmasından dolayı en az yükü getirmesi beklenir. Tablo 3'te kullanılan araçlar ve servis özelliklerinin listesi bulunmaktadır.

Tablo 3. Performans Test Araçları

Kaynak	Ölçülen Değer	Birim	Araç	Versiyon
Network	Network bant genişliği	Mbps	Qperf	0.4.10
Network	Network Gecikmesi	µs	Qperf	0.4.10
Hesaplama	Hesaplama	Saniye	PXZ	4.999.9beta (build 20170929)
Hesaplama	Hesaplama	Farklı birimlerde çıktı üretiyor	GeekBench	4.2.3
Bellek	Bellek Veri Çıktı Değeri	MB/s	STREAM	
Depolama	Okuma/Yazma Hızı	MB/s	Bonnie++	

Deney Planı ve Akış Şeması

- Betik 1 (Fiziksel Sunucu)
- Betik 2 (Sanal Sunucu)
- Betik 3 (Docker Konteyneri)
- N = Çalıştırma Sayısı
- X = Betik Numarası



Şekil 12. Deney Akış Şeması

5.1.2. Deney Sonrası Veriler Ve Analizler

Etki Alanı Bilgisi Bilgi Teknolojisi Metodolojisi (DoKnowMe) (Li, O'Brien & Kihl, 2016) takip edildiğinde, deney sonrası aktivitelerin içine ham verinin elde edilmesi, bunların iyileştirilmesi ve analiz edilmesi girmektedir. Bu bilgiler çalışmanın bir sonraki bölümü olan Bulgular ve Yorumlar bölümünde ele alınacaktır.

Bu bölümde tüm etki alanı bilgileri bizim çalışmamızda test edilen kaynakların ölçümleri sırasında elde edilen ham veriler ortaya konmuş, iyileştirilmiş ve analiz edilmiştir.

Veriler ve Toplanması

Bu çalışma boyunca kullanılan Tablo 1 de kullanılan araçlar ile her bir kaynak için, her üç ortama aynı büyüklüklerde ve aynı parametreler kullanılarak performans testleri uygulanmıştır. Verilerin tutarlılığı ve tekrarlanabilirliğini sağlamak için her test araçlara parametrik olarak 5 defa tekrarlatılmış ve çıktı değerleri bu beş test koşumunun ortalaması alınarak hesaplanmıştır. Bu veriler Excel ortamına üç farklı ortamın, aynı kaynakları üzerinde elde edilen değerleri olarak kaydedilmiştir. Testler boyunca elde edilen ham veriler, oluşturulan tablolar, kullanılan betik komutları EK'ler bölümüne konulmuştur.

Verilerin Çözümlenmesi ve Yorumlanması

Toplanan ham veriler, Excel üzerinde tablolar halinde her bir kaynak için ayrı ayrı tutulmuştur. Performans kayıp oranları hesaplanırken, fiziksel kaynak üzerinde elde edilen değerler temel alınmıştır. Bu temel değeler üzerinden yüzdesel olarak konteyner ve hipervizör tabanlı sanallaştırma ortamlarının performans kayıpları hesaplanmıştır. Hesaplamalar aşağıdaki formül kullanılarak yapılmıştır.

$$P_K = \frac{|B_m - B_F|}{B_F} X \frac{1}{100}$$

- P_K = Performans Kaybı (%)
- B_M = Test edilen sistemin Ölçülen Değeri
- B_F = Fiziksel Sistemin Ölçülen Değeri

Denklem 1. Performans Fark Denklemi

BÖLÜM 6. BULGULAR VE YORUMLAR

Bu bölümde her bir kaynak için yapılan ölçüm sonuçlarına yer verilmiştir. Kullanılan karşılaştırma araçları ve sonuç birimleri Tablo 5.1 verilmiştir. Performans karşılaştırma sonuçlarında, yapılan ölçüm birimine göre bazen "Yüksek olan daha iyi", bazen "Düşük olan daha iyi" değerlendirilen durumlar olmaktadır. Bu durumlar sonuç tablo veya grafiklerinde belirtilmiştir.

6.1. İşlemci Performansı Testleri

İşlemci performansı değerlendirilirken 2 farklı araç kullanılmıştır. Bunlardan ilki veri sıkıştırma aracı ve diğeri işlemci üzerinde şifreleme, kayan nokta hesaplamaları ve tam sayı işlemleri yapan işlemci GeekBench aracıdır. Performans kayıpları hesaplaması için 1. Denklem kullanılmıştır. Teknolojilerin fiziksel sunuculara göre performans kayıpları karşılaştırılacağından fiziksel sunucunun sonuçları temel alınmıştır.

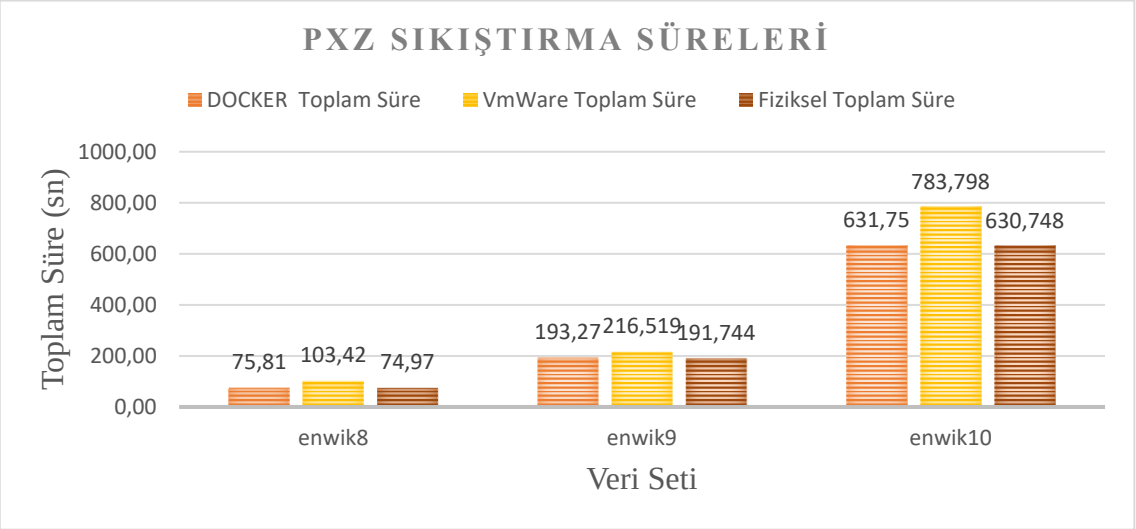
6.1.1. Veri Sıkıştırma Ölçümleri

İşlemci Performansını ölçmek için ilk olarak veri sıkıştırma aracı kullanılmıştır. Bu araç kullanılarak boyutları farklı olan, aynı veri tipine sahip üç adet test veri seti ile çalışılmıştır. Test veri seti olarak sıkıştırma algoritmaları testlerinde sıklık ile kullanılan Wikipedia sitesinin, boyutları 10^8 bayt, 10^9 bayt ve 10^{10} bayt olan yığın dosyaları kullanılmıştır. (Mahoney, 2018) Sıkıştırma aracı olarak PXZ 4.999.9beta (build 20170929) versiyonu kullanılmıştır.

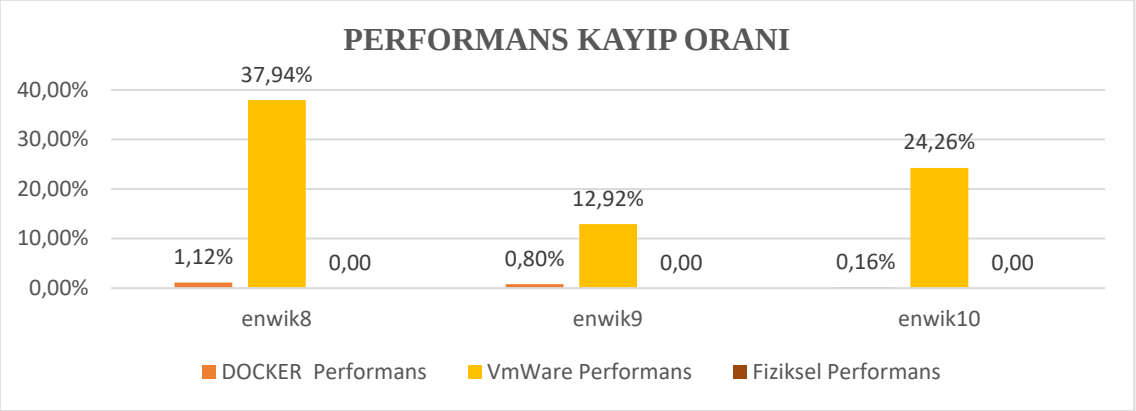
Disk ve bellek işlemlerinin, işlemci performansına olan etkileri en aza indirmek için, test verilerini sıkıştırma işleminden önce "vmtouch (Hoyte, 2017)" programı ile belleğe yüklenmesi sağlanmıştır. Bu sayede verilerin direk bellekten okunması sağlanmıştır. Sıkıştırma işlemi, en yüksek oranda sıkıştırma ve en yüksek sayıda işlemci iş parçacığı kullanacak şekilde çalıştırılmıştır.

Sıkıştırma işlemi, aynı veri setleri için, her teknoloji üzerinde çalıştırılmış ayrı ayrı çalıştırılmıştır. Her çalıştırmadan sonra işlemci ve sistem kaynak kullanımın normal değerlere dönmesi için beş dakika ara verilmiştir. Verilerin ham hali EK'lerde verilmiştir.

Farklı büyüklüklerdeki veri setleri ve karşılaştırdığımız teknolojilerinin sıkıştırma süreleri ve performans kayıpları Şekil 13 ve Şekil 14'de gösterilmiştir. Bu grafiklerde "Daha düşük değer daha iyi" ifade etmektedir.



Şekil 13. Veri Sıkıştırma Süreleri



Şekil 14. Sıkıştırma Performans Kayıp Oranları

6.1.2. Veri Sıkıştırma Sonuçlar ve Analiz

Veri sıkıştırma süreleri farklı veri setlerinde birbirlerine paralel fakat farklı sonuçlar göstermiştir. İlk veri seti ele alındığında fiziksel makine performansına en yakın Docker görülmektedir. Vmware sanal makinesi ise bu iki sistem daha kötü performans göstermiştir. İkinci ve üçüncü veri setlerinde de aynı durum gözlenmektedir.

Veri sıkıştırma işlemindeki performans kayıpları ise Şekil 14'de verilmiştir. Elde edilen sonuçlara göre 10^8 bayt boyutundaki ilk veri setinde Vmware'in fiziksel sisteme göre %37,94 oranında performans kaybı yaşadığı görülmektedir. Aynı veri setinde Docker %1,12 oranında performans kaybı yaşamıştır.

Veri boyutu 10^9 olduğunda performans kayıp oranlarının her iki teknoloji için azaldığı görülmektedir. İkinci veri seti için Vmware %12,92, Docker ise %0,80 performans kaybı yaşamıştır.

Veri boyutu 10^{10} olduğunda iki teknolojinin de ilk veri setine oranla daha az performans kaybı yaşadığı görülmektedir. Özellikle Docker ile fiziksel makine arasında performans farkı çok daha aza inmiştir. Üçüncü veri seti için Vmware %24,26, Docker %0.16 performans kaybı yaşamıştır.

Veri sıkıştırma işlemlerinde, veri boyutu büyüdükçe performans kayıplarının azaldığı görülmüştür. Özellikle Docker konteyneri fiziksel sisteme oldukça yakın sonuçlar üretmeyi başarmıştır.

6.1.3. Kayan Nokta, Şifreleme ve Tam Sayı İşlemleri

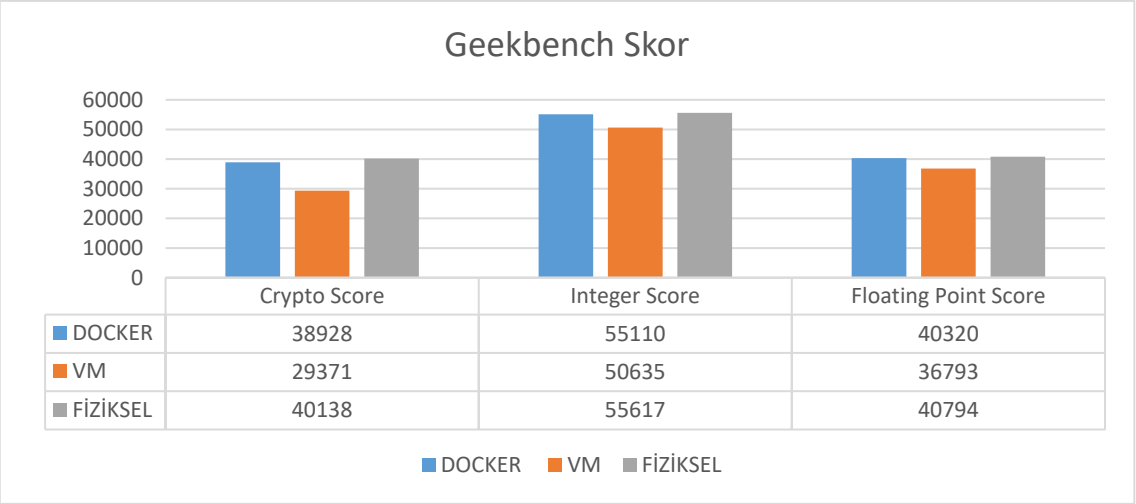
Bu ölçümler için GeekBench performans aracı kullanılmıştır. Bu araç daha önceki bölümlerde açıklandığı üzere sistem üzerinde çeşitli matematiksel işlerler yaparak özellikle işlemci ve bellek üzerinde yük oluşturmaktadır. Yapılan testlerde bu aracı işlemci skorları alınmak üzere kullanılmıştır.

Bu araç 3 ana bölüm üzerinden skorlar çıkarmaktadır. Şifreleme, Tam sayı ve Kayan Nokta işlemleri bölümleri için skor elde etmek için çeşitli iş yükleri oluşturmaktadır. Örnek olarak Şifreleme skoru belirlemek için endüstride çok kullanılan blok şifreleme standardı AES kullanarak 256-bit Anahtar ile 32 Mb veri şifrelemektedir.

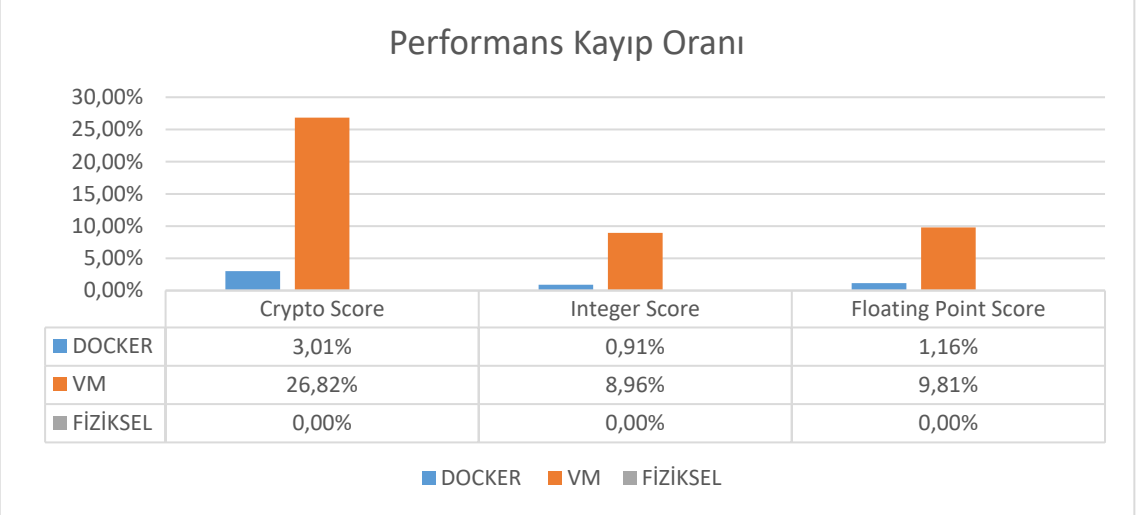
Kayan nokta hesaplamaları için Genel matris çarpımı, konuşma algılama, N-Body fiziği gibi farklı işlemler yapmaktadır.

6.1.4. Kayan Nokta, Şifreleme ve Tam Sayı İşlemleri Sonuçlar ve Analiz

GeekBench 4.2.3 versiyonu ile yapılan testlerin skor ve performans kayıp grafikleri Şekil 15 ve Şekil 16'da gösterilmiştir. İşlemci skor grafiklerinde "Yüksek olan daha iyidir" durumu geçerlidir.



Şekil 15. GeekBench Skor



Şekil 16. Performans Kayıp Oranı

Şekil 15 ve Şekil 16'da görüldüğü üzere Docker konteyneri fiziksel makine skorlarına en yakın skorları üretmektedir. Vmware sanal makine skoru belirgin bir fark ile geride kalmaktadır.

Performans kayıpları açısından bakıldığında, sıkıştırma algoritma testlerine yakın veriler elde edilmiştir. Docker konteyneri Tam sayı işlemleri ve Kayan Nokta işlemlerinde %0,91 ve %1.16 performans kayıpları ile Fiziksel makine ile birbirine çok yakın değerler ürettiği görülmektedir. Aynı işlemler için Vmware sanal makinesi sırası ile %8,96 ve %9,81'lik puan kayıpları göstermiştir.

En yüksek performans kayıplarının şifreleme işlemlerinde yaşandığı görülmektedir. Docker konteyneri %3.01, Vmware Sanal Makinesi %26,82'lik bir performans kaybı yaşadığı görülmüştür.

6.2. Bellek Performans Testleri

Bellek performansı değerlendirilirken, bellek performans ölçümleri için endüstride ve yüksek başarımlı bilgisayarlar performans ölçümleri alanında kabul görmüş olan stream aracı kullanılmıştır. Stream aracı Virginia Üniversitesi çatısı altında halen geliştirilmeye devam etmektedir ve Fortran 77 dilinde yazılmıştır.

6.2.1. Stream Bellek Ölçümleri

Bu araç oldukça basit olup, dört adet uzun vektör işlemi yapmaktadır. Bu şekilde bellek bant genişliklerini ölçmekte ve MB/sn. biriminden sonuçlar üretmektedir. Dizi büyüklükleri işlemci önbellek etkisini dışarıda bırakmak için işlemci önbelleğinden daha büyük olarak seçilmektedir.

Tablo 4. Stream Vektör işlemleri. (McCalpin, 1996)

Stream Operations		Per Iteration	
Name	Kernel	bytes	FLOPS
COPY	a(i) = b(i)	16	0
SCALE	a(i) = q*b(i)	16	1
SUM	a(i) = b(i) + c(i)	24	1
TRIAD	a(i) = b(i) + q*c(i)	24	2

Beş farklı boyutlardaki veri dizileri kullanılmıştır. Stream aşağıdaki dört işlem ile bellek performansını ölçmektedir. (McCalpin, 1996)

COPY = Aritmetik işlem olmadan bellek üzerinde veri transfer oranlarını ölçer.

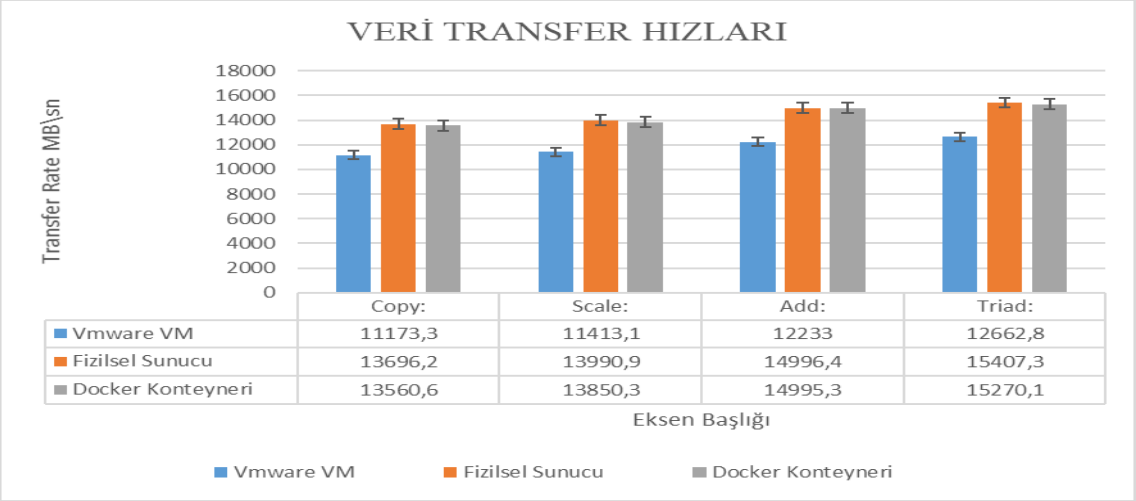
SCALE = Basit aritmetik işlemler ekleyerek bellek üzerinde veri transfer oranlarını ölçer.

SUM = c(i) gibi üçüncü bir işlem ekleyerek birden fazla yük ekleyerek bellek üzerinde veri transfer oranlarını ölçer.

TRIAD = Zincirleme, birbiri üzerine örtüşen, kaynaştırılmış işlemlere izin vererek bellek üzerinde veri transfer oranlarını ölçer.

Performans kayıpları hesaplaması için 1. Denklem kullanılmıştır. Teknolojilerin fiziksel sunuculara göre performans kayıpları karşılaştırılacağından fiziksel sunucunun sonuçları temel alınmıştır.

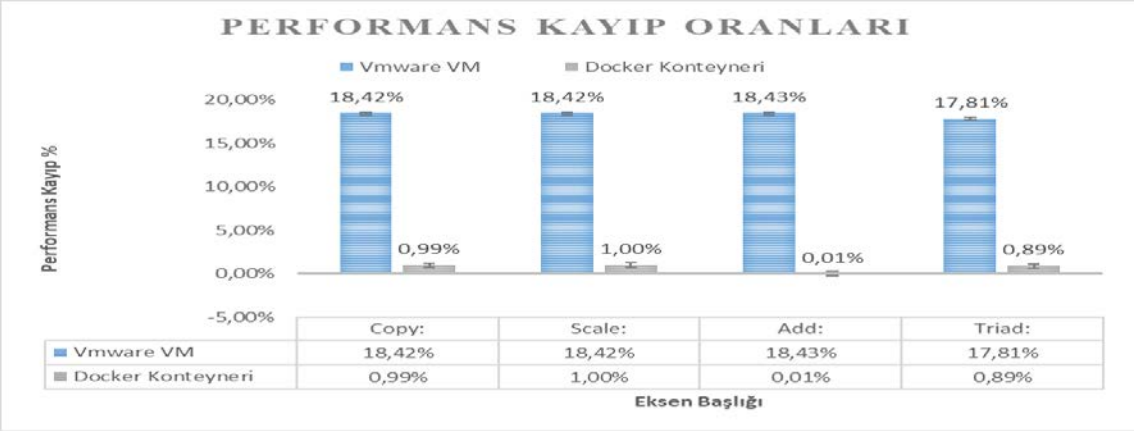
6.2.2. Stream Bellek Ölçüm Sonuçları



Şekil 17. Stream Veri Transfer Hızı Sonuçları

Stream aracı ile yapılan testlerde grafiklerde görüldüğü gibi Docker konteyneri ve fiziksel sunucu birbirlerine çok yakın değerler elde etmiştir. Vmware sanal makinesi veri transfer hızı diğer iki sisteme göre görünür bir şekilde daha kötü performans göstermiştir.

Stream aracının çalıştırdığı dört fonksiyonda birbirine yakın değerler elde edilmiştir. Şekil 18'de test edilen sistemlerin fiziksel sunucu temel alınarak yapılan performans kayıpları görülmektedir.



Şekil 18. Bellek Performans Kayıpları

Şekil 18'de görüldüğü üzere Vmware sanallaştırma katmanı yapılan dört uzun vektör işlemlerinde veri transfer hızı açısından %17 ile %18 arasında performans kaybına neden olmaktadır. Öte yandan Docker konteyneri veri transfer hızı açısından fiziksel makineye göre %0,89 ile %1 gibi düşük oranlarda performans kaybı yaşamıştır. Docker sanallaştırma katmanının, Vmware hipervizör göre çok daha iyi bir performans gösterdiği görülmüştür.

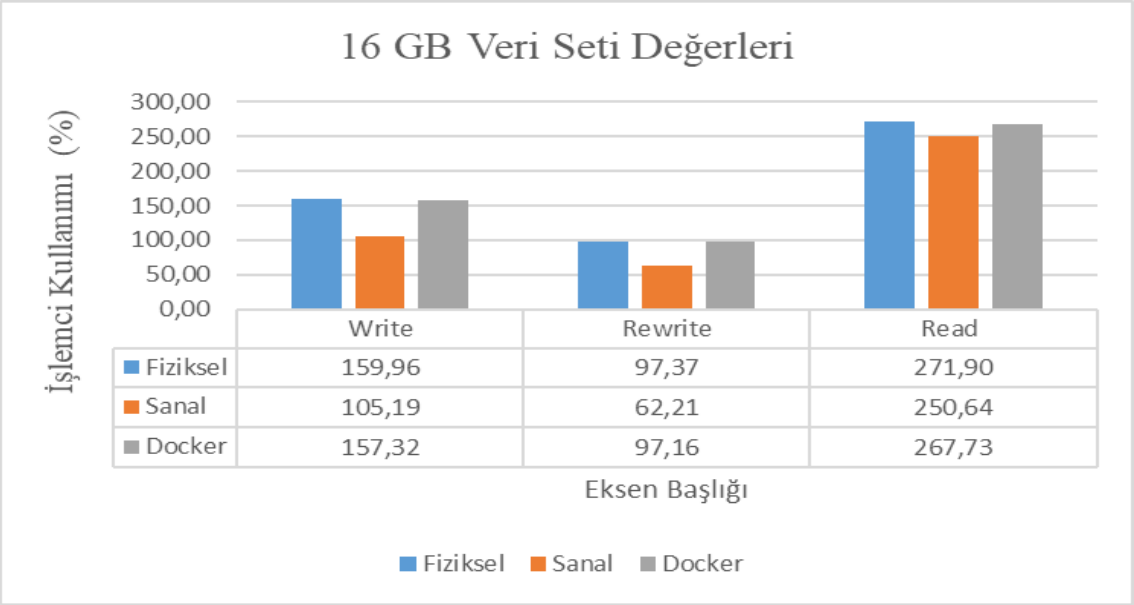
6.3. Sabit Disk G/Ç Performans Testleri

Sabit Disk G/Ç performansı ölçümleri için Bonnie++ aracı kullanılmıştır. Bu araç ile farklı büyüklüklerdeki veri setleri kullanılarak, her bir sistemin ardışık okuma, yazma ve tekrar yazma hızlarını ölçülmüştür. Belirtilen büyüklükteki veri setini, karakter karakter ya da blok blok yazma testleri yaparak her biri için sonuçlar üretmektedir.

Performans testleri sırasında, bellek ve önbellek etkisini azaltmak için Bonnie++ sistem belleğinin iki katı büyüklüğündeki veri seti kullanmaktadır. Sabit disk testleri için sunucuların bellek boyutları 16 GB verilmiştir. Farklı büyüklüklerde veri setleri ile çalışmak için Bonnie++ aracı çalıştırılırken "-r" parametresi ile sistem bellek boyutu verilip, uygulamanın sistem belleğini olduğundan daha küçük ya da daha büyük algılaması sağlanmıştır. Uygulama sistem belleğinin iki katı veri ile çalıştığından her farklı -r parametresi için farklı veri setleri elde edilmiştir.

6.3.1. Bonnie++ Sabit Disk G/Ç Ölçümleri

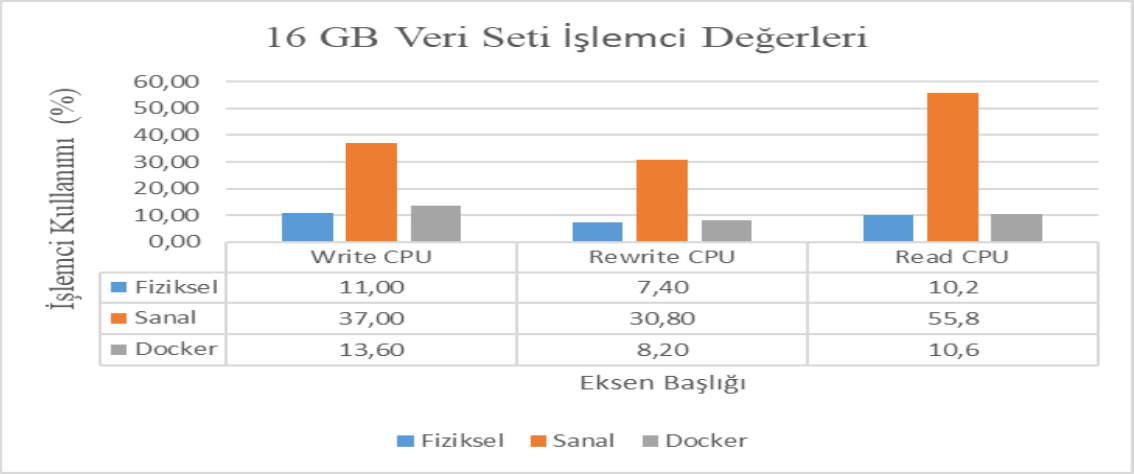
Bonnie++ ardışık blok okuması yaparken C programlama dilinin read () fonksiyonunu, karakter okuması yaparken putc () fonksiyonunu kullanmaktadır. Aşağıdaki grafikler farklı boyutlardaki veri blok okuma hızlarını göstermektedir. Bu grafik saniyede işlenen veri boyutunu göstermektedir. Bu grafiklerde "Daha yüksek değer daha iyi" ifade etmektedir.



Şekil 19. 16 GB Veri Okuma Yazma Değerleri

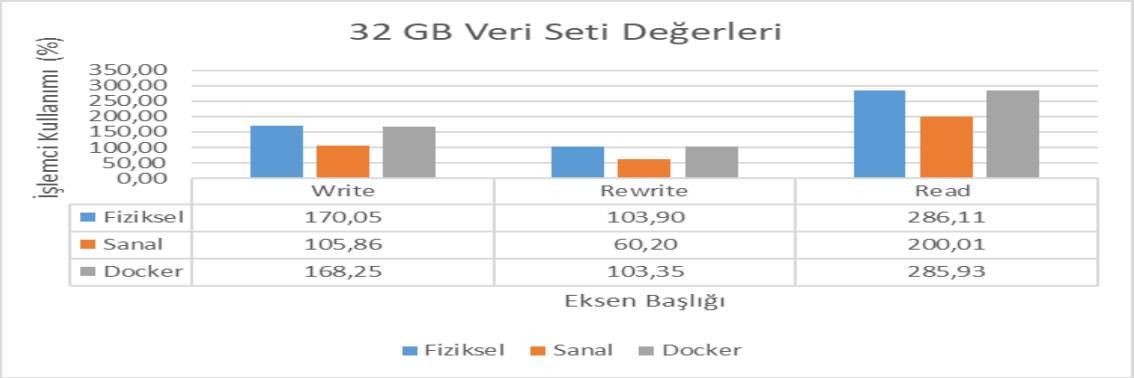
3 sistemin fiziksel bellek ile aynı boyutta olan, 16 GB veri seti ile yapılan okuma, yazma testlerinde en iyi performansı saniyede 159.96, 97, 271 MB ile fiziksel sunucu göstermiştir. Fiziksel sunucudan sonra Docker Konteyneri gelmekte ve en kötü performansı Vmware Sanal makinesi göstermektedir. Vmware Sanal makinesi ile Docker Konteynerinin arasında en fazla farkın yazma işlemi yaparken gerçekleştiği görülmektedir.

Bu işlemler yapılırken yapılan işlemlerin işlemci kullanım oranları aşağıdaki grafiklerde gösterilmiştir. Bu grafiklerde "Daha düşük değer daha iyi" ifade etmektedir.



Şekil 20. 16 GB Veri Seti İşlemci kullanım Oranları

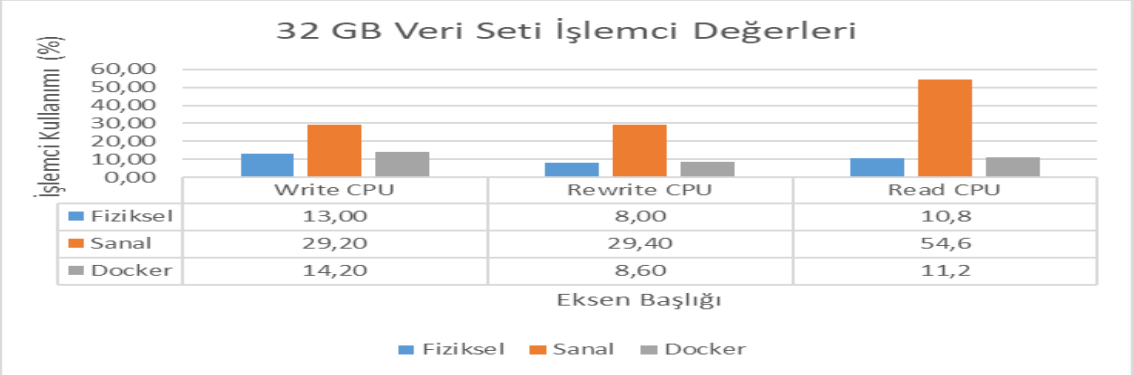
Okuma, yazma işlemi yaparken, işlemci kullanım oranlarına bakıldığında en fazla işlemci kullanımını Vmware sanal makinesi yapmaktadır. Fiziksel makine ile Docker konteyneri birbirine yakın oranlarda işlemci kullanmışlardır. İşlemci kullanımındaki artış, genel olarak sistemde yavaşlığa neden olduğu gözlemlenmiştir. Bir sonraki testler 32 GB büyüklüğündeki veri seti ile yapılmıştır. Bu grafiklerde "Daha yüksek değer daha iyi" ifade etmektedir.



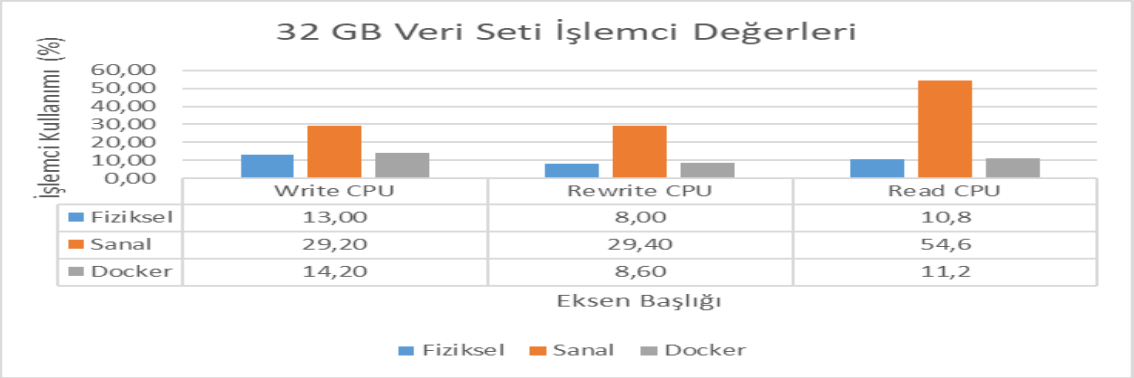
Şekil 21. 32 GB Veri Okuma Yazma Değerleri

Test edilen sistemler, fiziksel belleğin iki katı büyüklüğündeki 32 GB boyutundaki veri seti ile test edilmiştir. Yapılan üç işlem için en iyi performansı fiziksel sunucu daha sonra Docker Konteynerinin göstermiştir. 16 GB'lık veri setinde olduğu gibi, Vmware Sanal makinesi ile Docker konteyneri arasındaki en büyük fark yine yazma işlemi sırasında gerçekleştiği görülmektedir.

Aşağıdaki grafik 32 GB veri seti ile çalışılırken, okuma, yazma ve tekrar yazma işlemleri sırasında işlemci kullanım oranını göstermektedir. Bu grafiklerde "Daha düşük değer daha iyi" ifade etmektedir.



Şekil 22. 32 GB Veri Seti İşlemci kullanım Oranları

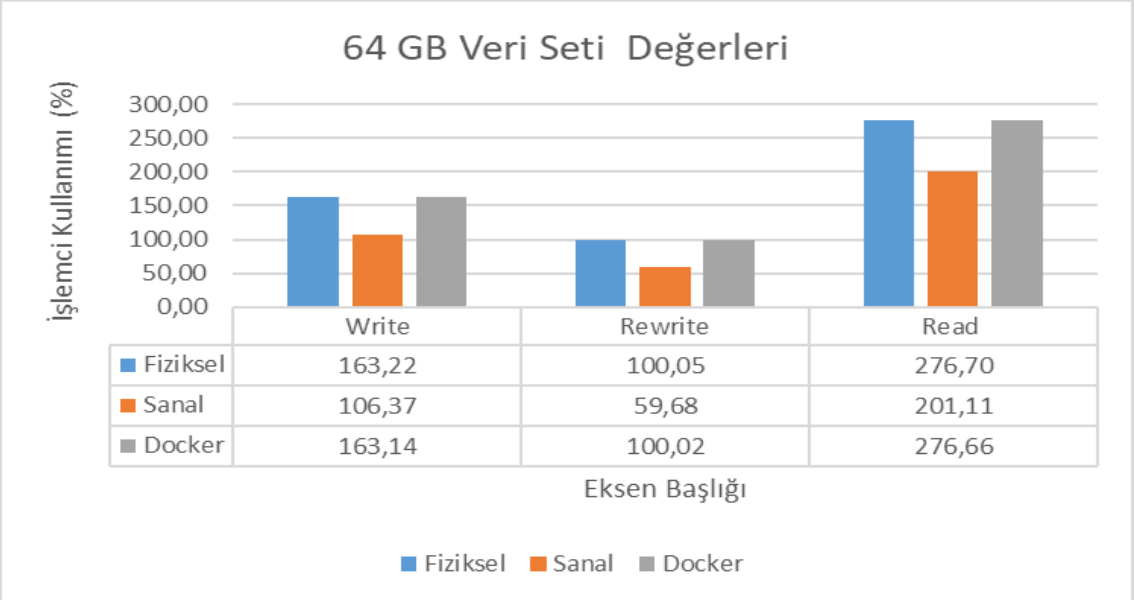


Şekil 23. 32 GB Veri Seti İşlemci kullanım Oranları

32 GB boyutundaki veri seti ile okuma, yazma işlemi yaparken, işlemci kullanım oranlarının her üç teknoloji için çok fazla değişmediği görülmektedir. Çalışılan veri setinin büyüklüğü genel sistemin performansında büyük fark oluşturmamıştır. Bir önceki veri setine göre sistem daha uzun süre grafikte görülen yük altıda çalışmıştır.

İşlemci kullanımın oranı en yüksek Vmware sanal makinesinde görülmektedir. Fiziksel makine ile Docker konteyneri birbirine çok yakın oranlarda işlemci kullanmışlardır.

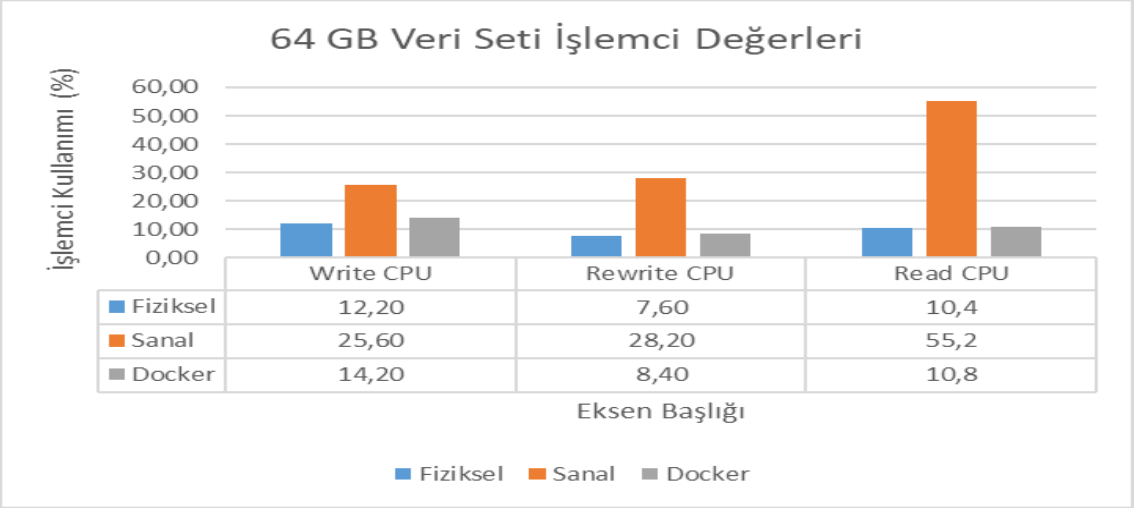
Aşağıdaki grafik 64 GB boyutundaki veri seti ile çalışılırken elde edilen, saniyede işlenen veri boyutunu göstermektedir. Bu grafiklerde “Daha yüksek değer daha iyi” ifade etmektedir.



Şekil 24. 64 GB Veri Okuma Yazma Değerleri

Test edilen sistemler, fiziksel belleğin dört katı büyüklüğündeki 64 GB boyutundaki veri seti ile test edilmiştir. Yapılan üç işlem için en iyi performansı fiziksel sunucu daha sonra Docker Konteynerinin göstermiştir. Diğer veri setlerinde olduğu gibi, Vmware Sanal makinesi ile Docker konteyneri arasındaki en büyük fark yazma işlemi sırasında gerçekleştiği görülmektedir.

Aşağıdaki grafik okuma, yazma ve tekrar yazma işlemleri sırasında işlemci kullanım oranını göstermektedir. Bu grafiklerde "Daha düşük değer daha iyi" ifade etmektedir.



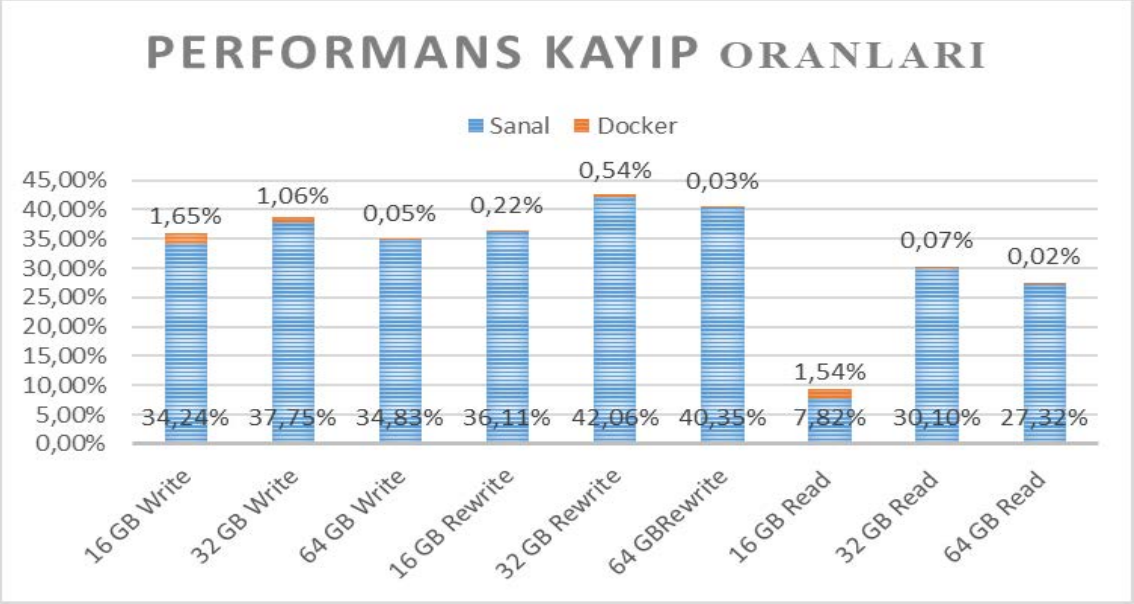
Şekil 25. 64 GB Veri Seti İşlemci Kullanım Oranları

64 GB boyutundaki veri seti ile okuma, yazma işlemi yaparken, işlemci kullanım oranlarının her üç teknoloji için diğer veri setleri ile yapılan testlere göre çok fazla değişmediği görülmektedir. Çalışılan veri setinin büyüklüğü genel sistemin performansında büyük fark oluşturmamıştır. Bir önceki veri setine göre sistem daha uzun süre grafikte görülen yük altıda çalışmıştır.

İşlemci kullanımın oranı en yüksek Vmware sanal makinesinde görülmektedir. Fiziksel makine ile Docker konteyneri birbirine çok yakın oranlarda işlemci kullanmışlardır.

6.3.2. Bonnie++ Sabit Disk G/Ç Sonuçlar ve Analiz

Yapılan testlerde bellek ve önbellek etkilerini azalmak için sistem belleğinden daha büyük veri setleri ile çalışılmıştır. Ölçümler sonunda elde edilen grafiklerde görüldüğü gibi sanallaştırma katmanı sabit disk operasyonlarında fiziksel sunuculara göre performans kaybına neden olmaktadır. Sanallaştırma katmanın türüne göre bu kayıplar değişmektedir. Şekil 24 fiziksel sistem performansı baz alınarak işlenen veri performans kayıpları gösterilmektedir.



Şekil 26. Performans Kayıp Oranları

Performans kayıp grafiklerinden anlaşılacağı üzere Vmware Sanal makinesi ardışık okuma işlemlerinde %7,82 ile %30 arasında performans kaybı göstermektedir. Veri boyutu, fiziksel belleğin üzerine çıktığı durumlarda performans kaybı çok daha fazla olduğu görülmektedir.

Vmware sanal makinesinde kullanılan hipervizör yazma ve tekrar yazma işlemleri yaparken, okuma işleminde olduğundan daha kötü performans gösterdiği görülmektedir.

Docker konteyneri açısından bakıldığında Vmware sanal makinesine göre çok daha az performans kaybı yaşadığı görülmektedir. Büyük veri setleri ile çalışıldığında, özellikle yazma işlemlerinde fiziksel makinaya çok yakın bir performans gösterdiği görülmektedir.

Docker konteyneri açısından okuma, yazma ve tekrar yazma işlemleri performans kaybı açısından, Vmware sanal makinesinde olduğu gibi ciddi farklılıklar yaratmamıştır.

Her üç işlem için de performans kayıpları birbirine yakın ve düşüktür. Ölçümlerde dikkat çeken nokta ise, veri boyutu büyüdükçe, konteynerin performans kaybının azalmasıdır.

6.4. Network Performans Testleri

İki teknolojinin network performansları değerlendirilirken network gecikme süresi ve network bant genişliği olmak üzere iki farklı açıdan değerlendirme yapılmıştır. Network gecikme süreleri "Qperf" aracı, network bant genişliği ölçümleri için "iperf" aracı kullanılmıştır.

Yapılan ölçümlerde yapılan testin özelliğine göre bazı durumlarda farklı mesaj boyutları bazı durumlarda farklı süreler boyunca testler uygulanmıştır.

Fiziksel makineye göre performans kayıpları işlemci performansı ölçümlerinde kullanılan formül ile yapılmıştır.

6.4.1. Network Bant Genişliği Ölçümleri

Network bant genişliği ölçümleri için iperf3 (IPerf, 2018) test aracı kullanılmıştır. Bu araç daha önce anlatıldığı IP ağlarında ulaşılabilecek en yüksek network bant genişliğine ulaşmaya çalışan sunucu istemci mimarisinde çalışan bir araçtır. Test edilecek sistem sunucu, ilgili yükün oluşturulacağı taraf ise istemci olarak ayarlanmıştır.

Bant genişliği testleri için, TCP protokolü kullanılmış ve her bir ortam için sırası ile 10, 30 ve 60 saniyelik testler koşulmuştur. Bu testler sonucunda elde edilen bant genişliği ve transfer edilen veri miktarı aşağıdaki tablolarda ve grafiklerde gösterilmiştir.

Tablo 5. Sanal Sunucu Network Bant Genişliği Değerleri

Sanal Sunucu İperf Network Bant Genişliği						
	10 sn.		**30 sn.**		**60 sn.**	
	Sender (Mbits/sec)	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**
	944	1100	942	3290	942	6580
	944	1100	943	3290	942	6580
	944	1100	943	3290	942	6570
	944	1100	942	3290	942	6580
	944	1100	943	3290	942	6580
Avg	**944**	**1100**	**942,6**	**3290**	**942**	**6578**

Tablo 6. Fiziksel Sunucu Bant Genişliği Değerleri

Fiziksel Sunucu İperf Network Bant Genişliği						
	10 sn.		**30 sn.**		**60 sn.**	
	Sender (Mbits/sec)	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	944	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
Avg	**943,2**	**1100**	**942**	**3290**	**942**	**6580**

Tablo 7. Docker Konteyneri Bant Genişliği Değerleri

Docker İperf Network Bant Genişliği						
	10 sn.		**30 sn.**		**60 sn.**	
	Sender (Mbits/sec)	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
Avg	**943**	**1100**	**942**	**3290**	**942**	**6580**

6.4.2. Network Bant Genişliği Ölçümleri Sonuçlar ve Analiz

Bant genişliği ölçümlerinde Docker konteyneri ve Sanal makine birbirine çok yakın değerler elde etmişlerdir. Gerek bant genişliği gerekse transfer edilen veri boyutlarına bakıldığında üç ortamın da aynı ya da birbirine çok yakın değerler elde ettikleri görülmüştür.

60 saniyelik testlerde Sanal makine tarafında sadece 2KByte kadar daha az veri transferi yapmış, Docker konteyneri tarafında ise fiziksel makine ile aynı değerler elde edilmiştir. Her üç ortam için test süresi 10 saniye baz alındığında 30 saniye ve 60saniyelik testlere göre ulaşılan network bant genişliğinin 1Mbit/sec kadar azaldığı görülmüştür.

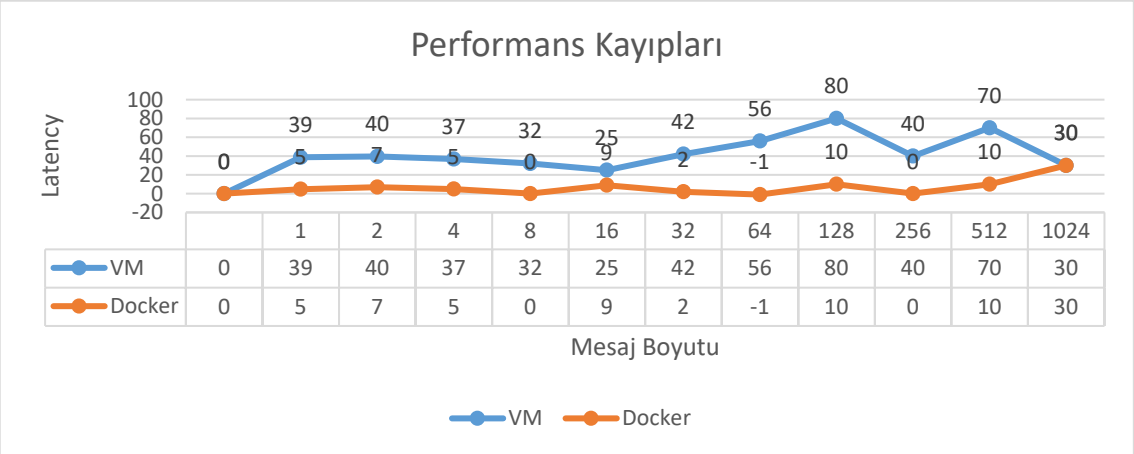
6.4.3. Network Gecikme Süreleri Ölçümleri

Network gecikme süresi ölçümleri için “Qperf” uygulaması kullanılmıştır. Bu uygulama istemci-sunucu mimarisinde çalışmaktadır. Test edilmekte olan sistem sunucu rolünde çalışmaktadır.

Network gecikme süreleri zamana ya da koşulan testin süresinden bağımsızdır. Network gecikme süresi etkileyen en büyük faktör, mesaj boyutları ve ortamda bulunan sunucu network kartları, ağ yönlendiricileri gibi network cihazlarının MTU (maximum transmission unit) boyutudur. Bu boyut istenildiğinde değiştirilebilmektedir ama bu değişikliğin geçerli olabilmesi için, üzerinde çalışılan veri iletim ağındaki tüm cihazlarda bu değişikliğin yapılması ve aynı değerin verilmesi gerekmektedir. Ortamda farklı değerler var ise en küçük değere göre network veri paketleri bölüneceğinde, daha yüksek girilmiş MTU değerleri anlamsız kalacaktır.

Test ortamımızda bu değer endüstri standartı 1500 bayt olarak kullanılmıştır. Testler boyunca 1 KB'den 1024 KB'ye kadar farklı mesaj boyutlarında testler yapılmış ve her ortam için network gecikme süreleri mikro saniye cinsinden hesaplanmıştır.

Şekil 27'de Mesaj Boyutuna bağlı olarak üç ortamın performans kayıp oranları ve Tablo 8'de elde edilen değerler gösterilmiştir.



Şekil 27. Performans Kayıpları

Tablo 8. Network Gecikme Değerleri

Ölçülen Değerler												
	Mesaj Boyutu (KB)	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
VM	**Latency (us)**	78	93	110	148	228	373	656	1210	2270	4460	8920
Docker	**Latency (us)**	43,9	60,3	78,1	116	212	333	599	1140	2230	4400	8920
Fiziksel	**Latency (us)**	39,2	53,4	73,2	116	203	331	600	1130	2230	4390	8890
Performans Kayıp Değerleri												
VM	**Gecikme Farkı %**	39	40	37	32	25	42	56	80	40	70	30
Docker	**Gecikme Farkı %**	5	7	5	0	9	2	-1	10	0	10	30

6.4.4. Network Gecikme Süreleri Ölçümleri ve Analizi

Yukarıdaki tablo ve grafiklerden anlaşılacağı üzere, farklı mesaj boyutlarında farklı gecikme süreleri hesaplanmıştır. Mesaj boyutu arttıkça Docker konteyneri ile hipervizör tabanlı Vmware sanal makinesinin gecikme süreleri birbirine yaklaşmıştır.

Özellikle küçük boyutlu mesaj testlerinde konteyner ile sanal makine arasındaki daha yüksek olduğu gözlemlenmiştir.

Docker konteyneri ile fiziksel makine arasında fark 8 ve 32 Kilobaytlık mesajlarda en aza inmiştir. En yüksek değerlere 1024 Kilobaytlık testlerde ulaşırken, Vmware sanal makinesi ile en düşük fark 16 Kilobaytlık testlerde, en yüksek fark ise 128 Kilobaytlık testlerde elde edilmiştir.

1024 Kilobayt ’tan daha düşük mesaj boyutlarında Docker konteyneri, sanal makineye göre network gecikmelerinde daha iyi performans gösterdiği görülmüştür.

6.5. LAMP Kümesi Performans Testleri

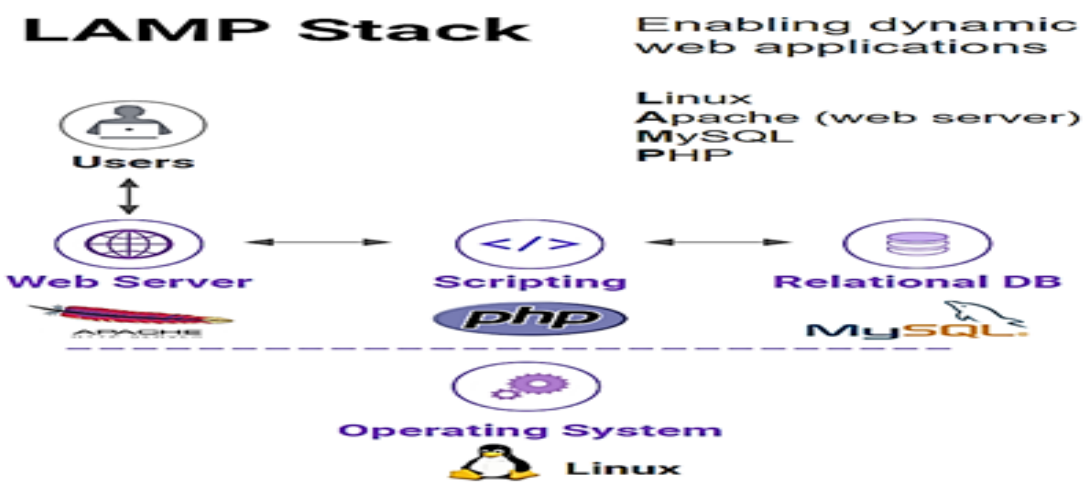
LAMP kümesi daha önceki bölümlerde anlatıldığı gibi endüstride çok sık kullanılan Linux, Apache Web Sunucusu, MySQL veri tabanı ve PHP uygulamalarını içeren bir ya da biden fazla sunucu grubu tarafından çalıştırılan uygulamalar bütünüdür. İhtiyaca göre bu uygulamalar tek bir sunucu üzerinde çalıştırılabildiği gibi, veri tabanı ayrı, uygulama sunucusu ayrı şekilde kullanımı da mevcuttur.

Bu çalışmada uygulamaların tamamını tek bir sanal sunucu ve tek bir Docker konteyneri içerisinde çalıştırıp Jmeter (Jmeter, 2018) yük testi aracı kullanılarak, farklı büyüklükler kullanılarak yük testleri yapılmıştır. Bu testlerdeki büyüklük eş zamanlı işlem yapan kullanıcı sayısı olarak ayarlanmıştır.

Tutarlı verilerin elde edilmesi ve ortamlar üzerinde gerekli yükün oluşturulabilmesi için hem sanal hem de Docker konteyneri için çekirdek sistem seviyesinde, kullanılabilir bellek seviyeleri 4GB’a düşürülmüştür.

Aşağıdaki şekil 26’da LAMP kümesi gösterilmiştir

Not: MySQL Oracle firması tarafından satın alındığı için, MySQL, MariaDB olarak geçmektedir ve testler bu veri tabanı sunucusu kullanılarak yapılmıştır.



Şekil 28. Lamp Stack (Hsu, 2017)

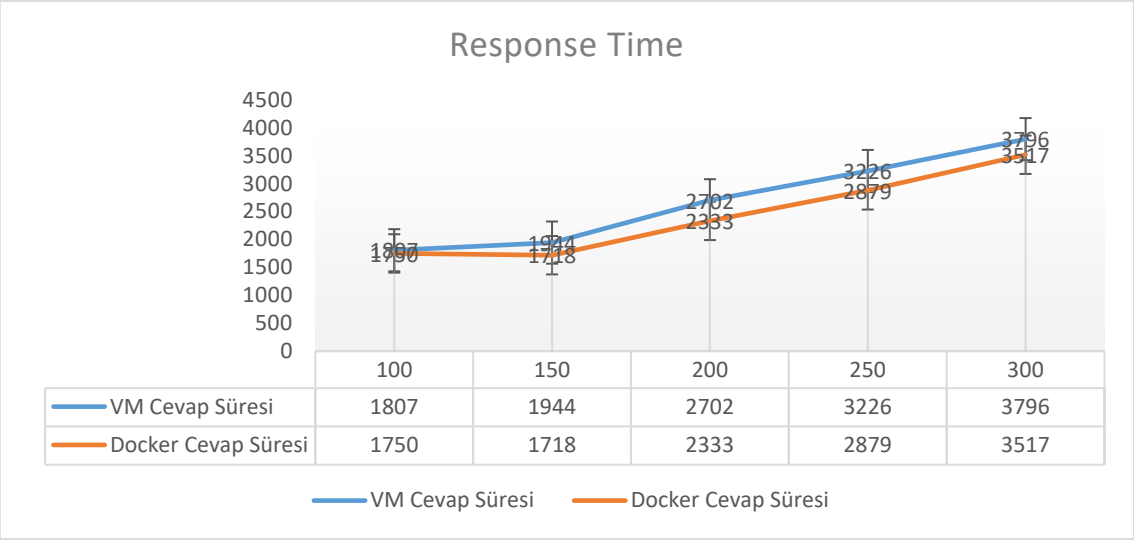
6.5.1. Veri tabanı INSERT İşlemleri

Veri tabanı insert işlemleri için MySQL veri tabanına "isim, soy isim ve e-mail adresi" bilgilerini kayıt eden PHP betiği yazılmış ve yük testi aracı ile 60 sn. boyunca bu sayfa çağrılarak veri tabanına kayıtların işlenmesi sağlanmıştır. Bu sayede iki ortamın 60 saniye boyunca yapabildiği işlem sayısı ve cevap süreleri, ortamların eş zamanlı işlem yapan kişi sayısı arttıkça performanslarındaki değişimler gözlemlenmiştir

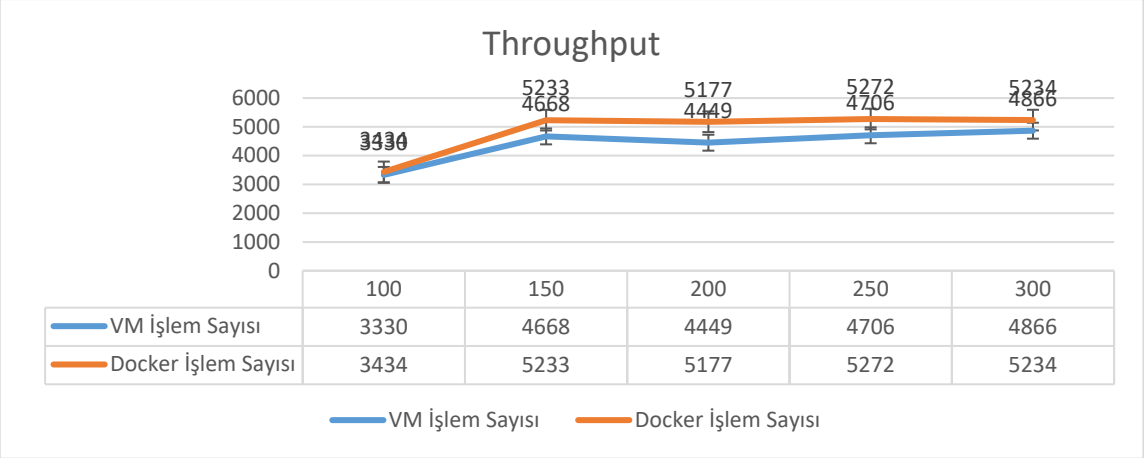
Her test öncesinde elde edilecek verilerin tutarlılığı için, veri tabanı tabloları silinip yeniden oluşturulmuştur. Aşağıdaki Tablo 9'da elde edilen değerleri gösterilmiştir. Cevap süresi grafiklerinde düşük olan, işlem sayısı tablolarında yüksek olan daha iyi demektir.

Tablo 9. İnsert İşlemleri Sonuçları

Thread Sayısı	**100**	**150**	**200**	**250**	**300**
VM İşlem Sayısı	3330	4668	4449	4706	4866
Docker İşlem Sayısı	3434	5233	5177	5272	5234
VM Cevap Süresi	1807	1944	2702	3226	3796
Docker Cevap Süresi	1750	1718	2333	2879	3517
VM MySQL	333200	466800	449900	470500	482100
Docker MySQL	343400	523100	517500	526600	523400



Şekil 29. Thread Sayısı ile Cevap Süresi Değişimi



Şekil 30. Thread Sayısı ile Yapılan İşlem Sayısı

6.5.2. Veri tabanı INSERT İşlemleri Sonuçlar ve Analizi

Yapılan testler sonucunda elde edilen verilere bakıldığında Docker konteynerinin Vmware sanal makinasından daha iyi performans gösterdiği görülmektedir.

Tablo 10. İşlem Sayıları

Thread Sayısı	100	150	200	250	300
VM İşlem Sayısı	3330	4668	4449	4706	4866
Docker İşlem Sayısı	3434	5233	5177	5272	5234
Performans Kayıp Oranı	3,028538148	10,796866	14,0621982	10,7359636	7,03095147

Tablo 11. Ortalama Cevap Süreleri

Thread Sayısı	100	150	200	250	300
VM Cevap Süresi	1807	1944	2702	3226	3796
Docker Cevap Süresi	1750	1718	2333	2879	3517
Performans Kayıp Oranı	3,154399557	11,6255144	13,6565507	10,7563546	7,34984194

Eş zamanlı 100 kullanıcı ile işlem yapılırken hem işlem sayısında hem de cevap süresi değerlerinde performans kayıp oranı %3 iken, eş zamanlı kullanıcı sayısı 200 olduğunda performans kayıp oranları %14 çıkmaktadır.

Veri tabanında oluşturulan kayıt sayılarında da benzer oranda kayıplar görülmektedir. Tablo 12'de Veri tabanı kayıt sayıları ve performans kayıpları görülmektedir.

Tablo 12. Veri tabanı Kayıt Sayıları

Thread Sayısı	100	150	200	250	300
VM MySQL	333200	466800	449900	470500	482100
Docker MySQL	343400	523100	517500	526600	523400
Performans Kayıp Oranı	2,97029703	10,76276047	13,06280193	10,65324725	7,890714559

6.5.3. Veri tabanı Okuma İşlemleri

Veri tabanı okuma işlemleri için, iki ortamdaki veri tabanları test verisi ile doldurulmuştur. Test datası olarak ad, soyadı ve e-mail adresi bilgilerini içeren 10bin adet kayıt oluşturulmuştur. Yine PHP betiği ile bu kayıtların tamamına sorgu gönderilip, 10 bin kayıtın listelenmesi istenmiştir.

Bu test veri tabanlarına INSERT işleminde olduğu gibi aynı tablo üzerinden eş zamanlı okuma işlemleri yapılarak performans ölçümleri yapılmıştır. Sırası ile 100, 150, 200, 250 ve 300 eş zamanlı işlemler 60 saniye boyunca çalıştırılıp değerler alınmıştır.

Aynı zamanda 60 saniye içerisinde eş zamanlı kullanıcı sayısı 0'dan 300 çıkacak ve 60 saniye boyunca 300 kullanıcı aktif olacak şekilde veri okuma testleri yapılmıştır.

Aşağıdaki tablo 13 ve 14'de eş zamanlı kullanıcı sayısına bağlı olarak yapılan işlem ve cevap süreleri gösterilmektedir.

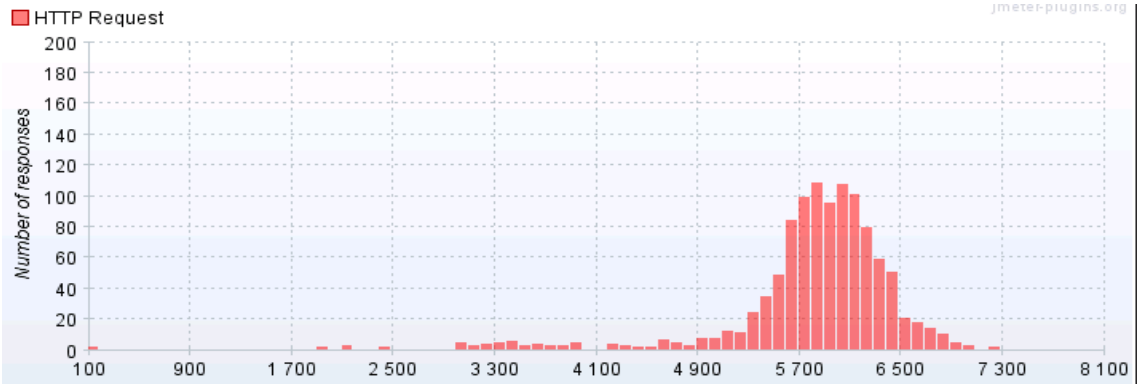
Tablo 13. İşlem Sayısı

Veri tabanından Okuma İşlem Sayısı						
Thread Sayısı	**100**	**150**	**200**	**250**	**300**	**0-300**
VM	1492	1518	1584	1611	1662	3060
Docker	1069	1078	1156	1201	1267	2304
Performans Kayıp Oranı	28,35	28,98	27,02	25,45	23,76	24,70

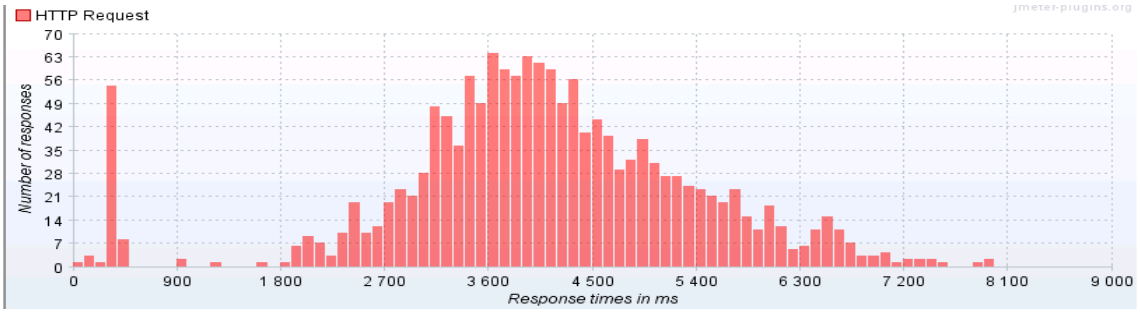
Tablo 14. Cevap Süresi

Veri tabanından Okuma Cevap Süresi						
Thread Sayısı	**100**	**150**	**200**	**250**	**300**	**0-300**
VM	4064	6091	11091	9842	11905	9520
Docker	5739	8641	7890	13738	15992	12842
Performans Kayıp Oranı	29,18	29,51	-40,57	28,35	25,55	25,862

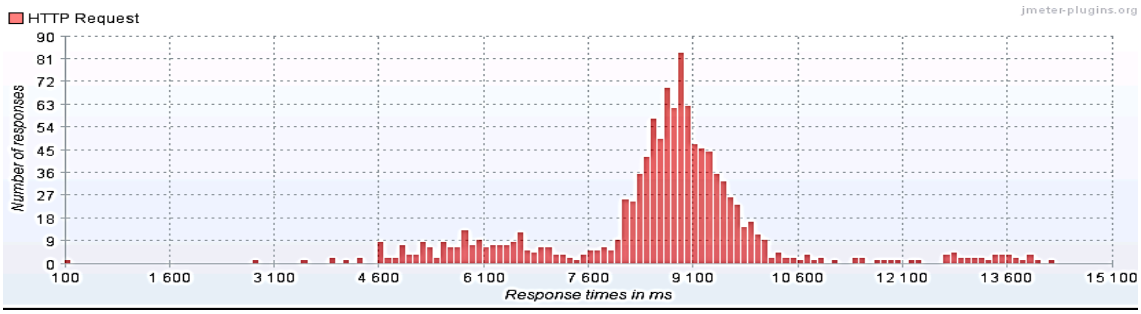
Aşağıdaki şekillerde aynı thread sayısında Docker ve Vmware sanal makinesinin, cevap sürelerinin thread sayısına göre dağılımı gösterilmektedir.



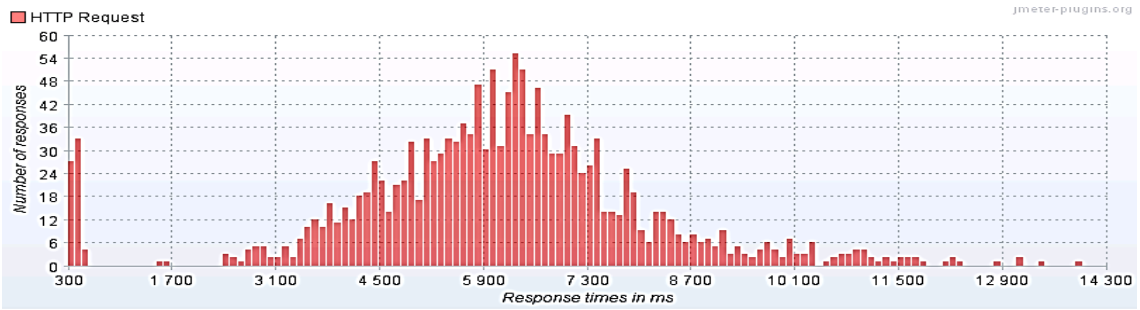
Şekil 31. Docker Cevap Süresi Dağılımı – 100 Thread



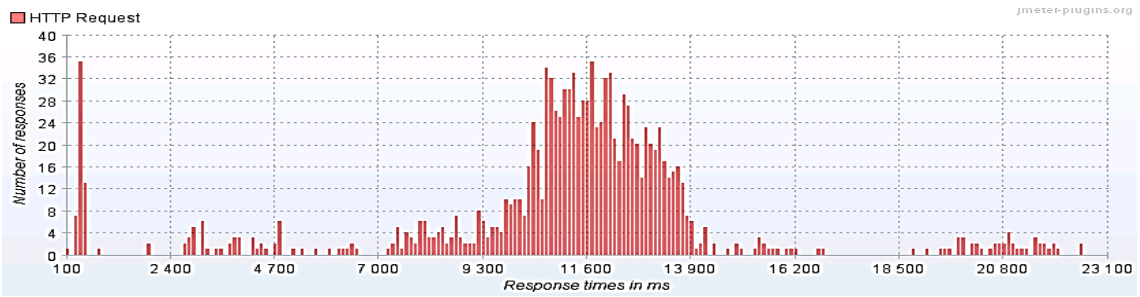
Şekil 32. Sanal Makine Cevap Süresi Dağılımı – 100 Thread



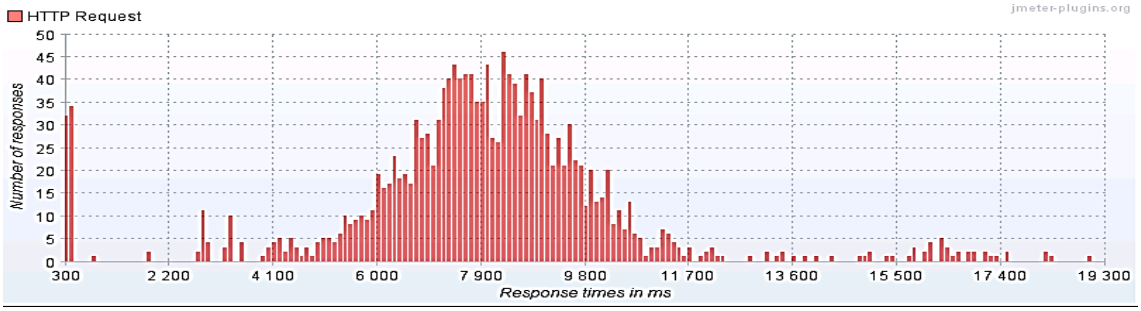
Şekil 33. Docker Cevap Süresi Dağılımı – 200 Thread



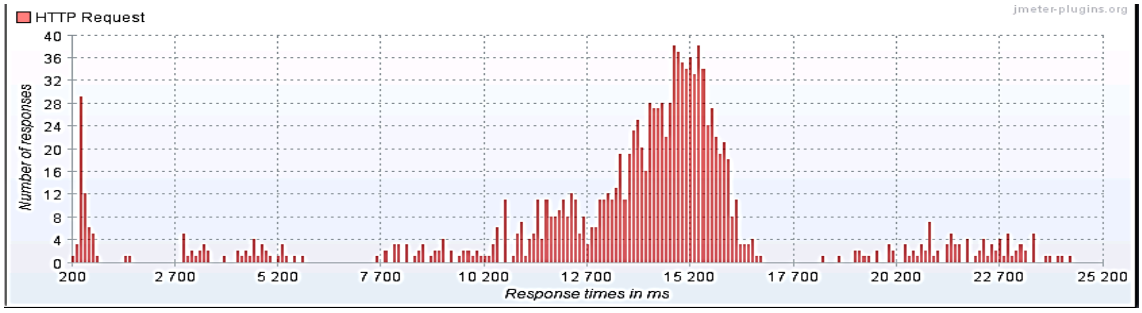
Şekil 34. Sanal Makine Cevap Süresi Dağılımı – 150 Thread



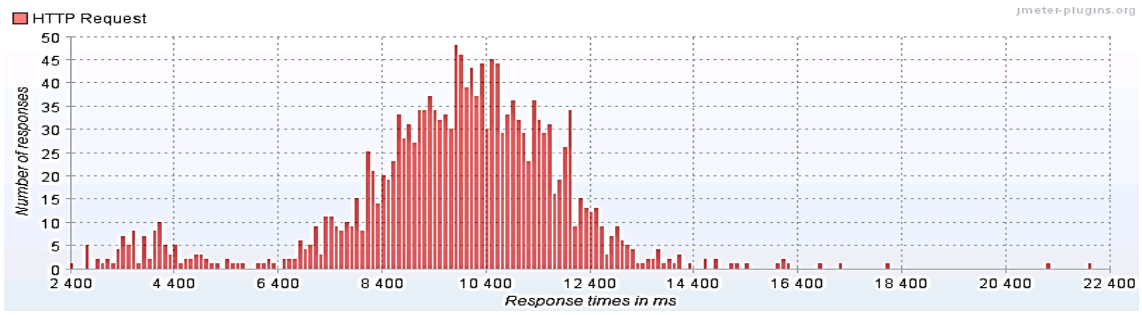
Şekil 35. Docker Cevap Süresi Dağılımı – 200 Thread



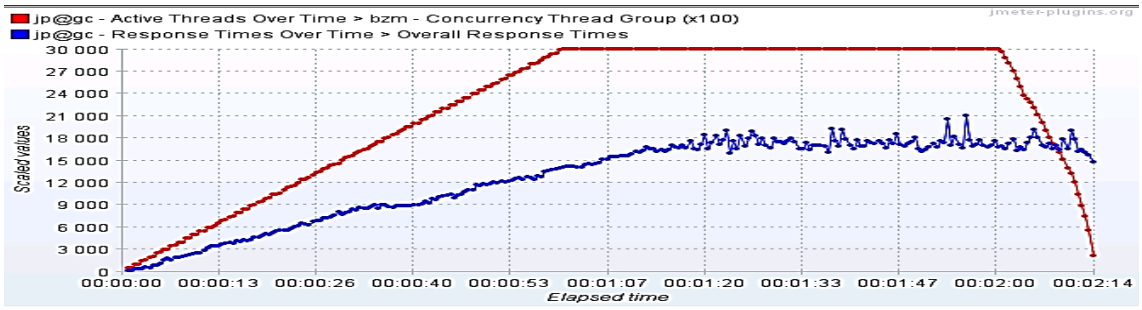
Şekil 36. Sanal Makine Cevap Süresi Dağılımı – 200 Thread



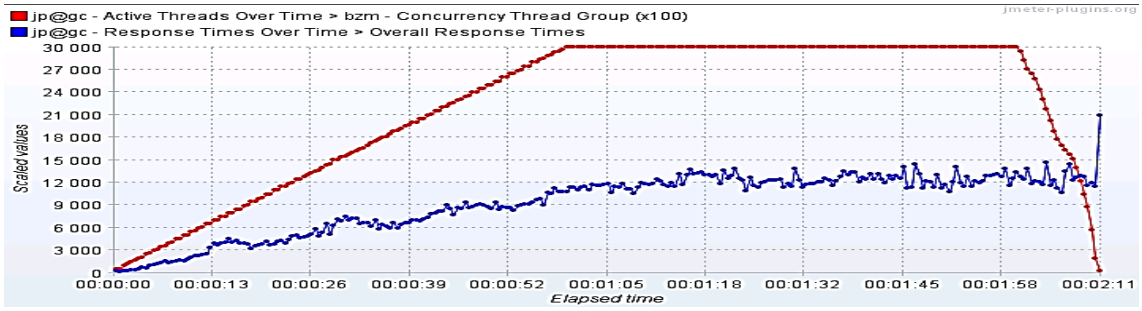
Şekil 37. Docker Cevap Süresi Dağılımı – 250 Thread



Şekil 38. Sanal Makine Cevap Süresi Dağılımı – 250 Thread



Şekil 39. Docker Aktif Thread Sayısı ve Cevap Süresi



Şekil 40. Sanal Makine Aktif Thread Sayısı ve Cevap Süresi

6.5.4. Veri tabanı Okuma İşlemleri Sonuçlar ve Analizi

Yapılan testlerden ve yukarıdaki tablo ve grafiklerden görüldüğü üzere, veri tabanından okuma işlemlerinde Vmware sanal makinesi, Docker konteynerine göre daha iyi sonuçlar vermiştir.

Tüm eş zamanlı kullanıcı işleri testlerinde yapılabilen okuma sayısı %24 ile %28 oranları arasında Sanal makine daha iyi performans göstermiştir. Yine aynı şekilde cevap sürelerine bakıldığında %25 ile %29 arasında daha iyi performans göstermiştir.

Veri tabanından okuma testleri hariç diğer testlerde Docker konteyner ’inin daha iyi performans gösterirken, okuma işlemlerinde Vmware sanal makinesinin daha iyi performans gösterdiği görülmüştür.

BÖLÜM 7. SONUÇ

Bu bölümde tez çalışması boyunca elde edilen verilerin ve sonuçların özetlendiği özet bölümü ve elde edilen bilgiler ışığında verilerin analizini içeren yargı bölümüne yer verilmiştir. Ayrıca deneysel bilişim biliminin bir gereği olarak, bu analizlerin ve sonuçları bir adım daha ileriye götürecek olan yeni çalışmalara yer verilmiştir.

Günümüzde kurumsal veri merkezlerinde ve bulut bilişim ortamlarında sıklık ile kullanılan sanallaştırma ve konteyner teknolojilerinin fiziksel sunucuya göre ve birbirlerine göre performans değerlerini karşılaştırmak, hangi teknolojinin hangi durumlar ya da hangi uygulama tipleri için daha uygun olduğunu görmek ve teknoloji seçimi için bir ipucu ve yol göstermek amacı ile çalışmanın önceki bölümlerinde anlatıldığı üzere birçok test ve analiz yapılmıştır.

Testleri ilk olarak, bir sunucunun performansı belirleyen donanımlarının yani işlemci, disk, bellek ve network donanımları ayrı ayrı bir etki alanı olarak ele alıp, bu ayrı alanlar üzerinde ayrı ayrı testler yapılmıştır. Sunucuların ve işletim sistemlerinin yapısı gereği bu alanları birbirinden tamamen ayrılıp test edilmesi imkânsız olduğundan, üçüncü parti yazılımlar ile bu etkileşimi en aza indirmeye çalıştık. Son olarak bu donanımların birlikte hizmet verdiği bir ortam üzerinde okuma, yazma işlemleri testlerini gerçekleştirdik. Yapılan testlerde iki teknoloji arasında farklılıkların, boyut, süre ve yapılan işlemlerin türüne göre farklılık gösterdiği görülmüştür.

İki teknoloji arasında en büyük fark depolama alanlarına yapılan yazma işlemlerinde ve işlemci yoğun işlem olan veri sıkıştırma işlemlerinde olduğu görülmektedir. En az fark ise network bant genişliği ve mesaj boyutuna bağlı olarak network gecikme sürelerinde görülmektedir.

Disk yazma işlemlerinde Vmware sanal makinesi, Docker konteynerine göre %30 un üzerinde kötü performans göstermiştir. Özellikle veri büyüklüğü fiziksel bellek miktarının üzerine çıktığı durumlarda Vmware sanal makinesi performans kayıpları %40'a kadar çıkmaktadır. Bu boyuttaki işlemlerde Docker konteyneri %1 kadar performans kaybı göstermiştir.

Yine Disk okuma işlemlerinde Docker konteyneri Vmware 'den veri büyüklüğüne bağlı olarak %6 ile %28 arasında daha iyi performans göstermiştir. Veri boyutu büyüdükçe performans farkının arttığı görülmektedir.

İşlemci performans testlerinde veri sıkıştırma, kayan nokta, şifreleme ve tam sayı matematik işlemleri yaptırılmıştır. Veri sıkıştırma işlemlerinde sırası ile $10^{8,}$ 10^{9} ve 10^{10} bayt boyutunda veri setleri kullanılmıştır. Bu işlemlerde veri boyutu arttıkça Docker konteynerinin ve Vmware sanal makinesinin performansı artmakta olduğu görülmüştür. 10^{8} bayt veri setinde %38 e yakın olan Vmware performans kaybı 10^{9} bayt veri setinde %13 düşmektedir. Tüm veri setlerinde Docker konteyneri Vmware sanal makinesine göre en az %12 oranında daha iyi performans göstermiştir.

İşlemci performans testlerinde yapılan diğer test sunucular üzerinde şifreleme, tamsayı hesaplamaları ve kayan noktalı matematik işlemleri yapılan testlerdir. Bu test sonuçlarında şifreleme işlemlerinde Docker konteyneri %23 daha iyi performans göstermiştir. Tamsayı ve kayan nokta matematik işlemlerinde aradaki farkın azaldığı görülmüştür. Bu işlemler için performans kayıp farkı %8-9 olarak ölçülmüştür.

Bellek işlemleri testlerinde bellek üzerine ekleme, kopyalama gibi işlemlerin veri transfer hızları ölçülmüştür. Bellek işlemlerinde Docker konteyneri fiziksel sunucuya göre en fazla okuma işleminde %1 performans kaybı yaşadığı görülmüştür. Vmware sanal makinesi ise tüm işlemlerde %18 kadar performans kaybı yaşadığı görülmüştür.

Network işlemelerinde bant genişliği ve network gecikmeleri ölçülmüştür. Bant genişliği ölçümlerinde iki teknolojinin performans kayıpları birbirlerine çok yakın ve %1 altında olduğu görülmüştür. Network gecikme sürelerinde ise mesaj boyutlarına göre farklılık görülmektedir.

Son olarak tüm bu bileşenlerin bir arada kullanıldığı LAMP kümesi okuma yazma işlemleri test edilmiştir. Bu testte sanallaştırma katmanın tüm bu bileşenleri ne kadar performanslı yönettiği de ölçülmüştür.

Lamp Kümesi veri tabanına yazma işlemlerinde eş zamanlı kullanıcı sayısına bağlı olarak performans değerleri değişmektedir. 100 kullanıcı eş zamanlı olarak veri tabanına yazma işlemi yaparken %3 olan performans ve cevap süresi farkı, 150 kullanıcıda %10-11, 200 kullanıcıda %13-14 olmaktadır. 300 kullanıcı sonrasında bu fark tekrar %7 olmaktadır. Eş zamanlı kullanıcı sayısı arttığında önbellekleme devreye girmek ve performans kaybı azalmaktadır.

Lamp kümesi Okuma işlemlerinde ise Docker konteyneri, Vmware sanal makinesine göre daha kötü performans göstermiştir. Gerek yapılan işlem sayısı gerekse cevap sürelerinde Vmware %28-30 civarında daha iyi performans göstermektedir.

Sanallaştırma ve konteyner teknolojileri kullanımı gerek işletim maliyetlerini azaltmak gerek sağladıkları yönetimsel kolaylıklar gerekse kaynakları verimli kullanmak için çok sık tercih edilen ve kullanılan teknolojilerdir. Bu çalışmanın birinci amacı bu teknolojilerin fiziksel sunucuya göre performans kayıplarının ölçülmesi ve bu iki teknolojilerin birbirlerine göre göreceli olarak performans ölçümlerinin yapılıp ortaya konmasıdır. Diğer bir amacı ise teknoloji seçiminde daha doğru ve verimli bir tercih yapmak için kıyaslanabilir veriler ortaya koymaktır.

İki sanallaştırma teknolojisinde araya giren sanallaştırma katmanı ve kullanılan teknoloji nedeniyle fiziksel sunucuya göre performans kayıpları ile karşılaşmaktadır. Yapılan bütün testlerde fiziksel sunucunun sanallaştırılmış ortamlardan daha iyi performans gösterdiği görülmüştür.

Özellikle işlemci yoğun ve disk yazma yoğun işlemler ve uygulamalar için Vmware sanal makinesi fiziksel ve Docker konteynerine göre çok daha kötü performans göstermektedir. Bu tarz uygulamalar için, fiziksel sunucu kullanımı ya da konteyner kullanımı daha uygun görünmektedir.

Bellek yoğun işlem yapan uygulamalarda performans kayıpları, işlemci ve disk işlemlerine göre daha olsa da burada %18 kadar bir kayıp görülmektedir. Bu tarz uygulamalarda performans açısından sırası ile fiziksel sunucu ve konteyner kullanmak daha avantajlı durmaktadır.

Network yoğun işlemlerde performans kayıplarının en az yaşandığı işlemlerdir.

Bu bilgiler doğrultusunda sanallaştırma teknoloji tercihinde konteyner kullanımı daha performanslı olarak görülmektedir.

Gelecekte yapılabilecek çalışmalar olarak, bu çalışmada tek bir fiziksel kaynak üzerinde tek bir sanal makine ya da tek bir konteyner çalıştırdık. Mevcut fiziksel kaynakları başka istemciler ile paylaşmadan sadece bu ortamlara ayırmış olduk. Günümüz veri merkezlerinde ve bulut bilişim ortamlarında bu fiziksel kaynaklar birden fazla sanal sunucu ya da konteyner tarafından paylaşılıyor olacaktır, böyle bir durumda sunucuların ya da konteynerlerin performans değerlerinde değişiklik olacaktır. Hipervizör katmanının ve çekirdek sistemin sınırlı kaynakları birden fazla sunucu ve konteyner arasında paylaştırılması ve yönetmesi ayrı problem ve performansı değerlendirilmesi gereken bir alan olarak karşımıza çıkacaktır.

EK'LER

1. Bellek Testleri Ham Verileri ve Betik Dosyaları

Tablo 15. Vmware Bellek Ham Sonuçları

Vmware Sunucu				
Function	**Best Rate MB/s**	**Avg time**	**Min time**	**Max time**
Copy:	11173.3	0.014352	0.014320	0.014368
Scale:	11413.1	0.014098	0.014019	0.014369
Add:	12233.0	0.019738	0.019619	0.020037
Triad:	12662.8	0.019027	0.018953	0.019331

Tablo 16. Fiziksel Bellek Ham Sonuçları

Fiziksel Sunucu				
Function	**Best Rate MB/s**	**Avg time**	**Min time**	**Max time**
Copy:	13696.2	0.012294	0.011682	0.014371
Scale:	13990.9	0.012047	0.011436	0.014110
Add:	14996.4	0.016774	0.016004	0.019303
Triad:	15407.3	0.016187	0.015577	0.018831

Tablo 17. Docker Konteyner Bellek Ham Sonuçları

Docker Konteyneri				
Function	**Best Rate MB/s**	**Avg time**	**Min time**	**Max time**
Copy:	13560.6	0.014109	0.011799	0.014689
Scale:	13850.3	0.013976	0.011552	0.014782
Add:	14995.3	0.018920	0.016005	0.019578
Triad:	15270.1	0.018492	0.015717	0.019405

Stream Betik Komutları

```
#!/bin/sh
# just run this on dokcerhosts
# Set up the dockerfile
if [ "$#" -ne 1 ]; then
echo "Usage: $0 numberOfSockets (specify as 1 or 2)"
exit 1
fi
if [ "$1" -eq 1 ]; then
rm -f Dockerfile
ln -s Dockerfile.oneSocket Dockerfile
elif [ "$1" -eq 2 ]; then
rm -f Dockerfile
```

```
ln -s Dockerfile.twoSocket Dockerfile
else
echo "Usage: $0 numberOfSockets (specify as 1 or 2)"
exit 1
fi
# building stream
make
# build the container
docker rm stream:latest
docker build -t stream .
mkdir -p results
log="results/docker.log"
now=`date`
echo "Running stream, started at $now"
echo "--------------------------------------------------------------------------------" >> $log
echo "Running stream, started at $now" >> $log
docker run --rm stream >> $log
docker rm stream:latest
echo "" >> $log
echo -n "Test completed at "; date
#!/bin/sh
# just run this on server02
if [ "$#" -ne 1 ]; then
echo "Usage: $0 numberOfSockets (specify as 1 or 2)"
exit 1
fi
if [ "$1" -eq 1 ]; then
numaopts=" --physcpubind=0-7,16-23 --localalloc "
echo "Running on one socket with numactl $numaopts"
elif [ "$1" -eq 2 ]; then
numaopts=" --physcpubind=0-31 --interleave=0,1 "
echo "Running on two sockets with numactl $numaopts"
else
echo "Usage: $0 numberOfSockets (specify as 1 or 2)"
exit 1
fi
# build the code
make
now=`date`
echo "Running stream, started at $now"
echo "--------------------------------------------------------------------------------" >>
results/linux.log
echo "Running stream with numaopts < $numaopts >, started at $now" >>
results/linux.log
time numactl $numaopts ./bin/stream.exe >> results/linux.log
echo "" >> results/linux.log
echo -n "Experiment completed at "; date
```

2. CPU Testleri Ham Verileri ve Betik Dosyaları

PXZ CPU Testi Betik Komutları

!/bin/sh
/usr/bin/time -a -o vm_time.txt -f "Percentage of CPU = %P/Elapsed time in a clock format = %E/CPU seconds Kernel = %S/CPU seconds User = %U " pxz -k -f -9 enwiki-latest-abstract.xml

Tablo 18. PXZ Sıkıştırma Sonuçları Ham Verileri 10^9 bayt

Enwik9 Yığın Dosyası 10^9 Bayts							
Docker Konteyneri		**Vmware Sanal Makinesi**		**Fiziksel Sunucu 16 Çekirdek**		**Fiziksel Sunucu 20 Çekirdek**	
% CPU	**Time**	**% CPU**	**Time**	**% CPU**	**Time**	**% CPU**	**Time**
413	213,95	465	183,52	429	200,93	427	189,78
442	191,95	427	218,08	449	199,22	449	190,67
433	188,2	443	209,12	431	206,2	431	205,3
446	181,7	432	215,94	434	208,14	434	191,08
446	187,74	437	226,97	435	205,75	435	191,59
423	209,61	437	223,25	424	211,96	424	190,45
426	193,94	437	220,79	434	204,31	434	196,79
425	196,55	423	230,82	442	204,65	442	190,81
439	186,08	423	229,04	450	202,89	450	181,4
448	182,99	441	207,66	421	210,56	421	189,57
Ortalama	**Ortalama**	**Ortalama**	**Ortalama**	**Ortalama**	**Ortalama**	**Ortalama**	**Ortalama**
434,1	193,271	436,5	216,519	434,9	205,461	434,7	191,744

Tablo 19. PXZ Sıkıştırma Sonuçları Ham Verileri 10^{10} bayt

enwik10 Yığın Dosyası 1010 Bayts							
Docker Konteyner		**Vmware**		**Fiziksel Sunucu 16 Core**		**Fiziksel Sunucu 20 Core**	
% CPU	**Time**	**% CPU**	**Time**	**% CPU**	**Time**	**% CPU**	**Time**
1541	636,03	1516	777,93	1242	812,07	1560	628,95
1553	627,09	1513	777,81	1236	815,43	1549	631,82
1558	626,44	1505	782,93	1223	829,41	1548	633,75
1552	624,93	1498	802,74	1245	807,38	1555	625,4
1518	650,8	1510	780,31	1221	824,01	1538	636,62
1559	626,94	1521	759,53	1249	810,41	1534	637,64
1541	632,91	1497	794,39	1223	826,4	1569	619,8
1547	631,54	1521	757,17	1239	814,42	1567	624,35
1561	630,49	1419	837,2	1241	815,02	1549	631,1
1555	630,33	1494	767,97	1245	808,12	1526	638,05
Ortalam	**Ortalam**	**Ortalam**	**Ortalam**	**Ortalama**	**Ortalama**	**Ortalama**	**Ortalama**
1548,5	631,75	1499,4	783,798	1236,4	816,267	1549,5	630,748

3. Sabit Disk Testleri Ham Verileri ve Betik Dosyaları

Fiziksel Makine Ham Verileri

format_version,bonnie_version,name,concurrency,seed,file_size,io_chunk_size,putc,putc_cpu,put_block,put_block_cpu,rewrite,rewrite_cpu,getc,getc_cpu,get_block,get_block_cpu,seeks,seeks_cpu,num_files,max_size,min_size,num_dirs,file_chunk_size,seq_create,seq_create_cpu,seq_stat,seq_stat_cpu,seq_del,seq_del_cpu,ran_create,ran_create_cpu,ran_stat,ran_stat_cpu,ran_del,ran_del_cpu,putc_latency,put_block_latency,rewrite_latency,getc_latency,get_block_latency,seeks_latency,seq_create_latency,seq_stat_latency,seq_del_latency,ran_create_latency,ran_stat_latency,ran_del_latency

1.97,1.97,docker-host-178,1,1527697629,4G,,794,99,185049,14,183110,11,2041,99,5597227,99,+++++,+++,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,19372us,215ms,59480us,4718us,304us,348us,344us,475us,507us,401us,629us,336us

1.97,1.97,docker-host-178,1,1527697629,4G,,766,99,185052,16,182977,10,1969,99,5769530,99,+++++,+++,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,16483us,576ms,57201us,5425us,307us,339us,408us,477us,508us,338us,346us,312us

1.97,1.97,docker-host-178,1,1527697629,4G,,772,99,184381,13,183229,10,2131,99,5719953,99,+++++,+++,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,16553us,616ms,59583us,4663us,301us,932us,556us,547us,651us,601us,341us,312us

1.97,1.97,docker-host-178,1,1527697629,4G,,767,99,183188,13,181652,11,1970,99,5772396,99,+++++,+++,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,15940us,125ms,66871us,5106us,39us,335us,352us,733us,458us,420us,337us,325us

1.97,1.97,docker-host-178,1,1527697629,4G,,767,99,181284,13,183407,10,1945,99,5795919,99,+++++,+++,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,16781us,127ms,58832us,5162us,341us,1430us,351us,681us,852us,588us,329us,319us

format_version,bonnie_version,name,concurrency,seed,file_size,io_chunk_size,putc,putc_cpu,put_block,put_block_cpu,rewrite,rewrite_cpu,getc,getc_cpu,get_block,get_block_cpu,seeks,seeks_cpu,num_files,max_size,min_size,num_dirs,file_chunk_size,seq_create,seq_create_cpu,seq_stat,seq_stat_cpu,seq_del,seq_del_cpu,ran_create,ran_create_cpu,ran_stat,ran_stat_cpu,ran_del,ran_del_cpu,putc_latency,put_block_latency,rewrite_latency,getc_latency,get_block_latency,seeks_latency,seq_create_latency,seq_stat_latency,seq_del_latency,ran_create_latency,ran_stat_latency,ran_del_latency

1.97,1.97,docker-host-178,1,1527704119,32G,,805,99,144282,11,89715,7,915,39,251291,9,1387,16,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,12888us,861ms,745ms,1221ms,158ms,119ms,437us,509us,366us,567us,325us,337us

1.97,1.97,docker-host-178,1,1527704119,32G,,803,99,180172,14,109176,8,1097,48,300618,11,850.5,19,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,16573us,778ms,455ms,1220ms,132ms,140ms,844us,550us,453us,352us,577us,368us

1.97,1.97,docker-host-178,1,1527704119,32G,,807,99,188153,13,114243,9,1171,51,314690,12,1414,15,16,,,,,

+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,10601us,605ms,501ms,1216ms,143ms,123ms,376us,711us,591us,317us,329us,321us
1.97,1.97,docker-host-178,1,1527704119,32G,,804,99,182133,14,111026,8,1075,49,303328,11,1177,27,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,12823us,822ms,310ms,1229ms,132ms,133ms,604us,1009us,624us,578us,579us,472us
1.97,1.97,docker-host-178,1,1527704119,32G,,805,99,175911,13,107820,8,1060,48,294980,11,1414,15,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,12307us,867ms,931ms,1217ms,133ms,128ms,370us,728us,377us,319us,334us,317us

Docker Konteyner Ham Verileri

format_version,bonnie_version,name,concurrency,seed,file_size,io_chunk_size,putc,putc_cpu,put_block,put_block_cpu,rewrite,rewrite_cpu,getc,getc_cpu,get_block,get_block_cpu,seeks,seeks_cpu,num_files,max_size,min_size,num_dirs,file_chunk_size,seq_create,seq_create_cpu,seq_stat,seq_stat_cpu,seq_del,seq_del_cpu,ran_create,ran_create_cpu,ran_stat,ran_stat_cpu,ran_del,ran_del_cpu,putc_latency,put_block_latency,rewrite_latency,getc_latency,get_block_latency,seeks_latency,seq_create_latency,seq_stat_latency,seq_del_latency,ran_create_latency,ran_stat_latency,ran_del_latency
1.97,1.97,33a09fc4e37f,1,1527752065,64G,,710,99,154744,13,94839,8,982,50,263571,10,568.7,18,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,21770us,933ms,746ms,1208ms,149ms,130ms,423us,1024us,1032us,544us,329us,839us
1.97,1.97,33a09fc4e37f,1,1527752065,64G,,719,99,186089,16,113450,9,1096,57,312016,12,642.7,24,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,20011us,688ms,615ms,580ms,134ms,142ms,591us,643us,658us,506us,344us,1003us
1.97,1.97,33a09fc4e37f,1,1527752065,64G,,720,99,173066,15,106143,9,991,54,291803,11,638.3,24,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,16273us,768ms,385ms,1221ms,141ms,190ms,566us,728us,824us,591us,341us,625us
1.97,1.97,33a09fc4e37f,1,1527752065,64G,,726,99,155382,13,96022,8,916,48,267710,10,666.9,21,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,18424us,892ms,498ms,1221ms,133ms,148ms,624us,678us,738us,624us,746us,757us
1.97,1.97,33a09fc4e37f,1,1527752065,64G,,771,99,166427,14,101660,8,1001,53,281384,11,591.2,9,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,13509us,363ms,596ms,1220ms,128ms,170ms,617us,678us,1044us,597us,941us,790us

format_version,bonnie_version,name,concurrency,seed,file_size,io_chunk_size,putc,putc_cpu,put_block,put_block_cpu,rewrite,rewrite_cpu,getc,getc_cpu,get_block,get_block_cpu,seeks,seeks_cpu,num_files,max_size,min_size,num_dirs,file_chunk_size,seq_create,seq_create_cpu,seq_stat,seq_stat_cpu,seq_del,seq_del_cpu,ran_create,ran_create_cpu,ran_stat,ran_stat_cpu,ran_del,ran_del_cpu,putc_latency,put_block_latency,rewrite_latency,getc_latency,get_block_latency,seeks_latency,seq_create_latency,seq_stat_latency,seq_del_latency,ran_create_latency,ran_stat_latency,ran_del_latency
1.97,1.97,33a09fc4e37f,1,1527748793,32G,,742,99,186695,15,113401,9,1080,61,313489,12,1039,26,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++

++,+++,22156us,815ms,503ms,386ms,131ms,147ms,528us,533us,944us,572us,341us,798us
1.97,1.97,33a09fc4e37f,1,1527748793,32G,,717,99,177501,15,110267,9,1010,56,303009,12,1075,25,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,14165us,739ms,341ms,1218ms,119ms,121ms,545us,689us,760us,867us,352us,712us
1.97,1.97,33a09fc4e37f,1,1527748793,32G,,764,99,174130,14,106860,9,1008,58,294564,11,1061,25,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,13622us,793ms,487ms,944ms,130ms,140ms,560us,691us,853us,1034us,349us,844us
1.97,1.97,33a09fc4e37f,1,1527748793,32G,,756,99,165986,14,101759,8,907,55,282916,11,1064,25,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,13235us,289ms,498ms,1218ms,129ms,125ms,532us,517us,840us,319us,345us,1008us
1.97,1.97,33a09fc4e37f,1,1527748793,32G,,758,99,157136,13,96842,8,941,54,269959,10,1054,24,16,,,,,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,12505us,965ms,564ms,1218ms,144ms,134ms,634us,702us,879us,411us,335us,1070us

VM Sanal makinesi ham verileri

format_version,bonnie_version,name,concurrency,seed,file_size,io_chunk_size,putc,putc_cpu,put_block,put_block_cpu,rewrite,rewrite_cpu,getc,getc_cpu,get_block,get_block_cpu,seeks,seeks_cpu,num_files,max_size,min_size,num_dirs,file_chunk_size,seq_create,seq_create_cpu,seq_stat,seq_stat_cpu,seq_del,seq_del_cpu,ran_create,ran_create_cpu,ran_stat,ran_stat_cpu,ran_del,ran_del_cpu,putc_latency,put_block_latency,rewrite_latency,getc_latency,get_block_latency,seeks_latency,seq_create_latency,seq_stat_latency,seq_del_latency,ran_create_latency,ran_stat_latency,ran_del_latency
1.97,1.97,vm,1,1527708464,64G,,629,99,109111,26,61676,28,1237,98,214231,56,206.5,1177,16,,,,,20505,93,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,31632us,672ms,420ms,35827us,173ms,1143ms,408us,764us,799us,396us,15us,1702us
1.97,1.97,vm,1,1527708464,64G,,627,99,108397,25,61829,30,1166,99,199920,54,218.0,1101,16,,,,,18920,37,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,33114us,754ms,470ms,29701us,141ms,780ms,497us,784us,1223us,881us,17us,915us
1.97,1.97,vm,1,1527708464,64G,,548,99,109771,27,61589,28,1172,99,211003,57,212.2,1179,16,,,,,20161,41,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,32955us,335ms,729ms,90830us,151ms,1099ms,856us,741us,1477us,729us,19us,526us
1.97,1.97,vm,1,1527708464,64G,,539,99,109044,23,60597,28,1198,99,206297,56,226.7,1185,16,,,,,21319,91,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,34943us,282ms,546ms,33064us,160ms,1228ms,735us,797us,1340us,473us,15us,390us
1.97,1.97,vm,1,1527708464,64G,,550,99,108306,27,59874,27,1192,99,198225,53,236.3,1169,16,,,,,20489,51,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,32751us,1536ms,422ms,157ms,151ms,875ms,802us,906us,792us,1004us,14us,1332us
format_version,bonnie_version,name,concurrency,seed,file_size,io_chunk_size,putc,putc_cpu,put_block,put_block_cpu,rewrite,rewrite_cpu,getc,getc_cpu,get_block,get_block_cpu,seeks,seeks_cpu,num_files,max_size,min_size,num_dirs,file_chunk_size,seq_create,seq_create_cpu,seq_stat,seq_stat_cpu,seq_del,seq_del_cpu,ran_create,ran_create_cpu,ran_stat,ran_stat_cpu,ran_del,ran_del_cpu,putc_latency,put_block_latency,rewrite_late

ncy,getc_latency,get_block_latency,seeks_latency,seq_create_latency,seq_stat_latency,
seq_del_latency,ran_create_latency,ran_stat_latency,ran_del_latency
1.97,1.97,vm,1,1527701645,32G,,561,99,108184,35,60722,27,1210,99,206297,54,316.
1,1118,16,,,,,24493,55,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,32
871us,159ms,289ms,34049us,148ms,1150ms,1800us,788us,804us,958us,10us,1803us
1.97,1.97,vm,1,1527701645,32G,,550,99,108755,25,60535,28,1217,99,206306,55,328.
4,1082,16,,,,,23608,49,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,33
723us,117ms,432ms,24638us,156ms,340ms,830us,767us,919us,971us,22us,1255us
1.97,1.97,vm,1,1527701645,32G,,553,99,109366,27,63326,30,1128,99,226962,59,314.
3,1107,16,,,,,26133,52,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,32
716us,204ms,407ms,26757us,135ms,1884ms,824us,764us,1538us,858us,14us,1075us
1.97,1.97,vm,1,1527701645,32G,,553,99,106740,26,57585,27,1213,99,184291,51,329.
0,1123,16,,,,,22110,61,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,33
579us,361ms,307ms,25586us,171ms,580ms,843us,1039us,1888us,1305us,16us,1471us
1.97,1.97,vm,1,1527701645,32G,,565,99,108963,33,66050,35,1171,99,200173,54,336.
7,1074,16,,,,,26317,58,+++++,+++,+++++,+++,+++++,+++,+++++,+++,+++++,+++,32
377us,297ms,345ms,33417us,144ms,866ms,1054us,776us,900us,361us,355us,390us

4. Network Testleri Ham Verileri ve Betik Dosyaları

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:44:15 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054255.134867.22ee
    TCP MSS: 1448 (default)
[4] local 172.27.23.217 port 49058 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ID] Interval   Transfer   Bandwidth   Retr Cwnd
[4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec   0   402 KBytes
[4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec   0   402 KBytes
[4]   9.00-10.00 sec   112 MBytes   941 Mbits/sec   0   402 KBytes
- -
Test Complete. Summary Results:
[ID] Interval   Transfer   Bandwidth   Retr
[4]   0.00-10.00 sec 1.10 GBytes   943 Mbits/sec   0   sender
[4]   0.00-10.00 sec 1.10 GBytes   941 Mbits/sec   receiver
CPU Utilization: local/sender 1.3% (0.3%u/1.0%s), remote/receiver 3.3% (0.3%u/3.0%s)

```
iperf Done.
########################################################################
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:44:38 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054278.010691.3b92
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49062 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   962 Mbits/sec    0    421 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   4.00-5.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   7.00-8.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   8.00-9.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   942 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.2%u/1.3%s), remote/receiver 6.5% (0.6%u/5.9%s)
iperf Done.
########################################################################
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:44:55 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054295.822218.6d9b
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49066 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   114 MBytes   960 Mbits/sec    0    383 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    383 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    402 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
```

```
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    402 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.2%u/1.2%s), remote/receiver 3.6% (0.4%u/3.2%s)
iperf Done.
########################################################################


root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:45:15 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545054315.208221.4f97
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49070 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    397 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]   8.00-9.00   sec   113 MBytes   947 Mbits/sec    0    525 KBytes
[  4]   9.00-10.00  sec   112 MBytes   942 Mbits/sec    0    525 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   944 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.1%u/1.3%s), remote/receiver 3.4% (0.5%u/2.9%s)
iperf Done.


root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
```

```
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:45:54 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054354.683660.5f68
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49078 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    410 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    410 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   942 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.1%u/1.3%s), remote/receiver 5.3% (0.5%u/4.9%s)
iperf Done.


root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:46:18 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054378.997535.6783
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49082 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   114 MBytes   960 Mbits/sec    0    389 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    389 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    406 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    406 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    406 KBytes
```

```
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    406 KBytes
[  4]   6.00-7.00   sec   112 MBytes   942 Mbits/sec    0    406 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    406 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    406 KBytes
[  4]   9.00-10.00  sec   113 MBytes   945 Mbits/sec    0    468 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  15.00-16.00  sec   112 MBytes   942 Mbits/sec    0    468 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  18.00-19.00  sec   112 MBytes   942 Mbits/sec    0    468 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
[  4]  28.00-29.00  sec   112 MBytes   942 Mbits/sec    0    468 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    468 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.1% (0.0%u/1.1%s), remote/receiver 4.4% (0.4%u/4.0%s)
iperf Done.
################################################################################
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:47:06 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054426.706577.11e3
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49086 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    400 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
```

```
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    400 KBytes
[  4]   9.00-10.00  sec   113 MBytes   950 Mbits/sec    0    578 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  11.00-12.00  sec   112 MBytes   942 Mbits/sec    0    578 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  13.00-14.00  sec   112 MBytes   942 Mbits/sec    0    578 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  15.00-16.00  sec   112 MBytes   942 Mbits/sec    0    578 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  17.00-18.00  sec   112 MBytes   942 Mbits/sec    0    578 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    578 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.1% (0.0%u/1.0%s), remote/receiver 4.0% (0.4%u/3.6%s)
iperf Done.
###########################################################################
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:47:48 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054468.728719.6cc2
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49090 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    404 KBytes
```

```
[  4]   1.00-2.00   sec   112 MBytes   942 Mbits/sec    0    404 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    423 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]   6.00-7.00   sec   112 MBytes   942 Mbits/sec    0    423 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]   9.00-10.00  sec   112 MBytes   942 Mbits/sec    0    423 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  23.00-24.00  sec   112 MBytes   942 Mbits/sec    0    423 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    423 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.1% (0.0%u/1.1%s), remote/receiver 4.6% (0.6%u/4.0%s)
iperf Done.
###############################################################################
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:48:31 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054511.590452.7764
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49094 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
```

```
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    403 KBytes
[  4]   1.00-2.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   9.00-10.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  20.00-21.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.2% (0.0%u/1.1%s), remote/receiver 12.1% (1.2%u/11.0%s)
iperf Done.
########################################################################
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:49:18 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054558.437906.7b54
    TCP MSS: 1448 (default)
```

[4] local 172.27.23.217 port 49098 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ID] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 115 MBytes 961 Mbits/sec 0 396 KBytes
[4] 1.00-2.00 sec 112 MBytes 941 Mbits/sec 0 396 KBytes
[4] 2.00-3.00 sec 112 MBytes 941 Mbits/sec 0 396 KBytes
[4] 3.00-4.00 sec 112 MBytes 941 Mbits/sec 0 396 KBytes
[4] 4.00-5.00 sec 112 MBytes 941 Mbits/sec 0 396 KBytes
[4] 5.00-6.00 sec 112 MBytes 942 Mbits/sec 0 414 KBytes
[4] 6.00-7.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 7.00-8.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 8.00-9.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 9.00-10.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 10.00-11.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 11.00-12.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 12.00-13.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 13.00-14.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 14.00-15.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 15.00-16.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 16.00-17.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 17.00-18.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 18.00-19.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 19.00-20.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 20.00-21.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 21.00-22.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 22.00-23.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 23.00-24.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 24.00-25.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 25.00-26.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 26.00-27.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 27.00-28.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 28.00-29.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
[4] 29.00-30.00 sec 112 MBytes 941 Mbits/sec 0 414 KBytes
- -
Test Complete. Summary Results:
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-30.00 sec 3.29 GBytes 942 Mbits/sec 0 sender
[4] 0.00-30.00 sec 3.29 GBytes 941 Mbits/sec receiver
CPU Utilization: local/sender 1.1% (0.0%u/1.1%s), remote/receiver 4.1% (0.6%u/3.6%s)
iperf Done.
##
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:50:10 GMT
Connecting to host 172.27.23.212, port 5201

Cookie: baremetal-177.1545054610.941894.7196
TCP MSS: 1448 (default)
[4] local 172.27.23.217 port 49102 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test

[ID]	Interval		Transfer	Bandwidth	Retr	Cwnd
[4]	0.00-1.00	sec	115 MBytes	961 Mbits/sec	0	413 KBytes
[4]	1.00-2.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	2.00-3.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	3.00-4.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	4.00-5.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	5.00-6.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	6.00-7.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	7.00-8.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	8.00-9.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	9.00-10.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	10.00-11.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	11.00-12.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	12.00-13.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	13.00-14.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	14.00-15.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	15.00-16.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	16.00-17.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	17.00-18.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	18.00-19.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	19.00-20.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	20.00-21.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	21.00-22.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	22.00-23.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	23.00-24.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	24.00-25.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	25.00-26.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	26.00-27.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	27.00-28.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	28.00-29.00	sec	112 MBytes	942 Mbits/sec	0	413 KBytes
[4]	29.00-30.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	30.00-31.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	31.00-32.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	32.00-33.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	33.00-34.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	34.00-35.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	35.00-36.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	36.00-37.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	37.00-38.00	sec	112 MBytes	942 Mbits/sec	0	413 KBytes
[4]	38.00-39.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	39.00-40.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	40.00-41.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes
[4]	41.00-42.00	sec	112 MBytes	941 Mbits/sec	0	413 KBytes

[4] 42.00-43.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 43.00-44.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 44.00-45.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 45.00-46.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 46.00-47.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 47.00-48.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 48.00-49.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 49.00-50.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 50.00-51.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 51.00-52.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 52.00-53.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 53.00-54.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 54.00-55.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 55.00-56.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 56.00-57.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 57.00-58.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 58.00-59.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
[4] 59.00-60.00 sec 112 MBytes 941 Mbits/sec 0 413 KBytes
- -
Test Complete. Summary Results:
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-60.00 sec 6.58 GBytes 942 Mbits/sec 0 sender
[4] 0.00-60.00 sec 6.58 GBytes 941 Mbits/sec receiver
CPU Utilization: local/sender 0.9% (0.0%u/0.9%s), remote/receiver 13.1% (1.6%u/11.4%s)
iperf Done.
##
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:52:07 GMT
Connecting to host 172.27.23.212, port 5201
Cookie: baremetal-177.1545054727.364015.5db6
TCP MSS: 1448 (default)
[4] local 172.27.23.217 port 49106 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
[ID] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 115 MBytes 960 Mbits/sec 0 397 KBytes
[4] 1.00-2.00 sec 112 MBytes 942 Mbits/sec 0 397 KBytes
[4] 2.00-3.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes
[4] 3.00-4.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes
[4] 4.00-5.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes
[4] 5.00-6.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes
[4] 6.00-7.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes
[4] 7.00-8.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes
[4] 8.00-9.00 sec 112 MBytes 941 Mbits/sec 0 397 KBytes

```
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    397 KBytes
[  4]  10.00-11.00  sec   113 MBytes   950 Mbits/sec    0    583 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  13.00-14.00  sec   112 MBytes   942 Mbits/sec    0    583 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  30.00-31.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  31.00-32.00  sec   112 MBytes   942 Mbits/sec    0    583 KBytes
[  4]  32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  36.00-37.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  37.00-38.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  39.00-40.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  40.00-41.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  46.00-47.00  sec   112 MBytes   942 Mbits/sec    0    583 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  50.00-51.00  sec   112 MBytes   942 Mbits/sec    0    583 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
```

```
[  4]  57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
[  4]  59.00-60.00  sec   112 MBytes   941 Mbits/sec    0    583 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.0%u/0.9%s), remote/receiver 3.4% (0.4%u/3.1%s)
iperf Done.

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:53:33 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545054813.175525.5e0f
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49110 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    404 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   8.00-9.00   sec   112 MBytes   942 Mbits/sec    0    424 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  10.00-11.00  sec   112 MBytes   942 Mbits/sec    0    424 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  14.00-15.00  sec   112 MBytes   942 Mbits/sec    0    424 KBytes
[  4]  15.00-16.00  sec   112 MBytes   942 Mbits/sec    0    424 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  19.00-20.00  sec   112 MBytes   942 Mbits/sec    0    424 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  22.00-23.00  sec   112 MBytes   942 Mbits/sec    0    424 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    424 KBytes
```

[4] 25.00-26.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 26.00-27.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 27.00-28.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 28.00-29.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 29.00-30.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 30.00-31.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 31.00-32.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 32.00-33.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 33.00-34.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 34.00-35.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 35.00-36.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 36.00-37.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 37.00-38.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 38.00-39.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 39.00-40.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 40.00-41.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 41.00-42.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 42.00-43.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 43.00-44.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 44.00-45.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 45.00-46.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 46.00-47.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 47.00-48.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 48.00-49.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 49.00-50.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 50.00-51.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 51.00-52.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 52.00-53.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 53.00-54.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 54.00-55.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 55.00-56.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 56.00-57.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 57.00-58.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 58.00-59.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 59.00-60.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
- -
Test Complete. Summary Results:
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-60.00 sec 6.58 GBytes 942 Mbits/sec 0 sender
[4] 0.00-60.00 sec 6.58 GBytes 941 Mbits/sec receiver
CPU Utilization: local/sender 1.1% (0.1%u/1.0%s), remote/receiver 4.5% (0.5%u/4.0%s)
iperf Done.
##
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 14:00:35 GMT

Connecting to host 172.27.23.212, port 5201
Cookie: baremetal-177.1545055235.349804.47f2
TCP MSS: 1448 (default)
[4] local 172.27.23.217 port 49114 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test

[ID]	Interval		Transfer	Bandwidth	Retr	Cwnd
[4]	0.00-1.00	sec	115 MBytes	961 Mbits/sec	0	399 KBytes
[4]	1.00-2.00	sec	112 MBytes	942 Mbits/sec	0	399 KBytes
[4]	2.00-3.00	sec	112 MBytes	941 Mbits/sec	0	399 KBytes
[4]	3.00-4.00	sec	112 MBytes	941 Mbits/sec	0	399 KBytes
[4]	4.00-5.00	sec	112 MBytes	941 Mbits/sec	0	399 KBytes
[4]	5.00-6.00	sec	112 MBytes	941 Mbits/sec	0	399 KBytes
[4]	6.00-7.00	sec	112 MBytes	942 Mbits/sec	0	416 KBytes
[4]	7.00-8.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	8.00-9.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	9.00-10.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	10.00-11.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	11.00-12.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	12.00-13.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	13.00-14.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	14.00-15.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	15.00-16.00	sec	112 MBytes	942 Mbits/sec	0	416 KBytes
[4]	16.00-17.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	17.00-18.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	18.00-19.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	19.00-20.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	20.00-21.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	21.00-22.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	22.00-23.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	23.00-24.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	24.00-25.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	25.00-26.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	26.00-27.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	27.00-28.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	28.00-29.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	29.00-30.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	30.00-31.00	sec	112 MBytes	942 Mbits/sec	0	416 KBytes
[4]	31.00-32.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	32.00-33.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	33.00-34.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	34.00-35.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	35.00-36.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	36.00-37.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	37.00-38.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	38.00-39.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	39.00-40.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes
[4]	40.00-41.00	sec	112 MBytes	941 Mbits/sec	0	416 KBytes

```
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  46.00-47.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  50.00-51.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  55.00-56.00  sec   112 MBytes   942 Mbits/sec    0    416 KBytes
[  4]  56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    416 KBytes
[  4]  59.00-60.00  sec   112 MBytes   942 Mbits/sec    0    416 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.1%u/1.0%s), remote/receiver 2.1% (0.2%u/1.9%s)
iperf Done.
```

```
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 14:01:59 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545055319.802051.4575
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49118 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   962 Mbits/sec    0    426 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
```

```
[  4]   7.00-8.00   sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  30.00-31.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]  31.00-32.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  36.00-37.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]  37.00-38.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  39.00-40.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  40.00-41.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  46.00-47.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  50.00-51.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  52.00-53.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  54.00-55.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
```

```
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  56.00-57.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]  57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  59.00-60.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.0%u/1.0%s), remote/receiver 12.9%
(1.4%u/11.4%s)
iperf Done.
###############################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 90
Connecting to host 172.27.23.212, port 5201
[  4] local 172.27.23.217 port 49212 connected to 172.27.23.212 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   114 MBytes   960 Mbits/sec    0    392 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    392 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    411 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]  14.00-15.00  sec   113 MBytes   946 Mbits/sec    0    508 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  28.00-29.00  sec   112 MBytes   942 Mbits/sec    0    508 KBytes
```

```
[  4] 29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 30.00-31.00  sec   112 MBytes   942 Mbits/sec    0    508 KBytes
[  4] 31.00-32.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 36.00-37.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 37.00-38.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 39.00-40.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 40.00-41.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 46.00-47.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 50.00-51.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 59.00-60.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 60.00-61.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 61.00-62.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 62.00-63.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 63.00-64.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 64.00-65.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 65.00-66.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 66.00-67.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 67.00-68.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 68.00-69.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 69.00-70.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 70.00-71.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 71.00-72.00  sec   112 MBytes   942 Mbits/sec    0    508 KBytes
[  4] 72.00-73.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 73.00-74.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 74.00-75.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 75.00-76.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4] 76.00-77.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
```

```
[  4]  77.00-78.00  sec   112 MBytes   942 Mbits/sec    0    508 KBytes
[  4]  78.00-79.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  79.00-80.00  sec   112 MBytes   942 Mbits/sec    0    508 KBytes
[  4]  80.00-81.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  81.00-82.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  82.00-83.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  83.00-84.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  84.00-85.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  85.00-86.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  86.00-87.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  87.00-88.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  88.00-89.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
[  4]  89.00-90.00  sec   112 MBytes   941 Mbits/sec    0    508 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-90.00  sec  9.87 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-90.00  sec  9.86 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.0%u/1.0%s), remote/receiver 12.9% (1.4%u/11.4%s)
iperf Done.
###############################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 12:53:48 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545051228.334033.6c21
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 48996 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    404 KBytes
[  4]   1.00-2.00   sec   112 MBytes   942 Mbits/sec    0    404 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   4.00-5.00   sec   112 MBytes   942 Mbits/sec    0    404 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    404 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    404 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
```

```
[ 4]  0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.1%u/1.3%s), remote/receiver 0.5% (0.0%u/0.4%s)
iperf Done.
#########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 12:54:35 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545051275.810010.0a79
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49000 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    410 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    410 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    410 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.1%u/1.3%s), remote/receiver 2.0% (0.2%u/1.9%s)
iperf Done.
#########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 12:54:59 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545051299.069898.36a2
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49004 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
```

```
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    409 KBytes
[  4]   1.00-2.00   sec   112 MBytes   942 Mbits/sec    0    409 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    409 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    409 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.3%u/1.1%s), remote/receiver 4.3% (0.3%u/4.0%s)
iperf Done.
########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 12:55:21 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545051321.589221.49a0
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49008 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    399 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    399 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    419 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    419 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.4% (0.1%u/1.2%s), remote/receiver 4.2% (0.3%u/4.0%s)
iperf Done.
```

```
#########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 10 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 12:55:47 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545051347.132885.79f6
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49012 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    402 KBytes
[  4]   1.00-2.00   sec   112 MBytes   942 Mbits/sec    0    419 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    419 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    419 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.10 GBytes   943 Mbits/sec    0             sender
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.3% (0.1%u/1.2%s), remote/receiver 6.5% (0.8%u/5.7%s)
iperf Done.
#########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:31:01 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545053461.487347.14d8
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49018 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    407 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
```

```
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   4.00-5.00   sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  15.00-16.00  sec   112 MBytes   942 Mbits/sec    0    426 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    426 KBytes
[  4]  22.00-23.00  sec   113 MBytes   951 Mbits/sec    0    621 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    621 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.1% (0.0%u/1.1%s), remote/receiver 0.1% (0.0%u/0.1%s)
iperf Done.
########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:31:51 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545053511.527031.3f12
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49022 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
```

```
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    411 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    411 KBytes
[  4]   4.00-5.00   sec   112 MBytes   942 Mbits/sec    0    431 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    431 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.1% (0.1%u/1.0%s), remote/receiver 5.8% (0.7%u/5.1%s)
iperf Done.
################################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:33:43 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545053623.502677.7e14
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49026 connected to 172.27.23.212 port 5201
```

Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ID] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 115 MBytes 960 Mbits/sec 0 406 KBytes
[4] 1.00-2.00 sec 112 MBytes 942 Mbits/sec 0 406 KBytes
[4] 2.00-3.00 sec 112 MBytes 941 Mbits/sec 0 406 KBytes
[4] 3.00-4.00 sec 112 MBytes 941 Mbits/sec 0 406 KBytes
[4] 4.00-5.00 sec 112 MBytes 941 Mbits/sec 0 406 KBytes
[4] 5.00-6.00 sec 112 MBytes 941 Mbits/sec 0 406 KBytes
[4] 6.00-7.00 sec 112 MBytes 943 Mbits/sec 0 424 KBytes
[4] 7.00-8.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 8.00-9.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 9.00-10.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 10.00-11.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 11.00-12.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 12.00-13.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 13.00-14.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 14.00-15.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 15.00-16.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 16.00-17.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 17.00-18.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 18.00-19.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 19.00-20.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 20.00-21.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 21.00-22.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 22.00-23.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 23.00-24.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 24.00-25.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 25.00-26.00 sec 112 MBytes 942 Mbits/sec 0 424 KBytes
[4] 26.00-27.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 27.00-28.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 28.00-29.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
[4] 29.00-30.00 sec 112 MBytes 941 Mbits/sec 0 424 KBytes
- -
Test Complete. Summary Results:
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-30.00 sec 3.29 GBytes 942 Mbits/sec 0 sender
[4] 0.00-30.00 sec 3.29 GBytes 941 Mbits/sec receiver
CPU Utilization: local/sender 1.0% (0.1%u/0.9%s), remote/receiver 3.0% (0.3%u/2.7%s)
iperf Done.
##

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:34:23 GMT
Connecting to host 172.27.23.212, port 5201

```
    Cookie: baremetal-177.1545053663.894887.0bda
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49030 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    403 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    403 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   9.00-10.00  sec   112 MBytes   943 Mbits/sec    0    444 KBytes
[  4]  10.00-11.00  sec   112 MBytes   942 Mbits/sec    0    444 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  16.00-17.00  sec   112 MBytes   942 Mbits/sec    0    444 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    444 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  3.29 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-30.00  sec  3.29 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.1%u/0.9%s), remote/receiver 8.2% (1.1%u/7.0%s)
iperf Done.
###########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 30 -V
iperf 3.1.3
```

Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:35:23 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545053723.463614.5c5e
    TCP MSS: 1448 (default)
[4] local 172.27.23.217 port 49034 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 30 second test
[ID] Interval   Transfer   Bandwidth   Retr Cwnd
[4]   0.00-1.00   sec   115 MBytes   962 Mbits/sec   0   430 KBytes
[4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4]   9.00-10.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 10.00-11.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 11.00-12.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 12.00-13.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 13.00-14.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 14.00-15.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 15.00-16.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 16.00-17.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 17.00-18.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 18.00-19.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 19.00-20.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 20.00-21.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 21.00-22.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 22.00-23.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 23.00-24.00 sec   112 MBytes   942 Mbits/sec   0   430 KBytes
[4] 24.00-25.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 25.00-26.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 26.00-27.00 sec   112 MBytes   942 Mbits/sec   0   430 KBytes
[4] 27.00-28.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
[4] 28.00-29.00 sec   112 MBytes   942 Mbits/sec   0   430 KBytes
[4] 29.00-30.00 sec   112 MBytes   941 Mbits/sec   0   430 KBytes
- -
Test Complete. Summary Results:
[ID] Interval   Transfer   Bandwidth   Retr
[4]   0.00-30.00 sec 3.29 GBytes   942 Mbits/sec   0   sender
[4]   0.00-30.00 sec 3.29 GBytes   941 Mbits/sec   receiver
CPU Utilization: local/sender 1.1% (0.0%u/1.0%s), remote/receiver 4.8% (0.4%u/4.4%s)
iperf Done.
##

```
root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:36:10 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545053770.168652.6196
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49038 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    402 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    402 KBytes
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   5.00-6.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]   8.00-9.00   sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  10.00-11.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  13.00-14.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  19.00-20.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  24.00-25.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  26.00-27.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  29.00-30.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  30.00-31.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  31.00-32.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
```

```
[  4]  35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  36.00-37.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  37.00-38.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  39.00-40.00  sec   112 MBytes   942 Mbits/sec    0    421 KBytes
[  4]  40.00-41.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  46.00-47.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    421 KBytes
[  4]  48.00-49.00  sec   113 MBytes   951 Mbits/sec    0    622 KBytes
[  4]  49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  50.00-51.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    622 KBytes
[  4]  59.00-60.00  sec   112 MBytes   942 Mbits/sec    0    622 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.1%u/1.0%s), remote/receiver 7.8% (1.0%u/6.8%s)
iperf Done.
###############################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC
2018 x86_64
Time: Mon, 17 Dec 2018 13:38:05 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545053885.272959.4814
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49042 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60
second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   114 MBytes   960 Mbits/sec    0    390 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    390 KBytes
```

```
[  4]   2.00-3.00   sec   112 MBytes   942 Mbits/sec    0    407 KBytes
[  4]   3.00-4.00   sec   112 MBytes   942 Mbits/sec    0    407 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  18.00-19.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    407 KBytes
[  4]  20.00-21.00  sec   113 MBytes   950 Mbits/sec    0    592 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  26.00-27.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  30.00-31.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  31.00-32.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  36.00-37.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  37.00-38.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  39.00-40.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  40.00-41.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  42.00-43.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  46.00-47.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  49.00-50.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
```

```
[  4]  50.00-51.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  56.00-57.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
[  4]  57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    592 KBytes
[  4]  59.00-60.00  sec   112 MBytes   942 Mbits/sec    0    592 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 1.0% (0.1%u/0.9%s), remote/receiver 4.9% (0.5%u/4.4%s)
iperf Done.
########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:39:24 GMT
Connecting to host 172.27.23.212, port 5201
      Cookie: baremetal-177.1545053964.020377.4ab0
      TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49046 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec    0    403 KBytes
[  4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   3.00-4.00   sec   112 MBytes   942 Mbits/sec    0    403 KBytes
[  4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   5.00-6.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   6.00-7.00   sec   112 MBytes   942 Mbits/sec    0    403 KBytes
[  4]   7.00-8.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]   9.00-10.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  10.00-11.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  11.00-12.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  12.00-13.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  13.00-14.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  14.00-15.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  15.00-16.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  16.00-17.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
```

```
[  4]  17.00-18.00  sec   112 MBytes   941 Mbits/sec    0    403 KBytes
[  4]  18.00-19.00  sec   113 MBytes   950 Mbits/sec    0    587 KBytes
[  4]  19.00-20.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  20.00-21.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  21.00-22.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  22.00-23.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  23.00-24.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  24.00-25.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  25.00-26.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  26.00-27.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  27.00-28.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  28.00-29.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  29.00-30.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  30.00-31.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  31.00-32.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  36.00-37.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  37.00-38.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  39.00-40.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  40.00-41.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  46.00-47.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  50.00-51.00  sec   112 MBytes   942 Mbits/sec    0    587 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  57.00-58.00  sec   112 MBytes   942 Mbits/sec    0    587 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
[  4]  59.00-60.00  sec   112 MBytes   941 Mbits/sec    0    587 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
```

CPU Utilization: local/sender 0.9% (0.1%u/0.9%s), remote/receiver 8.4% (0.9%u/7.5%s)
iperf Done.
##

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:40:52 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054052.670108.2c3d
    TCP MSS: 1448 (default)
[4] local 172.27.23.217 port 49050 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
[ID] Interval   Transfer   Bandwidth   Retr Cwnd
[4]   0.00-1.00   sec   115 MBytes   961 Mbits/sec   0   414 KBytes
[4]   1.00-2.00   sec   112 MBytes   941 Mbits/sec   0   414 KBytes
[4]   2.00-3.00   sec   112 MBytes   941 Mbits/sec   0   414 KBytes
[4]   3.00-4.00   sec   112 MBytes   941 Mbits/sec   0   414 KBytes
[4]   4.00-5.00   sec   112 MBytes   941 Mbits/sec   0   414 KBytes
[4]   5.00-6.00   sec   112 MBytes   942 Mbits/sec   0   414 KBytes
[4]   6.00-7.00   sec   112 MBytes   941 Mbits/sec   0   414 KBytes
[4]   7.00-8.00   sec   112 MBytes   943 Mbits/sec   0   434 KBytes
[4]   8.00-9.00   sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4]   9.00-10.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 10.00-11.00 sec   112 MBytes   942 Mbits/sec   0   434 KBytes
[4] 11.00-12.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 12.00-13.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 13.00-14.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 14.00-15.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 15.00-16.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 16.00-17.00 sec   112 MBytes   941 Mbits/sec   0   434 KBytes
[4] 17.00-18.00 sec   112 MBytes   942 Mbits/sec   0   434 KBytes
[4] 18.00-19.00 sec   113 MBytes   951 Mbits/sec   0   631 KBytes
[4] 19.00-20.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 20.00-21.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 21.00-22.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 22.00-23.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 23.00-24.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 24.00-25.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 25.00-26.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 26.00-27.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 27.00-28.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 28.00-29.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 29.00-30.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 30.00-31.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes
[4] 31.00-32.00 sec   112 MBytes   941 Mbits/sec   0   631 KBytes

```
[  4]  32.00-33.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  33.00-34.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  34.00-35.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  35.00-36.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  36.00-37.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  37.00-38.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  38.00-39.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  39.00-40.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  40.00-41.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  41.00-42.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  42.00-43.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  43.00-44.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  44.00-45.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  45.00-46.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  46.00-47.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  47.00-48.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  48.00-49.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  49.00-50.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  50.00-51.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  51.00-52.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  52.00-53.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  53.00-54.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  54.00-55.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  55.00-56.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  56.00-57.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  57.00-58.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  58.00-59.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
[  4]  59.00-60.00  sec   112 MBytes   941 Mbits/sec    0    631 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec  6.58 GBytes   942 Mbits/sec    0             sender
[  4]   0.00-60.00  sec  6.58 GBytes   941 Mbits/sec                  receiver
CPU Utilization: local/sender 0.9% (0.0%u/0.9%s), remote/receiver 6.7% (0.7%u/6.0%s)
iperf Done.
###########################################################################

root@baremetal-177:~# iperf3 -c 172.27.23.212 -t 60 -V
iperf 3.1.3
Linux baremetal-177 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Time: Mon, 17 Dec 2018 13:42:10 GMT
Connecting to host 172.27.23.212, port 5201
    Cookie: baremetal-177.1545054130.982936.14f4
    TCP MSS: 1448 (default)
[  4] local 172.27.23.217 port 49054 connected to 172.27.23.212 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 60 second test
```

[ID]	Interval		Transfer	Bandwidth	Retr	Cwnd
[4]	0.00-1.00	sec	114 MBytes	960 Mbits/sec	0	393 KBytes
[4]	1.00-2.00	sec	112 MBytes	942 Mbits/sec	0	410 KBytes
[4]	2.00-3.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	3.00-4.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	4.00-5.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	5.00-6.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	6.00-7.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	7.00-8.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	8.00-9.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	9.00-10.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	10.00-11.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	11.00-12.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	12.00-13.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	13.00-14.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	14.00-15.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	15.00-16.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	16.00-17.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	17.00-18.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	18.00-19.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	19.00-20.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	20.00-21.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	21.00-22.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	22.00-23.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	23.00-24.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	24.00-25.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	25.00-26.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	26.00-27.00	sec	112 MBytes	941 Mbits/sec	0	410 KBytes
[4]	27.00-28.00	sec	113 MBytes	951 Mbits/sec	0	618 KBytes
[4]	28.00-29.00	sec	112 MBytes	942 Mbits/sec	0	618 KBytes
[4]	29.00-30.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	30.00-31.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	31.00-32.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	32.00-33.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	33.00-34.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	34.00-35.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	35.00-36.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	36.00-37.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	37.00-38.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	38.00-39.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	39.00-40.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	40.00-41.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	41.00-42.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	42.00-43.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	43.00-44.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	44.00-45.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	45.00-46.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes
[4]	46.00-47.00	sec	112 MBytes	941 Mbits/sec	0	618 KBytes

[4] 47.00-48.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 48.00-49.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 49.00-50.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 50.00-51.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 51.00-52.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 52.00-53.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 53.00-54.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 54.00-55.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 55.00-56.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 56.00-57.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 57.00-58.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 58.00-59.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
[4] 59.00-60.00 sec 112 MBytes 941 Mbits/sec 0 618 KBytes
- -
Test Complete. Summary Results:
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-60.00 sec 6.58 GBytes 942 Mbits/sec 0 sender
[4] 0.00-60.00 sec 6.58 GBytes 941 Mbits/sec receiver
CPU Utilization: local/sender 1.0% (0.0%u/1.0%s), remote/receiver 7.9% (0.7%u/7.2%s)
iperf Done.

Tablo 20. Vmware Iperf Ham Verileri

Vmware Iperf Ham Verileri						
	10 sn		**30 sn**		**60 sn**	
	Sender (Mbits/sec)	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**
	944	1100	942	3290	942	6580
	944	1100	943	3290	942	6580
	944	1100	943	3290	942	6570
	944	1100	942	3290	942	6580
	944	1100	943	3290	942	6580
Avg	**944**	**1100**	**942,6**	**3290**	**942**	**6578**

Tablo 21. Fiziksel Iperf Ham Verileri

Fiziksel Iperf Ham Verileri						
	10 sn		**30 sn**		**60 sn**	
	Sender (Mbits/sec)	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**	**Sender (Mbits/sec)**	**Transfered (KBytes)**
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	944	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
Avg	**943,2**	**1100**	**942**	**3290**	**942**	**6580**

Tablo 22. Docker Iperf Ham Verileri

Docker Iperf Ham Verileri						
	10 sn		30 sn		60 sn	
	Sender (Mbits/sec)	Transfered (KBytes)	Sender (Mbits/sec)	Transfered (KBytes)	Sender (Mbits/sec)	Transfered (KBytes)
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
	943	1100	942	3290	942	6580
Avg	**943**	**1100**	**942**	**3290**	**942**	**6580**

5. LAMP Kümesi Testleri Ham Verileri ve Betik Dosyaları

Jmeter Betikleri

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="4.0" jmeter="4.0 r1823414">
<hashTree>
<TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
<stringProp name="TestPlan.comments"></stringProp>
<boolProp name="TestPlan.functional_mode">false</boolProp>
<boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
<boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
<elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
<collectionProp name="Arguments.arguments"/>
</elementProp>
<stringProp name="TestPlan.user_define_classpath"></stringProp>
</TestPlan>
<hashTree>
<ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group-DOCKER" enabled="true">
<stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
<elementProp name="ThreadGroup.main_controller" elementType="LoopController" guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
<boolProp name="LoopController.continue_forever">false</boolProp>
<intProp name="LoopController.loops">-1</intProp>
</elementProp>
<stringProp name="ThreadGroup.num_threads">350</stringProp>
<stringProp name="ThreadGroup.ramp_time">1</stringProp>
<boolProp name="ThreadGroup.scheduler">true</boolProp>
<stringProp name="ThreadGroup.duration">60</stringProp>
<stringProp name="ThreadGroup.delay"></stringProp>
</ThreadGroup>
```

<hashTree>
<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP Request" enabled="true">
<elementProp name="HTTPsampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" enabled="true">
<collectionProp name="Arguments.arguments"/>
</elementProp>
<stringProp name="HTTPSampler.domain">172.27.23.212</stringProp>
<stringProp name="HTTPSampler.port"></stringProp>
<stringProp name="HTTPSampler.protocol">http</stringProp>
<stringProp name="HTTPSampler.contentEncoding"></stringProp>
<stringProp name="HTTPSampler.path">/read.php</stringProp>
<stringProp name="HTTPSampler.method">GET</stringProp>
<boolProp name="HTTPSampler.follow_redirects">true</boolProp>
<boolProp name="HTTPSampler.auto_redirects">false</boolProp>
<boolProp name="HTTPSampler.use_keepalive">true</boolProp>
<boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
<stringProp name="HTTPSampler.embedded_url_re"></stringProp>
<stringProp name="HTTPSampler.connect_timeout"></stringProp>
<stringProp name="HTTPSampler.response_timeout"></stringProp>
</HTTPSamplerProxy>
<hashTree/>
<ResultCollector guiclass="ViewResultsFullVisualizer" testclass="ResultCollector" testname="View Results Tree" enabled="true">
<boolProp name="ResultCollector.error_logging">true</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>

<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report" enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">C:\Users\ALIDOG\Documents\tez\READ-DCOKER</stringProp>
<boolProp name="useGroupName">true</boolProp>

</ResultCollector>
<hashTree/>
<ResultCollector guiclass="RespTimeGraphVisualizer" testclass="ResultCollector" testname="Response Time Graph" enabled="false">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">C:\Users\ALIDOG\Documents\tez\100th-120sn-docker-gra</stringProp>
<stringProp name="RespTimeGraph.graphtitle">Response Time Graph</stringProp>
<stringProp name="RespTimeGraph.interval">1000</stringProp>
</ResultCollector>
<hashTree/>
<kg.apc.jmeter.vizualizers.CorrectedResultCollector guiclass="kg.apc.jmeter.vizualizers.TransactionsPerSecondGui" testclass="kg.apc.jmeter.vizualizers.CorrectedResultCollector" testname="jp@gc - Transactions per Second" enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>

<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
</kg.apc.jmeter.vizualizers.CorrectedResultCollector>
<hashTree/>
<kg.apc.jmeter.vizualizers.CorrectedResultCollector
guiclass="kg.apc.jmeter.vizualizers.ResponseTimesOverTimeGui"
testclass="kg.apc.jmeter.vizualizers.CorrectedResultCollector" testname="jp@gc -
Response Times Over Time" enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>

<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">500</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
</kg.apc.jmeter.vizualizers.CorrectedResultCollector>
<hashTree/>
<kg.apc.jmeter.vizualizers.CorrectedResultCollector
guiclass="kg.apc.jmeter.vizualizers.BytesThroughputOverTimeGui"
testclass="kg.apc.jmeter.vizualizers.CorrectedResultCollector" testname="jp@gc -
Bytes Throughput Over Time" enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">

<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
</kg.apc.jmeter.vizualizers.CorrectedResultCollector>
<hashTree/>
<kg.apc.jmeter.vizualizers.CompositeResultCollector
guiclass="kg.apc.jmeter.vizualizers.CompositeGraphGui"
testclass="kg.apc.jmeter.vizualizers.CompositeResultCollector" testname="jp@gc -
Composite Graph" enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>

<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">500</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
<collectionProp name="COMPOSITE_CFG">
<collectionProp name=""/>
<collectionProp name=""/>
</collectionProp>
</kg.apc.jmeter.vizualizers.CompositeResultCollector>
<hashTree/>
<kg.apc.jmeter.vizualizers.CorrectedResultCollector
guiclass="kg.apc.jmeter.vizualizers.ResponseTimesDistributionGui"
testclass="kg.apc.jmeter.vizualizers.CorrectedResultCollector" testname="jp@gc -
Response Times Distribution" enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>

<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">100</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
</kg.apc.jmeter.vizualizers.CorrectedResultCollector>
<hashTree/>
</hashTree>
</hashTree>
</hashTree>
</jmeterTestPlan>

PHP Betikleri

```
<?php
$servername = "localhost";
$username = "root";
$password = "XXXXX";
$dbname = "mysql";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);}
for($i = 0; $i<10000; $i++) {
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
if ($conn->query($sql) === TRUE) {
   echo "New record created successfully";
} else {
   echo "Error: " . $sql . "<br>" . $conn->error;
}
}
$conn->close();
?>
<?php
$servername = "localhost";
$username = "root";
$password = "XXX";
$dbname = "mysql";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
$sql = "SELECT firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
   // output data of each row
   while($row = mysqli_fetch_assoc($result)) {
      echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"].
"<br>";
   }
} else {
   echo "0 results";
}
mysqli_close($conn);
?>
```

KAYNAKÇA

Amies, Alex, Qiang Guo Tong & Harm Sluiman. "Infrastructure as a Service Cloud Concepts," **Developing and Hosting Applications on the Cloud**: 22. IBM Press, 2012.

Apparao P., Iyer, R., Zhang, X., Newell, D., Adelmeyer, T. *Characterization & Analysis of a Server Consolidation Benchmark on Xen*. 2007.

Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz & Andy Konwinski. "A View Of Cloud Computing," *Communications of the ACM53*: 50-58. New York: 2010.

Axboe, Jens. "GITHUB," **GITHUB**. *https://github.com/axboe/fio*. 2018. (24 Nisan 2018).

Biederman, Eric W. "Multiple Instances of the Global Linux Namespaces.," *Linux Symposium, Volume One*: 102-112. Ottawa: 2006.

Chou, Yung. "Microsoft TechNet," Microsoft, *https://blogs.technet.microsoft.com/yungchou/2010/11/15/cloud-computing-primer-for-it-pros/*. 2010. (01 Mayıs 2018).

Claassen, Joris, Ralph Koning & Paola Grosso. "Linux containers networking: Performance and scalability of kernel modules.," *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*: 713-716. Istanbul, Turkey: 2016.

Coffey, Patrick , Niyanta Mogre, Jason Beliveau & Andrew Harner. *Benchmarking the Amazon Elastic Compute Cloud (EC2)*. 2011.

Coker, Russell. "Bonnie++," **Coker**. *https://www.coker.com.au/bonnie++/*. 2018. (11 Nisan 2018).

Cordeiro, Thiago, Douglas Damalio, Nadilma Pereira, Patricia Endo, André Palhares, Glauco Gonçalves, Djamel Sadok & Judith Kelner. "Open Source Cloud Computing Platforms," *Grid and Cooperative Computing (GCC), 2010 9th International Conference*: 1-6. Nanjing, China: 2011.

cs.virginia.edu. "STREAM Benchmark Reference Information," Department of Computer Science School of Engineering and Applied Science , *https://www.cs.virginia.edu/stream/ref.html*. 2018. (12 Nisan 2018).

Datanyze. "Datanyze Market Share," *https://www.datanyze.com/market-share/containerization*. 2017. (16 Nisan 2018).

datanyze.com. "datanyze.com," datanyze.com, *https://www.datanyze.com/market-share/virtualization/*. 2017. (17 Şubat 2018).

de la Fé Herrero, José Manuel & Sergio Vega Gutiérrez. "LinuX Containers for Opennebula," **Open Nebula**. *https://opennebula.org/lxc-containers-for-opennebula/*. 2016. (18 Ocak 2019).

Dergan, Alan. "See/Hear/Do," *http://alandargan.com/using-vagrant-docker/*. 2014. (02 Eylül 2018).

"Docker CE," Docker Inc, *https://docs.docker.com/release-notes/docker-ce/#18030-ce-2018-03-21*. 2018. (20 Mayıs 2018).

docs.docker.com. "docker-architecture," Docker Inc, *https://docs.docker.com/engine/docker-overview/#docker-architecture*. 2018. (26 Nisan 2018).

docs.docker.com. "Understand the architecture," *https://docs.docker.com/engine/docker-overview/#docker-architecture*. 2018. (21 Eylül 2018).

Feitelson, Dror G. *Experimental Computer Science:The Need for a Cultural Change*. 2. Jerusalem: School of Computer Science and Engineering The Hebrew University of Jerusalem, 2006.

Felter, Wes, Alexandre Ferreira, Ram Rajamony & Juan Rubio. "An updated performance comparison of virtual machines and Linux containers.," *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium*: Philadelphia, PA, USA: 2015.

Fuertes, W., Vergara, J. E. L., Pincha, J., Aules, H., Jacome, L., Grijalva M. *Analytical Expression to Predict the Overhead Produced by the VMware and Xen Virtualization Tools*. Spain: 2012.

GeekBench. "Geekbench 4.," Primate Labs, *https://www.geekbench.com/index.html*. 2017. (23 Nisan 2018).

GeekBench. "Geekbench 4 CPU Workloads," *https://www.geekbench.com/doc/geekbench4-cpu-workloads.pdf*. 2018. (26 Mayıs 2018).

Gentoo Foundation. "Sysbench," **Gentoo Foundation**. Gentoo Linux, *https://wiki.gentoo.org/wiki/Sysbench*. 2017. (20 Nisan 2018).

George, Johann. "linux," **linuxDie**. *https://linux.die.net/man/1/qperf*. 2018. (11 Nisan 2018).

Herrod, Stephen Alan. "Systems Research And Development At VMware," **ACM SIGOPS Operating Systems Review**. 44: 4, 2010.

Hoyte, Doug. "The Virtual Memory Toucher," *https://hoytech.com/vmtouch/*. 2017. (02 Mayıs 2018).

"HP ProLiant BL460c Gen8 Server Blade," *https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c03262548*. [t.y.]. (11 Mayıs 2018).

Hsu, Edward. "The SMACK Stack is the New LAMP Stack," **Mesosphere.com**. *https://mesosphere.com/blog/smack-stack-new-lamp-stack/*. 2017. (17 Ocak 2019).

Hwang, Jinho , Sai Zeng & Frederick Wu. "A component-based performance comparison of four hypervisors.," *Integrated Network Management IFIP/IEEE International Symposium*: 269-273. Ghent, Belgium: 2013.

Hwanju, Kim. "CPU virtualization and scheduling," *https://www.slideshare.net/HwanjuKim/3cpu-virtualization-and-scheduling?qid=88008596-0cb4-4b0d-9a60-15c96d61d7df&v=&b=&from_search=1*. 2015. (22 Eylül 2018).

IOZone.org. "IOzone Filesystem Benchmark," *http://www.iozone.org/*. 2018. (23 Nisan 2018).

IPerf. "iPerf - The ultimate speed test tool for TCP, UDP and SCTP," *https://iperf.fr/*. 2018. (25 Nisan 2018).

Jmeter, Apache. "Apache Jmeter," **Apache Software Foundation**. *https://jmeter.apache.org/index.html*. 2018. (02 Ekim 2018).

Li, Dingding , Mianxiong Dong, Yong Tang, Laurence T. Yang, Kaoru Ota & Gansen Zhao. "Triple-L: Improving CPS Disk I/O Performance in a Virtualized NAS Environment.," **IEEE SYSTEMS JOURNAL**. 11, 1: 152-155, Mart 2017.

Li, Zheng, Maria Kihl, Qinghua Lu & Jens A. Andersson. "Performance Overhead Comparison between Hypervisor and Container based Virtualization.," *IEEE 31st International Conference on Advanced Information Networking and Applications*: 1-8. Taipei City, Taiwan: 2017.

Li, Dingding , Xiaofei Liao, Hai Jin, Bingbing Zhou & Qi Zhang. "A New Disk I/O Model of Virtualized Cloud Environment.," *IEEE Transactions on Parallel and Distributed Systems*: 1130-1132. 2013.

Linux Containers. "Linux Containers," *https://linuxcontainers.org/lxc/introduction/*. 2018. (22 Haziran 2018).

Li, Zheng, Liam OBrien, Rainbow Cai & He Zhang. "Towards a taxonomy of performance evaluation of commercial Cloud services," *IEEE Fifth International Conference on Cloud Computing*: Honolulu: 2012.

Li, Zheng, Liam O'Brien & Maria Kihl. "DoKnowMe: Towards a Domain Knowledgedriven Methodology for Performance Evaluation," **SIGMETRICS Performance Evaluation Review**. 01 03 2016.

Liu, Jiajun & Qinfen Hao. "Research on Optimizing KVM's Network Performance.," *Internet Technology and Applications (iTAP), 2011 International Conference IEEE*: Wuhan, China: 2011.

Mahoney, Matt. "Large text compression benchmark," **About the Test Data**. *http://mattmahoney.net/dc/textdata.html*. 2018. (10 Mayıs 2018).

Marinescu, Dan & Reinhold Kröger. *State of the art in autonomic computing and virtualization.* 2. Wiesbaden, Germany: Wiesbaden University of Applied Sciences, 2007.

McCalpin, John. "The STREAM Benchmark," **Virginia Edu**. *http://www.cs.virginia.edu/~mccalpin/papers/bandwidth/node2.html#SECTION00020000000000000000*. 1996. (12 Nisan 2018).

McCalpin, John. "The STREAM Benchmark," *http://www.cs.virginia.edu/~mccalpin/papers/bandwidth/node2.html#SECTION00020000000000000000*. 1996. (12 Nisan 2018).

McCalpin J. "The STREAM Benchmark.Philosophy and Justification," Virginia, *http://www.cs.virginia.edu/~mccalpin/papers/bandwidth/node3.html*. 1996. (23 Mayıs 2018).

Mell, Peter & Timothy Grance. "The NIST Definition of Cloud Computing," **NIST Special Publication**. 800-145: 1-3, 24 07 2011.

Menage , Paul. "www.kernel.org," *https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt*. 2006. (12 Haziran 2018).

Novy, Jindrich. "PXZ—parallel LZMA compressor using liblzma," **Fedora People**. *https://jnovy.fedorapeople.org/pxz/*. 2009. (23 Mart 2018).

Nuttcp. "Nuttcp Network Performance Measurement Tool," *https://www.nuttcp.net/Welcome%20Page.html*. 2016. (29 Nisan 2018).

Oguchi, Yoshihiko & Tetsu Yamamoto. "Server Virtualization Technology and Its Latest Trends.," **FUJITSU Science Tech Jounal**. 44: 46-52, 01 2008.

Oljira, Dejene Boru , Anna Brunstrom & Javid Taheri. "Analysis of Network Latency in Virtualized Environments.," *Global Communications Conference (GLOBECOM), 2016 IEEE*: Washington, DC, USA: 2016.

Rad, Babak Bashari, Harrison John Bhatti & Mohammad Ahmadi. "An Introduction to Docker and Analysis of its Performance.," *International Journal of Computer Science and Network Security*: 228-234. Kuala Lumpur: Asia Pacific University of Technology and Innovation Technology Park, Mart 2017.

Shea, Ryan , Feng Wang, Haiyang Wang & Jiangchuan Liu. "Deep investigation into network performance in virtual machine based cloud environments.," *IEEE INFOCOM 2014 - IEEE Conference on Computer Communication*: 1285-1292. Toronto: 2014.

Shujaa, Junaid, Abdullah Gania & Sajjad A. Madani. "A Qualitative Comparison of MPSoC Mobile and Embedded Virtualization Techniques," 2-3. 2016.

Smith, Zack. "Bandwidth: a memory bandwidth benchmark.," **ZSmith**. *https://zsmith.co/bandwidth.html*. 2017. (26 Nisan 2018).

Smith, J. , E & Ravi Nair. "The Architecture of Virtual Machines," **Computer**. ss. 32-38, 2005.

Soltesz, Stephen, Herbert Pötzl , Marc E. Fiuczynski, Andy Bavier & Larry Peterson. "Container-based operating system virtualization: A scalable, high-performance Alternative to Hypervisors," *EuroSys European Conference on Computer Systems 2007*: 275-287. Lisbon: 2007.

Stevens, Alan. "Theregister," *http://www.theregister.co.uk/2011/06/29/hybrid_cloud/*. 2011. (15 Ocak 2018).

Tong, Guan, Hai Jin, Xia Xie, Wenzhi Cao & Pingpeng Yuan. "Measuring and Analyzing CPU Overhead of Virtualization System," *IEEE Asia -Pacific Services Computing Conference*: 243-247. Jeju Island, South Korea: 2011.

UnixBench. "Google Code Arşiv," Google Inc, *https://code.google.com/archive/p/byte-unixbench/*. 2018. (24 Nisan 2018).

Vaquero, Luis M., Luis Rodero Merino, Juan Caceres & Maik Lindner. "A Break in the Clouds: Towards a Cloud Definition," **ACM SIGCOMM Computer Communication Review**. Volume 39, Number 1: 51-53, 1 Ocak 2009.

Vaughan-Nichols, Steven J. "New Approach to Virtualization Is a Lightweight," **Computer**. ss. 12-14, 20 Kasım 2006.

VmWare Inc. "Understanding Full Virtualization, Paravirtualization, and Hardware Assist," **Vmware Techpapers**. *https://www.vmware.com/techpapers/2007/understanding-full-virtualization-paravirtualizat-1008.html*. 2008. (22 Mayıs 2018).

VMware vSphere. "VMware vSphere 5.1 Documentation Center," Vmware Inc, *https://pubs.vmware.com/vsphere-51/index.jsp#com.vmware.vsphere.resmgmt.doc/GUID-69CDC049-8B42-4D26-8B47-94961B1777A4.html*. 2015. (11 Mayıs 2018).

"VMware vSphere 6.5 Documentation," VMware Inc, *https://pubs.vmware.com/vsphere-6-5/index.jsp*. 2017. (15 Mayıs 2018).

VMware White Papers. "TOP 4 CRITERIA TO CONSIDER WHEN SELECTING A VIRTUALIZATION PLATFORM," **insight.com**. *https://www.insight.com/content/dam/insight-web/en_US/pdfs/vmware/vmware-4-criteria-when-selecting-a-virtualization-platform.pdf*. 2018. (27 Ekim 2018).

Voorsluys, William, James Broberg & Rajkumar Buyya. "INTRODUCTION TO CLOUD COMPUTING," **Cloud Computing: Principles and Paradigms**: 1-43. Newyork: John Wiley & Sons, Inc, 2011.

Wang, Guohui & T. S. Eugene Ng. "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center.," *IEEE INFOCOM 2010.*: San Diego, CA, USA: 2010.

Xavier, Miguel G., Marcelo V. Neves, Fabio D. Rossi, Tiago C. Ferreto & Timoteo Lange. "Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments," *Parallel, Distributed and Network-Based Processing (PDP), 21st Euromicro International Conference*: 1-8. Belfast, UK: 2013.

Zeng, Hao, Baosheng Wang, Wenping Deng & Weiqi Zhang. "Measurement and Evaluation for Docker Container Networking.," *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017 International Conference IEEE*: 106-108. Nanjing, China: 2017.

Zhang , Qi, Lu Cheng & Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges," **The Brazilian Computer Society**. ss. 7-8, 8 Ocak 2010.