

T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SİTE GÜVENİRLİK MÜHENDİSLİĞİ YAKLAŞIMI KULLANILARAK YEREL BULUT İZLEME HİZMETLERİNİN İNCELENMESİ

YÜKSEK LİSANS

CANBERK KOÇ

Tez Danışmanı

DR. ÖĞR. ÜYESİ BORA UĞURLU

ÇANAKKALE – 2025



T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SİTE GÜVENİRLİK MÜHENDİSLİĞİ YAKLAŞIMI KULLANILARAK YEREL BULUT İZLEME HİZMETLERİNİN İNCELENMESİ

YÜKSEK LİSANS TEZİ

CANBERK KOÇ

Tez Danışmanı
DR. ÖĞR. ÜYESİ BORA UĞURLU

ÇANAKKALE – 2025

T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

Canberk KOÇ tarafından Dr. Öğr. Üyesi Bora UĞURLU yönetiminde hazırlanan ve **21/01/2025** tarihinde aşağıdaki jüri karşısında sunulan "**Site Güvenirlik Mühendisliği Yaklaşımı Kullanılarak Yerel Bulut İzleme Hizmetlerinin İncelenmesi**" başlıklı çalışma, Çanakkale Onsekiz Mart Üniversitesi Lisansüstü Eğitim Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri	**İmza**
Dr. Öğr. Üyesi Bora UĞURLU (Danışman)	……
Prof. Dr. Bahadır KARASULU	……
Dr. Öğr. Üyesi Murat Olcay ÖZCAN	……....

Tez No : 10696874
Tez Savunma Tarihi : 21/01/2025

……
Doç. Dr. Melis ULU DOĞRU
Enstitü Müdürü

../../20..

ETİK BEYAN

Çanakkale Onsekiz Mart Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Yazım Kuralları'na uygun olarak hazırladığım bu tez çalışmasında; tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi, tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu, tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu tezde sunduğum çalışmanın özgün olduğunu, bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi taahhüt ve beyan ederim.

Canberk KOÇ
21/01/2025

TEŞEKKÜR

Bu tezin gerçekleştirilmesinde, çalışmam boyunca benden yardımlarını esirgemeyen saygıdeğer danışman hocam Dr. Öğr. Üyesi Bora UĞURLU, çalışma süresince tüm zorlukları benimle göğüsleyen, hayatımın her evresinde bana destek olan değerli aileme ve dostlarıma sonsuz teşekkürlerimi sunarım.

Canberk KOÇ

Çanakkale, Ocak 2025

ÖZET

SİTE GÜVENİRLİK MÜHENDİSLİĞİ YAKLAŞIMI KULLANILARAK YEREL BULUT İZLEME HİZMETLERİNİN İNCELENMESİ

Canberk KOÇ
Çanakkale Onsekiz Mart Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi
Danışman: Dr. Öğr. Üyesi Bora UĞURLU

21/01/2025, 54

Günümüzün artan kaynak ihtiyacı ve gelişmiş yazılım ortamlarındaki uygulamaların taşınabilmesi adına ortaya çıkan dağıtık sistemler, zaman içerisinde endüstrinin bulut bilişime ilerlemesine yol açmış ve bu konuda oluşan ihtiyacı artırmıştır. Bu ihtiyacın karşılanmasında hem bulut bilişim sistemlerinin hem de üzerinde çalışan yazılımların sağlıklı ve güvenilir şekilde kesintisiz hizmet verebilmesi için Site Güvenirlik Mühendisliği alanı ortaya çıkmıştır. Yapılan çalışmada bu iki kavramın en önemli bileşenlerinden biri olan izleme sistemleri üzerinde durulmuştur. Bu sistemlerin günümüzde uluslararası endüstride en çok kullanılan 3 büyük bulut servisi sağlayıcısı olan Amazon Web Servisleri, Google Bulut Platformu ve Microsoft Azure'un aynı servis bölgesinde kurgulanması yolu izlenmiştir. Üç önemli bulut yerleştirim mimarisi olan servis olarak altyapı, servis olarak platform ve servis olarak yazılım mimarilerinden servis olarak platform mimarisinde örnek uygulama yerleştirilmiştir. Yöntem olarak en çok kullanılan mikroservis örüntülerinden olan Uygulama Programlama Arayüzü Geçidi ve Olay Güdümlü örüntüye sahip örnek yazılım bu servis sağlayıcıların sunduğu sanal makineler üzerine yüklenmiştir. Örnek yazılım Golang dili ile geliştirilmiştir. Veri tabanları için ilişkisel veri tabanı türü, olay güdümlü örüntü için ise bulut servis sağlayıcılarının sundukları mesajlaşma araçlar kullanılmıştır. Daha sonrasında bu yazılımların sağlığı, sağlayıcıların sunduğu izleme yöntemleri ve endüstride sıkça kullanılan izleme yöntemlerinin uygulanması ile izlenmiştir. İzlemenin durumu çeşitli nesnel ölçütler olarak; sağlanan gösterge sayısı, maliyet, uyarı ulaşma süresi, veri toplama frekansı ve gösterge saklama süresi üzerinden karşılaştırılmıştır. Bu karşılaştırmanın sonucunda elde edilen

veriler doğrultusunda, Site Güvenirlik Mühendisliği açısından önemli olan bu ölçütler üzerinden sağlayıcıların tercih edilebilirlikleri üzerine çıkarımlar yapılmıştır.

Anahtar Kelimeler: Bulut bilişim, İzleme sistemleri, Site güvenirlik mühendisliği, Yazılım mimarisi, Uyarı sistemleri

ABSTRACT

THE INVESTIGATION OF NATIVE CLOUD MONITORING SERVICES WITH THE USE OF THE SITE RELIABILITY ENGINEERING APPROACH

Canberk KOÇ
Çanakkale Onsekiz Mart University
School of Graduate Studies
Master of Science Thesis in Computer Engineering
Advisor: Asst. Prof. Dr. Bora UĞURLU
21/01/2025, 54

The increasing demand for resources in today's world and the need for applications to be portable across advanced software environments have led to the emergence of distributed systems. Over time, this has driven the industry toward cloud computing and further intensified the demand in this domain. In order to meet this demand, the field of Site Reliability Engineering has been created so that both cloud computing systems and the software running on them can provide uninterrupted service in a healthy and reliable manner. The study focused on monitoring systems, which are one of the most important components of these two concepts. The way to structure these systems in the same region of Amazon Web Services, Google Cloud Platform and Microsoft Azure, the three most used cloud service providers in the international industry today, was followed. An example application was made on Platform as a service from the three important cloud deployment architectures, Infrastructure as service, Platform as a service, Software as a service. As a method, the sample software with the most used microservice architectures, API Gateway and event-driven architecture, was deployed on the virtual machines offered by these service providers. The sample software was developed with the Golang language, for databases, the relational database model has been utilized, while messaging tools provided by cloud service providers have been employed for event-driven patterns. Afterwards, the health of these software systems has been monitored through the implementation of both the monitoring methods provided by cloud service providers and those commonly used in the industry. The effectiveness of monitoring has been evaluated based on objective criteria such as number of provided metrics, cost, alert delivery time, data collection frequency,

and indicator retention period. In the light of the data obtained as a result of this comparison, in terms of Site Reliability Engineering, conclusions have been drawn regarding the providers' preferability based on these key metrics.

Keywords: Cloud computing, Monitoring systems, Site reliability engineering, Software architecture, Alerting systems

İÇİNDEKİLER

Sayfa No

JÜRİ ONAY SAYFASI i
ETİK BEYAN ii
TEŞEKKÜR iii
ÖZET iv
ABSTRACT vi
İÇİNDEKİLER viii
SİMGELER ve KISALTMALAR x
TABLOLAR DİZİNİ xii
ŞEKİLLER DİZİNİ xiii

BİRİNCİ BÖLÜM
GİRİŞ 1

1.1 Bulut Bilişim 3
1.2 Mikroservis Mimarisi 5
1.3 Site Güvenirlik Mühendisliği 7

İKİNCİ BÖLÜM
ÖNCEKİ ÇALIŞMALAR 10

ÜÇÜNCÜ BÖLÜM
ARAŞTIRMA YÖNTEMİ 19

3.1. Uygulama Programlama Arayüzü Geçidi Örüntüsü 20
3.1.1 Kullanıcı Ekleme 21
3.1.2 Kullanıcı Sorgulama 23
3.1.3 Ödeme Ekleme 24
3.1.4 Ödeme Sorgulama 26
3.2. Olay Güdümlü Örüntü 27

3.2.1 Görsel İşleme 29
3.2.2 Görsel İşlem Sonucu Sorgulama 31
3.3. İzleme Uygulaması 33
3.4. Nesnel Değerlendirme ve Ölçütler 34
3.4.1 Sanal Makine Ölçütleri 34
3.4.2 Mesajlaşma Servisleri Ölçütleri 36
3.4.3 Veri Tabanı Servisleri Ölçütleri 37

DÖRDÜNCÜ BÖLÜM
ARAŞTIRMA BULGULARI 39

4.1. Uygulama Programlama Arayüzü Geçidi Örüntüsüne Ait Bulgular 39
4.1.1 Uç Noktalarındaki Yük Testi Raporları 39
4.1.2 Servislere Ait Gösterge Durumları 42
Amazon Web Servisleri 42
Google Bulut Platformu 43
Microsoft Azure 44
4.2. Olay Güdümlü Örüntüye Ait Bulgular 45
4.2.1 Uç Noktalarındaki Yük Testi Raporları 46
4.2.2 Servislere Ait Gösterge Durumları 48
Amazon Web Servisleri 48
Google Bulut Platformu 49
Microsoft Azure 49

BEŞİNCİ BÖLÜM
SONUÇ ve ÖNERİLER 51

KAYNAKÇA 55
EKLER I
EK-1 API GEÇİDİ ÖRÜNTÜSÜ 100 VE 250 İSTEK YÜK TESTİ SONUÇLAR II
EK-2 OLAY GÜDÜMLÜ ÖRÜNTÜ 100 VE 250 İSTEK YÜK TESTİ SONUÇLARI VI
EK-3 API GEÇİDİ ÖRÜNTÜSÜ GÖSTERGE PANELİ DURUMLARI VIII
EK-4 OLAY GÜDÜMLÜ ÖRÜNTÜ GÖSTERGE PANELİ DURUMLARI XI

SİMGELER VE KISALTMALAR

SRE	Site Güvenirlik Mühendisliği (Site Reliability Engineering)
API	Uygulama Programlama Arayüzü (Application Programming Interface)
IAAS	Servis Olarak Altyapı (Infrastructure as a Service)
PAAS	Servis olarak Platform (Platform as a Service)
SAAS	Servis olarak Yazılım (Software as a Service)
MS	Milisaniye
S	Saniye
MiB	Mebibayt
KiB	Kibibayt
SLA	Servis Düzeyi Anlaşmaları (Service Level Agreements)
SLO	Servis Düzeyi Hedefleri (Service Level Objectives)
SLI	Servis Düzeyi Göstergeleri (Service Level Indicators)
GCP	Google Bulut Platformu (Google Cloud Platform)
AWS	Amazon Web Servisleri (Amazon Web Services
SOA	Servis Odaklı Mimari (Service Oriented Architecture)
SOAP	Basit Nesne Erişim Protokolü (Simple Object Access Protocol)
HTTP	Hiper Metin Aktarım Protokolü (Hypertext Transfer Protocol)
CI/CD	Sürekli Tümleştirme ve Sürekli Dağıtım (Continuous Integration/Continuous Delivery)
REST	Gösterimsel Durum İletimi (Representational State Transfer)
TCP	İletim Kontrol Protokolü (Transmission Control Protocol)
AMQP	Gelişmiş Mesajlaşma Kuyruğu Protokolü (Advanced Message Queuing Protocol)
JSON	JavaScript Nesne Gösterimi (JavaScript Object Notation)
QOS	Servis Kalitesi (Quality of Service)
DEVOPS	Geliştirme ve İşletim Kavramı (Development Operations)
AIOPS	Bilişim Teknolojileri İşletimi için Yapay Zeka (Artificial Intelligence for Information Technology Operations)
OECD	Ekonomik Kalkınma ve İş Birliği Örgütü (The Organization for Economic Cooperation and Development)

NIST	Amerika Birleşik Devletleri Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology)

TABLOLAR DİZİNİ

Tablo No	Tablo Adı	Sayfa No
Tablo 1	Altyapı tercihleri	19
Tablo 2	API Geçidi sanal makinesi AWS ölçüt durumları	43
Tablo 3	API Geçidi sanal makinesi GCP ölçüt durumları	44
Tablo 4	API Geçidi sanal makinesi Microsoft Azure ölçüt durumları	45
Tablo 5	AWS mesajlaşma servisi ölçüt durumları	48
Tablo 6	GCP mesajlaşma servisi ölçüt durumları	49
Tablo 7	Microsoft Azure mesajlaşma servisi ölçüt durumları	50
Tablo 8	Literatürde ele alınan çalışmaların farklı boyutları	51
Tablo 9	Bulut servis sağlayıcılarının izleme araçlarının karşılaştırılması	52

ŞEKİLLER DİZİNİ

Şekil No	Şekil Adı	Sayfa No
Şekil 1	Bulut bilişim servis sunum modelleri	4
Şekil 2	API Geçidi örüntüsü mimari tasarım	21
Şekil 3	Kullanıcı ekleme ucu swagger görseli	22
Şekil 4	Kullanıcı ekleme sıralama diyagramı	23
Şekil 5	Kullanıcı sorgulama ucu swagger örneği	23
Şekil 6	Kullanıcı sorgulama sıralama diyagramı	24
Şekil 7	Ödeme ekleme ucu swagger örneği	25
Şekil 8	Ödeme ekleme sıralama diyagramı	25
Şekil 9	Ödeme sorgulama ucu swagger örneği	26
Şekil 10	Ödeme sorgulama sıralama diyagramı	27
Şekil 11	Olay Güdümlü örüntü mimari tasarım	28
Şekil 12	Görsel işleme ucu swagger örneği	29
Şekil 13	Görsel işleme isteği sıralama diyagramı	30
Şekil 14	Görsel işleme sıralama diyagramı	31
Şekil 15	Görsel işleme sonucu sorgulama ucu swagger örneği	32
Şekil 16	Görsel işleme sonuç kontrolü sıralama diyagramı	32
Şekil 17	Yük testi algoritma akış şeması	33
Şekil 18	Kullanıcı ekleme yük testi sonuç görseli	40
Şekil 19	Kullanıcı sorgulama yük testi sonuç görseli	40
Şekil 20	Ödeme ekleme yük testi sonuç görseli	41
Şekil 21	Ödeme sorgulama yük testi sonuç görseli	42
Şekil 22	Görsel işleme yük testi sonuç görseli	46
Şekil 23	Görsel işlem sonucu sorgulama yük testi sonuç görseli	47

BİRİNCİ BÖLÜM
GİRİŞ

Yazılım geliştirme kavramının insan hayatına girdiği ilk zamanlardan itibaren en büyük sorunlardan birisi kaynak ihtiyacı olmuştur. İçinde bulunduğumuz çağın gelişim hızı bu gereksinimin gün be gün artmasına sebebiyet vermektedir. Bu gereksinimin karşılanması için uzun yıllardır üzerinde çalışılan dağıtık sistemler, günümüzde dağıtık sistemlerin bir alt dalı olan bulut bilişim servisleri ile de bizlerin karşısına sıklıkla çıkmaktadır. Ekonomik Kalkınma ve İş Birliği Örgütü (EKİBÖ) (The Organization for Economic Cooperation and Development - OECD) çalışmalarında da görüldüğü üzere birçok ülke ve kurum günden güne bulut bilişim servisleri üzerindeki kullanımını artırmaktadır (Ker, 2021). Synergy Research Group'un çalışmasına göre 2023 yılında bu alandaki harcamalar 270 Milyar Dolara ulaşmış ve artmaya devam etmektedir.

Yapılan çalışmada bulut bilişim alanının öneminin günden güne artmasının yanı sıra, gelişen ve büyüyen bu sistemlerin üzerinde barındırılacak yazılım, donanım gibi katmanların da güvenirliği göz önüne alınmıştır. Bu sistemlerin güvenilir şekilde hizmet verebilmesinin önemi, bilişim çağının ilkel dönemleri diye tabir edilebilen 1990'lar ve öncesinde yapılan çalışmalarda güvenilirlik ile rekabet edebilirlik arasında bir bağlantı olduğu vurgulanmıştır (Musa, 1980; Musa ve Everett, 1990). Günümüzde güvenilirlik alanı Google tarafından ortaya atılan Site Güvenirlik Mühendisliği (SGM) (Site Reliability Engineering - SRE) kavramıyla yazılım çerçevesinde kendisine yer bulmaktadır.

Yukarıda bahsi geçen kavramların kesişim noktası olan yazılım geliştirme süreçleri zaman içerisinde tekil ortamlardan dağıtık ortamlara geçmiştir. Yapılan çalışmalarda görüldüğü üzere değişen ortamlarla birlikte gelişen yazılım mimarileri öncelikle Servis Odaklı Mimari'yi (SOM) (Service Oriented Architecture - SOA) sonrasında da birbirlerine zıt olmamakla beraber Mikroservis mimarisini gündeme getirmiştir (Niknejad vd., 2020). Günümüzde bulut ortamlarında en yaygın kullanılan mimarilerden olan mikroservislerin, çalışma için örnek seçilmesinin sebebi budur. Bahsi geçen mikroservis örüntülerinden birçok çalışmada sıkça adı geçen Uygulama Programlama Arayüzü (UPA) (Application Programming interface - API) Geçidi örüntüsünün ve Olay Güdümlü örüntünün seçilmesi bu nedenledir (Aksakallı vd. 2021; Jhingran ve Rakesh, 2023).

Örüntülerin seçiminin ardından güvenilirlik, yazılım geliştirme ve bulut servislerinin en önemli kesişim alanlarından biri olan izleme sistemleri seçilmiştir. İzleme ve tepki kavramları güvenilirlik alanının en önemli silahları olup, aynı zamanda da en büyük sorunlarından biridir. Doğru izlenmeyen herhangi bir sistem, ileride sorunlarla karşılaşmaya mahkum olmasının yanı sıra bu sorunların çözümü esnasında da büyük sıkıntılara sebebiyet verecektir.

Site güvenirliğinin temel taşlarından Servis Düzeyi Anlaşmaları (SDA) (Service Level Agreements - SLA), Servis Düzeyi Hedefleri (SDH) (Service Level Objectives - SLO), Servis Düzeyi Göstergeleri (SDG) (Service Level Indicators - SLI) kavramları izleme sistemlerinden elde edilecek ölçütlerle anlamlandırılır. Bu kavramlar bahsi geçen alanın temel taşlarıdır. Site güvenirliği kavramının doğru olarak kurgulanabilmesi ve alanın anahtar noktaları olan servis düzeylerinin doğru ölçülebilmesi adına izleme sistemlerinin farklı ana sağlayıcılarda karşılaştırılması çalışmanın temelini oluşturmaktadır. Bu yöntemin seçilmesinin amacı konunun bulut bilişim bağlamında öneminin farklı ölçütler altında incelenmesi, farklı ortam ve sistem davranışlarıyla nasıl irdelenebileceğinin gösterilmesidir.

Konunun incelenmesi için birçok örüntü arasından iki tanesinin seçilmesi, karşılaştırmalarının servis sağlayıcılarının sunduğu varsayılan sistemler üzerinden yapılması, tek kişilik bir çalışma olması ve servis kullanım maliyetleri kısıtlayıcı unsurlar olarak göz önünde bulundurulmalıdır.

Çalışmanın bu bölümden sonraki kısmı şu şekilde sunulmuştur: İkinci bölümde, kuramsal çerçeve ve önceki çalışmalar ele alınmaktadır. Üçüncü bölümde araştırma yöntemi ve materyaller ele alınmaktadır. Dördüncü bölümde araştırma bulguları ele alınmaktadır. Son olarak beşinci bölümde ise sonuç ve öneriler yer almaktadır.

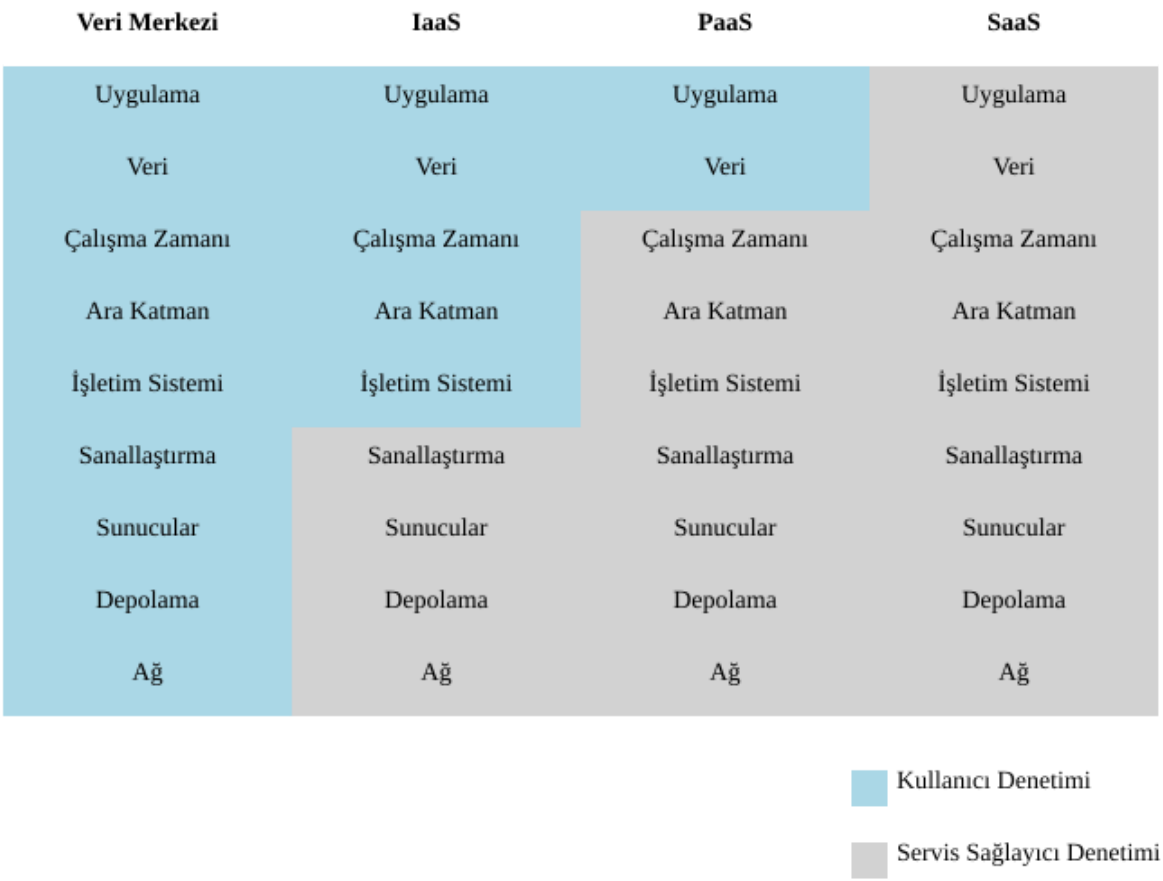
1.1. Bulut Bilişim

Yazılım geliştirme kavramının insan hayatına girdiği ilk zamanlardan itibaren gelişen kaynak ihtiyacı 1960'lı yıllarda dağıtık ve paylaşımlı sistemlerin gündeme gelmesine neden olmuştur. Bu sistemler zaman içerisinde gelişip, dallara ayrılarak 2006 yılında Amazon şirketinin bulut servisinin temel özelliklerini sağlamasıyla çağdaş bulut bilişim çağını başlatmıştır.

Bulut bilişim servislerinin sağlaması gereken temel özellikler Amerika Birleşik Devletleri Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) ve Avrupa komisyonuna göre şu şekilde sıralanmıştır (Mel ve Grance, 2011; European Commission, 2012):

- İsteğe göre ve öz hizmet: Kullanıcı kendi başına kendi istediği kaynaklardan yararlanabilmelidir.
- Geniş ağ ulaşımı: Servis sağlayıcısı çok yapımlı bir ortamda belirli yöntemlerle servislere ulaşılabilirlik sağlamalıdır.
- Kaynak paylaştırma: Bulut servis sağlayıcı çoklu kullanıcıya kullanıcının isteği doğrultusunda hizmet sağlayabilmek için dijital veya fiziksel kaynaklarını bir araya getirebilmeli bunları esnek bir şekilde dağıtabilmelidir.
- Hızlı esneklik: Servisler hızlıca ölçeklenebilir olmalıdır ve kullanıcıya saydam şekilde sürekli sorunsuz hizmet verebilmelidir.
- Ölçülmüş servisler: Sağlanan servisler sürekli olarak ölçülebilir olmalı gerekli durumlarda otomatik düzeltmeler sağlanabilmelidir. İzleme yeteneği ve ölçütler kullanılarak kaynakların kullanımı hem kullanıcı hem de sağlayıcıya açık ve net olarak paylaşılabilmelidir.

Bu özelliklerin yanı sıra servis yapıları da Şekil 1'deki gibi sıralanmışlardır;



Şekil 1. Bulut bilişim servis sunum modelleri.

Şekil 1'deki sınıflandırma günümüzde artan çeşitlerle birlikte XAAS sınıflandırması olarak geçmektedir. Başındaki X herhangi bir servisi temsil etmektedir. Şekil 1'de tanımlanan en çok kullanılan sunum yapıları şöyle açıklanabilmektedir:

- IAAS (Infrastructure as a Service): Servis olarak altyapı şeklinde geçmektedir. Ağ erişimi, sanal makineler gibi alt seviyeli servislerin sunulması manasına gelmektedir. Çalışmada kullanılan sanal makine servisleri bu yapıyı kullanmaktadır.
- PAAS (Platform as a Service): Servis olarak platform şeklinde geçmektedir. IAAS'ın bir üst seviyesi olarak aynı servislerin yanı sıra kullanıcının

sorumluluğundan işletim sistemi katmanlarını da çıkarmaktadır. Kullanılan izleme araçları bu yapıdaki servislerdendir.

- SAAS (Software as a Service): Servis olarak yazılım şeklinde geçmektedir. Kullanıcının tak çalıştır çözümler gibi sadece uygulamayı kullanma ile ilgilendiği en üst sanallaştırma seviyesine sahip yapıdadır.

Bahsi geçen NIST ve Avrupa Komisyonu gibi devlet kurumları çalışmalarının da yardımlarıyla beraber bulut bilişim kavramı daha düzenli ve daha geniş kitlelere hizmet edebilir hale gelmiştir. 2019 ve 2025 yılları arasında bulut bilişime geçişler ve tahminleri raporunda (Gartner, 2022) yer aldığı gibi, günümüzde milyarlarca dolar değere sahip hale gelen alanın ileride de önem kazanarak devam edeceği öngörülmektedir.

1.2. Mikroservis Yapısı

Kaynakların verimli kullanım ihtiyacı ve bu kaynaklara ulaşım kısıtları yazılım geliştirmenin temellerinin atıldığı zamanlardan itibaren mühendislerin üzerinde durduğu temel konulardan biri olmuştur. Bu ihtiyaç ve kısıtlar, teknolojideki ilerleyiş ve sistemlerin daha fazla kullanıcıya ulaşması ile birlikte, donanım ve yazılım kavramlarının gelişim sürecinin hızlanmasına yol açmıştır. 2000'li yılların başında monolitik sistemlerin taşınması, sorunlarının giderilmesi ve servislerin güncellenmesinde kesintilere sebebiyet vermesi gibi karşılaşılan zorluklara karşın SOA mimarisi alanda kabul görmeye başlamıştır.

Bu mimarinin en önemli noktası, monolitik bir sistemin sahip olduğu servislerin kullanımı açısından değerlendirilerek birden çok alt sisteme bölünmesidir. Bahsi geçen alt sistemlerin üstlendikleri sorumlulukların farklı olması, sistemin belirli bir bölümünün cevap veremez hale gelmesi durumunda bütün sistemin çökmemesine ve daha güvenilir bir yapıya kavuşmasına olanak sağlamaktadır. Mimari yaklaşımın bir servisin sahip olması gereken temel özellikler olarak adlandırdıkları şunlardır:

- Hizmet Odaklılık: Yazılan sistemler sunacağı işin etrafında özelleşmelidir.

- İşbirliktelik: Sistemler birbiriyle tektipleştirilmiş protokoller üzerinden iletişim kurmalı ve bu kullanılan dil, ortam gibi değişkenlere bağlı olmamalıdır.
- Gevşek Bağlantı: Sistemler diğer sistemlere en az bağlı olarak kurgulanmalıdır.
- Tekrar Kullanılabilirlik: Sistemler birimsel olarak kurgulanmalı, çoklu halde ve farklı uygulamalarda kullanılabilmelidir.
- Soyutlama: Sistemler iç mantıklarını dışarı sunmadan sadece arayüzlerle iletişim kurmalıdır.

Bu mimarinin işbirliktelik özelliği ile yazılım geliştirme çerçevesine eklediği bir başka kavram da Basit Nesne Erişim Protokolünün (Simple Object Access Protocol - SOAP) oluşmasıdır. Bu protokol bahsi geçen servisler arasındaki iletişimi tanımlayan protokollerden öncü olarak kabul görenlerindendir.

2010'lu yıllarda SOA yaklaşımını geliştirme ve daha dağıtık hale getirme ihtiyacı duyan endüstri Mikroservis mimarisini ortaya atmıştır. Amazon ve Netflix gibi büyük şirketlerin bu mimariyi yoğun kullanımı daha sonrasında da konteynırlaştırma kavramının hayatımıza girmesiyle mikroservisler yazılım geliştirmede temel yaklaşımlardan biri haline gelmiştir. Mikroservislerin temel özellikleri çoğunlukla SOA mimarisiyle örtüşmektedir fakat mikroservis mimarisini temellendiren Martin Fowler'a göre SOA daha çok bir yaklaşım olarak kalması ve içinde bir çok belirsizlik barındırması nedeniyle geliştirilmesi gereken bir kavramdı. Yine kendisine göre SOA'nın getirdiği temel özelliklere ek olarak tanımladığı özellikler şunlardır;

- Dağıtık veri: Her parça kendi verisinden sorumlu olmalıdır.
- Bağımsız konuşlandırma: Parçalar ayrık şekilde konuşlandırılabilir ve güncellenebilir olmalıdır.
- İzlenebilirlik: Parçalar kütüklenebilir, ölçütleri toplanabilir ve takip edilebilir olmalıdır.
- Dayanıklılık: Parçalar hata karşıtı ve hatalardan sonra düzeltilebilir olarak kurgulanmalıdır.

- Hafif iletişim**:** Parçalar aralarında düşük veri kullanan Gösterimsel Durum İletimi (GDİ) (Representational State Transfer- REST) gibi protokoller ile iletişim kurmalıdırlar.
- Altyapı Otomasyonu: Parçalar Sürekli Tümleştirme ve Sürekli Dağıtım (ST / SD) (Continuous Integration / Continuous Delivery - CI/CD) kavramları yazılım yaşam döngüsü prensipleri üzerine hızlıca konuşlandırılabilecek şekilde kurgulanmalıdır.
- Küçük Takımlar: Servisler bütün bir bölüm tarafından değil küçük takımlar tarafından yönetilebilir olmalıdır.

Bu gelişmeler mimarinin günümüzün çağdaş yazılım geliştirme yaklaşımlarına ve Devops kavramına da uyum sağlaması nedeniyle geniş bir alanda kullanılmasına yol açmıştır. Yoğun kullanımın getirdiği gelişmelerle beraber servisler arası iletişim protokolleri de gelişmiş ve SOAP gibi taşınan veri nedeniyle ağır olarak tabir edilen protokoller yerine daha hafif protokoller ortaya çıkmıştır. Bu protokollerden biri olan, çalışmada da servislerin arasında kullanılan REST protokolü Roy Fielding tarafından paylaşılmıştır (Fielding, 2000). REST protokolü ile oluşturulan uç noktalarının gösterimi için Swagger ismi verilen uygulama geliştirme arayüzü, görselleştirme ve belgelendirme aracı kullanılmıştır.

1.3 Site Güvenirlik Mühendisliği

Güvenilirlik hayatın her alanında hayati öneme sahip bir konu olmasından dolayı birçok endüstride üzerinde uzunca zamandır çalışmalar yapılan bir alandır. Yazılım geliştirme ve altyapıları açısından ürünlerin oluşturulması kadar bu ürünlerin güvenilir şekilde son kullanıcıya ulaştırılması da çok büyük öneme sahiptir. Bir ürüne yüksek ulaşılabilirlik ve güvenilirlikle ulaşılamadığı durumlarda bu ürüne karşı bağlılığın yitirildiği ve bunun da endüstriler adına büyük kayıplara sebebiyet verdiği araştırmalarda gösterilmiştir (Gefen, 2002). 2000'li yılların başında Google'ın güvenilirlik kavramını kendi hizmetleri açısından önemli olarak değerlendirmesi ile bu alan üstünde çalışan ekiplerini SRE ünvanıyla isimlendirmiştir. Alanın ortaya çıkmasında büyük rolü olan Ben Treynor Sloss'un da tanımladığı gibi, alan kısaca Devops felsefesinin gerçek hayatta uygulanması, yazılım mühendisliğine ağırlık verilerek, otomasyon ve eniyileştirme

yöntemleri ile operasyonel iş yükünün karşılanması olarak özetlenebilir (Google, t.y.).

Alanın temel amacı çok geniş ve karmaşık dağıtık sistemlere sahip şirketlerin bu sistemlerin kendilerini ve üzerinde çalışan uygulamalarının izlenebilirliği, güvenirliği ve başarımını artırmaktır. Bilişim teknolojilerinin her alanında hizmet sunabilen kavramın kullanıldığı yerlere bulut bilişim, e-ticaret, mobil uygulamalar, yayıncılık servisleri, bankacılık uygulamaları örnek olarak verilebilir. Bu alandaki temel kavramları dört ana başlık altında toplamak mümkündür;

Servis düzeyleri: Başarım ve güvenilirlik düzeylerinin belirlenmesi için kullanılan servis düzeyleri kavramları temellerin en başında gelmektedir. Bu kavram kendi içerisinde birçok alt kavram ile ilişkilidir. Bunlar sırasıyla Servis Düzeyi Göstergeleri (SLI), Servis Düzeyi Anlaşmaları (SLA) ve Servis Düzeyi Hedefi'dir (SLO).

SLI, bir sistemin veya servisin başarımı ve kalitesini ölçmek için kullanılan nicel ölçütlerdir. Bu gösterge, bir servisin ne kadar iyi performans gösterdiğini veya ne kadar kullanılabilir olduğunu belirlemektedir.

SLA, bir servis sağlayıcı ile müşteri arasında yapılan ve sunulan servisin kalitesini, performansını ve diğer ilgili kriterlerini tanımlayan bir anlaşmadır. Bu anlaşmalar SRE takımları ve onların çalıştığı diğer takımlar arasında yapılmalıdır. Örneğin "Sistemin yıl boyunca %90 ayakta kalması" gibidir. Bunlar bağlayıcı anlaşmalar olarak görülmelidir ve SRE takımları bu anlaşmaların altında kaldıklarında acil önlemler almalıdır.

SLO, bir sistemin veya servisin performansı için belirlenen ölçülebilir hedeftir. SLI'lar üzerinden yapılan bu hedefler SLA'nın üstünde olmalıdır ve tutturulmaya çalışılması bütün sistemin güvenirliğini artırmak adına önem arz etmektedir.

Arıza bütçeleri: Sistemlerin yenilenmesi ve güncellenmesi sırasında oluşabilecek kesintileri öngörmek adına arıza bütçeleri kavramı kullanılmaktadır. Bu kavram servis

düzeyleri kavramıyla uyumlu olmalıdır.

Otomasyon: SRE ekipleri sistemlerin konumlandırılmasında, güncellenmesinde ve arızalardan geri dönülmesinde otomasyonu sağlamayı esas hedeflerinden görmektedirler. Otomasyon, özellikle büyük ölçekli ve karmaşık sistemlerde insan müdahalesini en aza indirerek süreçlerin daha hızlı, güvenilir ve sürekli bir şekilde gerçekleşmesini sağlar.

Servis düzeyi göstergelerinin belirlenmesi ve toplanması: İzleme kavramı, SRE kavramı için SLI'ların belirlenmesi toplanması ve diğer servis düzeylerinin ölçülmesi adına büyük öneme sahiptir. Aynı zamanda önetkin izleme çalışmaları ile arızalar ortaya çıkmadan çıkarımlar yapmak güvenirliği ileriye taşımak için esas yöntemlerden biridir.

Yukarıda bahsedilen temel kavramların yanı sıra, alanın temel esaslarını belirlemek için yapılan çalışmalardan alınan derslerin doğrultusunda kullanılan yöntemlerden bazıları; güncel uygulamaların oluşturulup kullanılması, olay müdahalesi ve olay sonrası belgeleri oluşturup olay anlarında hızlı ve verimli hareket edilmesi, düzenli yük testlerinin yapılması, ölçeklenebilirliğin sağlanması, izleme sistemlerinin kurgulanması ve alınan verilere göre hareket planlarının oluşturulması şeklinde sıralanabilir.

İKİNCİ BÖLÜM
ÖNCEKİ ÇALIŞMALAR

Çalışmanın bu alanında çalışma yönteminin ve gerekli verilerin belirlenmesinde kullanılan kaynaklar incelenmiş ve bu kaynakların tezin konusuyla kesişimleri açısından değerlendirilmiştir.

İncelenen araştırma, dersler ve akademik dökümanlarda işlenen konular ile endüstride uygulanan yöntemler arasında farklılıklar olduğunu göstermektedir. Bu teoriden yola çıkılarak yapılan akademik çalışmaların gerçek hayatta karşılığının artması için daha fazla akademik kaynağa ve eğitime ihtiyaç duyulduğu kanısı paylaşılmıştır (Dickerson ve Chen, 2023).

Bulut bilişim ortamlarında yerleştirme olarak en çok kullanılan araçlardan biri olan Kubernetes üzerine yapılan araştırma sonucunda izleme alanında yapılan akademik çalışmalarda eksiklikler görülmüştür. Çalışmanın yapıldığı çerçevede sadece %1,9 oranında izleme alanına ilgi gösteren çalışmaların diğer çalışmalara oranı bu alanda bir açığı göstermektedir (Shamim vd., 2022).

Endüstrinin önde gelen Amazon, Netflix, Google gibi şirketleri dahil olmak üzere, son yıllarda mikroservis mimarisi kullanımı büyük bir alana yayılmıştır. Bununla beraber mikroservislerin ayakta kalmaları ve oluşturulmaları, birçok soru ve sorunu yanında getirmiştir. Bunun yanı sıra endüstri ve akademi arasında da uyumsuzluklar oluşmuştur. Bu sorunların tanımı ve uyumsuzlukların giderilmesi adına yapılan çalışmada güvenlik, mimari, izleme, yönetim vb. birçok alanda sorunlardan ve çözümlerin arandığından bahsedilmiştir (Soldani vd., 2018).

AT&T çalışanı olan yazarlar tarafından yapılan çalışmada kendi sistemleri üzerinde çalışan uygulamaların bulut servis sağlayıcılarına taşınmasında yaşanan zorluklar uygulama başarım izlemesi özelinde incelenmiştir. Bu zorluklar beş soru ile ele alınmıştır. Yapılan çalışma açısından en önemli iki soru kendi sistemleri ve bulut sistemler üzerindeki izleme araçları arasındaki farklılıkların ne olduğu ve iki ortam arasındaki başarım

izlemenin hangi etkenlere dayalı olduğudur. İlk soruya farklı ağ yapıları ve kaynak alt yapılarına göre izleme çözümlerinin araştırılması gerektiği, ikinci soruya ise farklılıkların maliyet ve işlevselliği açısından incelenmesi gerekliliği şeklinde cevap verilmiştir (Debroy vd., 2020).

Çalışmada bulut izleme araçları üzerinde düzenli bir sınıflandırma yaptıklarını belirtmektedirler. Bu araçların doğru kurgulanmasının bulut üzerinde çalışan uygulamalar için elzem olduğu kanısını vurgulamışlardır. Bir çok uygulama üzerinde çalışarak servis kalitesinin yükseltilmesi açısından izleme araçlarının önemi ve bu araçların ihtiyaç duyduğu öznitelikler araştırılmıştır. İncelenen çalışmanın bu çalışma açısından önemi, incelediği bir çok uygulama içinde bu çalışmanın da incelemeyi hedeflediği sistemlerin genel yapısı hakkında bilgiler vermesidir. Karşılaştırmaları yaparken ele alınması gereken öznitelikler hakkında fikirler sunması da önemli bir başka noktadır (Fatema vd., 2014).

Kaynakta araştırmacılar deneysel bir metodoloji ile bulut bilişimde başarım ölçümü ve bunun tekrarlı şekilde yapılabilmesi adına sekiz ilkeli bir yöntem ortaya koymuşlardır. Ortaya konan yöntem ile izlenen alanda ölçütlerin belirlenmesi ve bu ölçütlerin tekrarlanabilirliği gerektiğine yaklaşımları, başarım ölçümü adına önem arz etmektedir. Deneyler esnasında kullandıkları bulut bilişim araçları ile araçların servis düzeyi anlaşmalarının oluşturulması ve bu araçların maliyet açısından incelenmesi tez konusuna dayanak oluşturması açısından önem arz etmektedir (Papadopoulos vd., 2021).

İncelenen çalışmada araştırmacılar Amazon web servisleri ve Google bulut platformu bulut servis sağlayıcıları üzerinden bir başarım karşılaştırması yapmışlardır. Bu karşılaştırma için bir yük testi aracı ve bir uygulama programlama arayüzü servisi kullanmışlardır. Kaynak tez açısından Go dilinin neden kullanılması gerektiği üzerine ve izlemenin bu sağlayıcılar üzerinde nerelerde yapılması gerektiği adına bir öneme sahiptir (Noviani vd., 2022).

Kaynakta araştırmacılar birçok dilin bulut performanslarını kıyaslamışlardır. Python, PHP, Node.js ve Go karşılaştırılan diller olmakla beraber aynı veri tabanı ve kullanıcı arayüzü kurgusu altında bulut üzerinde yapılan çalışmalarda Golang dilinin daha

iyi sonuç verdiği kanısını paylaşmışlardır. İncelenen kaynak tez açısından dil seçiminde ve örnek kurguların kurulmasında önem taşımaktadır (Tanadechopon ve Kasemsontitum, 2023).

Çalışmada araştırmacılar inceleme yapılan zaman aralığında bulut izleme yöntemlerinin durumunu sorgulamışlardır. Bu sorgulamada o dönem içerisinde uygulamalar için ölçünleme (standardizasyon), endüstride izlemenin öneminin hak ettiği değeri görememesi ve endüstri içinde yeteri kadar uygulamanın bulunmaması gibi sorunların altını çizmişlerdir. Bu eksikliklerin güvenilirlik açısından istenilen seviyelere ulaşılamaması, servis kesintileri gibi sıkıntılara sebebiyet verdiği kanısına varılmıştır. Kaynakla ortak alanlarda incelemelerde bulunan bu çalışma açısından izlenen yöntem, ortak araçlar ve alanda bulunan araştırma talebi önem arz etmektedir (Tamburri vd., 2020).

Kaynakta araştırmacılar bulutta çalışan uygulamalar için genel geçer en iyi yöntemleri belirlemek için bir çalışma yapmışlardır. Bu çalışmada geniş bir literatür üzerinde çalışan araştırmacılar bahsi geçen yöntemleri sınıflandırmaya gayret etmişlerdir. Araştırma sonuçlarında yazılım geliştirme için zaman, kapsam ve iletişimin en büyük öneme sahip olduğu kanısına ulaşmışlardır. Tez açısından çalışmada bahsi geçen kavramlar ve bahsi geçen literatür araştırmasının getirdiği mimariler, yapılan örnek yazılımın mimarisi seçiminde önem arz etmiştir (Akbar vd., 2020).

Araştırmacılar bulut izleme mimarileri arasında bir karşılaştırma yapmışlardır. Bu karşılaştırmada mimarileri sınıflandıran çalışmacılar ortaya çıkan sonuçlar üzerinde başarım, ölçeklenebilirlik, kaynak kullanımı ve güvenlik özellikleri arasında karşılaştırmalarda bulunmuşlardır. Bu karşılaştırmalar sırasında bulut bilişimin servis olarak altyapı yöntemi üzerinde duran çalışmacılar başarım için farklı, güvenlik ve yalnızlaştırma için farklı mimariler önermişlerdir. Kaynağın tez açısından en önemli noktaları bulut izleme sistemlerinin nasıl çalıştığının incelemesi, bu mimariler üzerinde izleme çalışmaları yapması ve karşılaştırmalarda nelere önem verilmesi gerektiği hakkındaki paylaştığı bilgilerdir (Calero ve Aguado, 2015).

Kaynakta çalışmacılar bulut üzerinde servis olarak altyapı modeliyle sunulan servislerin izlenmesi için birkaç aracı karşılaştırmışlardır. Bu araştırmada kullandıkları araçları servis düzeyi anlaşmaları adına kullanmak için karşılaştırmaya sokmuşlardır. Karşılaştırma ölçütleri ulaşılabilirliğin, başarımın, disk yazma ve okumalarının gösterimi olarak alınmıştır. Bu çalışmanın bu tez için önemi ise kullanılan ortak araçların karşılaştırılması ve bu karşılaştırımın nasıl yapıldığı hakkında paylaşılan bilgilerdir (Absa vd., 2019).

Çalışmada araştırmacılar bulut ortamları izlemek için kullanılan uygulamaları incelemiş ve bu incelemeyi bir sınıflandırmaya dönüştürerek çıkarımlar yapmaya çalışmışlardır. Bu sınıflandırmayı mimari, kullanım alanı, kökeni, iletişim yöntemleri ve veri toplama yöntemleri altında yapmışlardır. Mimarileri tek katmanlı, çok katmanlı ve merkeziyetsiz; kullanım alanlarını ölçüt toplama, kütük, olay inceleme, sistem sağlığı izleme ve ağ izleme; kökenleri büyük işletmeler, küme hesaplama, bulut bilişim ve dağıtım hesaplama; iletişim yöntemlerini tekli yayın, çoklu yayın ve ara katmanlı yayın; veri toplama yöntemlerini ise karşıya yollayan ve karşıdan çeken şeklinde ele almışlardır. Bu inceleme geliştirilen araçların günden güne karmaşıklaşan bulut ortamlarına uyum sağlamak adına daha dağıtık ve karşıya veri yollayan yöntemlere doğru yöneldiğini göstermektedir. Ayrıca çalışma, bu karmaşık ortamlarda çalışmak adına ve ortamda olan her durumun izlenmesi zorunluluğu sebebiyle Site Güvenirlik Mühendisliği gibi rollere oluşan ihtiyacın da altını çizmektedir. Araştırmacılar araştırmanın sonucunda gelecekte bu ortamlarda çok katmanlı ve heterojen yapıların kullanılacağını tek bir tak çalıştır çözümün olmasının olası olmadığı kanısını paylaşmıştır. Yapılan çalışma açısından incelenen kaynak çok geniş bir literatür taradığı ve bahsi geçen alanlarda fikir vermesi açısından önem taşımaktadır (Ward ve Barker, 2014).

Çalışmada Site Güvenirlik Mühendisliği'nin üretim süreçleri üzerindeki etkisi üzerine çalışma gerçekleştirilmiştir. Bu araştırma esnasında Site Güvenirlik Mühendisliği alanının otomasyon, olay müdahale planlaması, servis düzeyi hedeflerinin belirlenmesi, sürekli gelişim gibi alanlarının üretim süreçlerinin iyileştirilmesi adına kullanımı üzerinde durulmuştur. Site Güvenirlik Mühendisliği takımlarının kullandığı yöntemlerden olan otomasyon ve genelleştirilmiş sürekli görevleri kullanarak karmaşık ve girift üretim

yapılarında iyileştirmelere gidilmesinin bu takımların olay müdahale planları, olay sonrası raporlarının oluşturulması gibi diğer görevleri adına zaman kazandırdığı belirtilmiştir. Bunların yanı sıra suçlamasız bir inceleme kültürü oluşturması ve adı geçen görevlerin daha düzgün hale getirilmesi süreçlerdeki kesintileri ve bu kesintilerin üretime etkisini en aza indirgeyeceği kanısı paylaşılmıştır. Araştırma Site Güvenirlik Mühendisliği izleme, zorluğu azaltma, riski en aza indirgeme ve otomasyon ilkelerini bir örnekle sunmuştur. Bu örnekte oluşan bir olayın takibi ve olaya müdahale önceliklendirmesi gibi yöntemler kullanarak arıza süresini en aza indirmeyi örneklendirmektedir. Araştırma Site Güvenirlik Mühendisliği kavramının günden günde daha fazla benimsenmesi yapay zekâ araçlarının kullanımı ve Bilişim Teknolojileri İşletimi için Yapay Zeka (BTİYZ) (Artificial Intelligence for IT Operations - AIOps) kavramının literatüre girmesi gibi bu alanın gelişim eğilimlerini incelemiştir ve bu yönelimin verimliliği artırarak, maliyetleri aşağı çekerek ve müşteri deneyimini iyileştirerek alanın gelecekte etkisinin artmasına katkıda bulunacağını belirtmektedir. Hatta alanın sanal dönüşüm için daha çevik, sağlam ve daha kaliteli servislerin sunulması açısından bir devrim yapma ihtimalinden bahsetmişlerdir. Tez açısından da Site Güvenirlik Mühendisliği alanının sahip olduğu önem ve gelecekte olası önemi, temel araştırma konularından birisi olması açısı ile incelenen çalışma önem taşımaktadır (Dave, 2023).

Çalışmada araştırmacılar izlemenin her gün gelişim gösteren bulut bilişim kavramı üzerindeki önemi üzerine durmuşlardır. Kavramın gelişim gösterme sebepleri olarak enerji kullanım verimliliği, sağladığı teknik kolaylıklar, kaynakların verimli kullanımı ve düşen maliyetler vurgusunda bulunmuşlardır. Araştırma izleme yöntemlerinin bulut bilişim üzerinde etkili, gerekli yönlerine eğilmiş ve kaynak hacim planlaması, kaynak yönetimi, veri merkezi kontrolleri, başarım eniyilemesi, maliyet takibi ve güvenlik gibi yönleri bu sınıflandırma içine almıştır. Araştırmacılar bulut izleme yöntemleri için katman seviyelerini, sanallaştırma seviyelerini, ölçüt ve testleri kullanarak bir sınıflandırma oluşturmuşlardır. Bu sınıflandırmanın ardından bir bulut izleme yönteminin sahip olması gereken ölçeklenebilirlik, esneklik, uyarlanabilirlik, zaman bağımsızlık, otonomluk, doğruluk, erişilebilirlik ve sağlamlık özelliklerinin temel özellikler olarak altını çizmişlerdir. Bu tanımlamalar ışığında araştırmacılar bahsi geçen kavramların sağlanmasında karşılaşılan zorluklar ve çözümlerin literatürdeki karşılıkları ve eksikliklerini incelemişlerdir. Büyük izleme verilerinin bulut üzerinde incelenmesi gibi

sorunlar üzerinde duran araştırmacılar birden çok araç üzerinde önerilerde bulunmuşlardır. Araştırmada ismi geçen CloudWatch, AzureWatch gibi bulut servislerinin kendi uygulamaları yanı sıra Nagios, OpenNebula gibi üçüncü kaynak diye tabir edilen uygulamalar da incelenmiştir. Alanda daha gelişmiş kök neden araştırmaları yapılması, daha sağlam ve dayanıklı sistemlerin kurulması, yeni yaklaşımlarla sorun bazlı çözümler üretilmesi, katmanlar arası ve servis sağlayıcıları arası geçişken yöntemler üretilmesi, enerji ve maliyet için verimli sistemler oluşturulması gibi birçok anahtar alanda araştırmaların yapılmasının gelecekte bulut bilişimin gelişmesine büyük katkı sağlayacağını belirtmişlerdir. Bu bahsi geçen kavramların hepsinin temelinde olan izleme kavramının öneminin günümüzde ve gelecekte artarak devam edeceği kanısını paylaşmışlardır. İncelenen çalışmanın, tezin belkemiği olan izleme hakkında açık alanlardan bahsi ve önerileri tez açısından önem arz etmektedir (Aceto vd., 2013).

Kaynakta bulut izlemeye uygun olabilecek üçüncü kaynak araçları servis olarak altyapı kavramı özelinde incelenmiştir. Araçları, bulut izlemenin temel kavramları olan izleme amacı, yerleşim mimarisi, veri toplama yöntemleri ve bu sistemlerin birbirleri ile karşılaştırılması şeklinde ele almışlardır. Araştırmanın esas eğilimi bu araçların büyük izleme verisi hacimleri üzerindeki davranışları, izlemenin projelere getireceği kaynak yükü, ayarlanabilirlik ve ölçeklenebilirlik üzerine kurgulanmıştır. Araçları mimarisel olarak dağıtık, tekil, basamaklı ve veri toplama yöntemleri olarak etmenli, etmensiz gibi kırılımlar ile sınıflandırmışlardır. Nagios, Zabbix gibi ismi açık kaynak izleme camiasında çokça geçen birçok uygulamanın yetileri, farklı bulut mimarilerindeki davranışları, güçlü ve zayıf yönlerini uygulamanın yazıldığı dil, yazılımcı arayüzü sağlaması, ölçeklenebilirliği, güvenlik, veri tutma ihtiyacı gibi birçok alanda karşılaştırmaya tabii tutmuşlardır. Karşılaştırmaların sonunda izleme kavramının derinliği ve her ihtiyaca farklı yaklaşımlar yapılması gerektiği aynı zamanda izlemenin doğru sağlanması için doğru araçları seçmenin de çok büyük öneme sahip olduğu kanısını paylaşmışlardır. Bunların yanı sıra araştırmacılar incelenen araçların incelenen zaman aralığında fiziksel, sanal ve uygulama katmanlarında eksiklere sahip olduğunu ve bu eksiklerin giderilmesi adına bu araçlar üzerine yeni çalışmalar yapılması konusunda da tavsiyelerde bulunmuşlardır. Bu çalışmanın tez özelinde önemi, ulaştıkları sonuca binaen kullanılan bulut sağlayıcılarına özel izleme uygulamalarının konumlandırılma şeklini desteklemesidir (Grati vd., 2015).

Çalışmada üç önder bulut servis sağlayıcısı olan Amazon web servisleri, Google bulut platformu ve Microsoft Azure sistemlerini karşılaştırmışlardır. Karşılaştırmayı yaparken büyüyen bulut bilişim pazarında doğru servis sağlayıcının seçiminde artan seçenekler ve zorluklar üzerinde durulmuştur. Bu seçimleri bazı sağlayıcıların hesaplama gücüne bazılarının maliyet azaltmaya ve bazılarının ise güvenilirlik veya gelişmiş ölçeklenebilirliğe odaklanması yönüyle ayrıştırmaya çalışmışlardır. Çalışma sunum modelleri, yerleşim yöntemleri, esneklik, ulaşılabilirlik, ölçeklenebilirlik, güvenilirlik, sanallaştırma ve otonomluk gibi bulut bilişimin temellerinden bahsederek seçimlerin dayanağını oluşturmaktadır. Bahsi geçen temellerde yapılan alt kırılımlı ve birçok karşılaştırmanın yanı sıra sağlayıcıların sunduğu farklı depolama sunumları, yapay zekâ makine öğrenmesi çözümleri gibi kullanıcıların hedeflerine katkı sağlayabilecek yönleri de önemli ölçütler olarak belirlemişlerdir. Her platformun kendi içinde güçlü yönleri olduğu ve kullanıcıların kendi ihtiyaçları doğrultusunda doğru servis sağlayıcısını seçmesinin büyük önem taşıdığı kanısını sunmuşlardır. Tez açısından kullanılan servis sağlayıcıların birçok yönden ele alınması ve izleme özelinde hangi ölçütlere önem verilmesi gerektiği üzerine paylaşılan fikirler önem arz etmektedir (Saraswat ve Tripathi, 2020).

Çalışmada araştırmacılar bulut bilişimde izleme araçları ve bu araçların etkili, kaynak dostu ve servis düzeylerine uygun kurgulanması üzerinde durmuşlardır. Çalışmanın ana hattını bir bulut izleme aracının yazılım katmanları ve altyapısal kurgusunun servis kalitesi, kaynak kullanımı, uygulama başarımı gibi alanlarda incelenmesi ile çıkarımlarda bulunması oluşturmaktadır. Bulut kavramlarının nasıl algılanacağı ve işleneceği, izlemenin boyutları ve bu boyutlara uygun uygulamaların oluşturulması, bu alanların gelişmesi üzerine katkılar sunulmuştur. Karşılaştırdıkları araçları üçüncü taraf veya buluta özel olarak sınıflandırmayan araştırmacılar, bu araçları izleme mimarisi, bulutta barındırma şekli, diğer araçlarla işbirliktelik, servis kalitesi matrisleri, yaşam döngüleri, servis düzeyi anlaşmaları ve görünürlük üzerinde karşılaştırmışlardır. Karşılaştırmalarının sonucunda var olan araştırmalarında desteğiyle, alanda daha fazla araştırma yapılması gerektiği kanısını vurgulamıştır. Bu kanının yanı sıra bu alan üzerinde yapılan çalışmaların devamında gelişen bulut ortamlarının daha güvenilir ve güvenli hale gelebileceği öneriminde bulunmuşlardır. Tez ile ortak alanları bulunduran bu çalışma mimari ve karşılaştırmalı değerlendirmelerde odak noktaları üstünde fikir vermesi dolayısıyla önem taşımaktadır (Kumar vd., 2017).

Açık kaynaklı ve ticari birçok izleme aracının incelendiği çalışmada araştırmacılar bulut bilişim özelinde bu araçların sınıflandırılması, karşılaştırmasını yapmışlardır. Araştırmacılar bu tür araçların bulut bilişim ortamları gibi hızlı ve karmaşık ortamların verimli ve yüksek başarımla çalışması açısından esas etkenlerden biri olduğu fikri üzerinde durmuşlardır. Bahsi geçen yazılımları lisans türlerinin yanı sıra izleme uygulamalarının temel özellikleri olarak belirledikleri ulaşılabilirlik, uyarlanabilirlik, ölçeklenebilirlik, esneklik, yalnız çalışabilirlik, sağlamlık, güvenilirlik gibi birçok ölçütle karşılaştırmaya tabi tutmuşlardır. Ayrıca sınıflandırma yapmaya gayret eden araştırmacılar bulut servisi tiplerini kamuya açık, kapalı ve karma; servis modellerini altyapısal, servis bazlı ve ortam bazlı; veri toplama yöntemlerini etmenli ve etmensiz; ölçüt özelliklerini geniş veya dar olarak almışlardır. Araçların başarımlarını da bu özelliklere göre ölçmeye gayret etmişlerdir. Araştırmacılar Datadog, New Relic gibi üçüncü kaynak sistemlerin yanı sıra Amazon Cloudwatch, Microsoft Cloud Monitor gibi buluta özel sistemleri ele alarak her birinin doğru amaç doğrultusunda kullanımının bulut güvenirliği, etkinliği için katkı sağlayacağı kanısını paylaşmışlardır. Bu araçların incelenmesi ve sonuçların nasıl karşılaştırılması hakkında fikir sahibi olunması açısından incelenen araştırma tez açısından önem taşımaktadır (Birje ve Bulla, 2020).

İzleme kavramının güncel bilgi teknolojileri üzerindeki etkisinin özellikle dağıtık sistemlerin en büyük örneklerinden bulut bilişimin ilerlemesi ile daha elzem hale geldiği fikriyle yola çıkılarak sunulan çalışmada araştırmacılar bu fikri bazı izleme araçlarının ve bu araçların alt yapısında kullanılan protokolleri inceleyerek desteklemişlerdir. SNMP, IPMI, WMI gibi protokollerin güçlü ve zayıf yönlerini Zabbix, Nagios ve Prometheus gibi araçların mimarileri üzerinde farklı durumlarda incelemişlerdir. Bu incelemeler sonucunda her aracın kullanıcının karşılaştığı zorluğa göre öne çıkan yönleri olduğu kanısını paylaşmışlardır. Bu ihtiyaçların ortak noktası olarak geniş ve dağıtık ortamlarda ölçeklenebilir, otonom, görselleştirilebilen ve hareket noktası sağlayabilen araçlar olduğu fikrini öne sürmüşlerdir. Bir başka bakış açısı olarak uyarı sistemlerinin bu izleme sistemlerine bileşik çalışmasının da önemi vurgulanarak bu tür kurguların güvenilirlik, etkinlik ve hatalara hızlı karşılık verilmesinde kolaylıklar sağlayacağından bahsedilmiştir. Bahsi geçen araçların hepsinin yeterli olduğunu belirtmelerinin yanı sıra sonuç olarak günden güne ilerleyen teknolojinin izleme alanında ihtiyacı artırdığı ve seçimler yapılırken karşılaşılan araçların değerlendirilmesinin durumlara özel olduğunu savunmuşlardır. Tez

açısından bu araştırmanın da diğer araştırmalar gibi karşılaştırmaların sadece araçlar arasındaki farklılıkların değil durumsal, mimarisel ve ortamsal farklılıkların da karşılaştırılması gerekliliği fikrini pekiştirmesi önem arz etmektedir (Kontsek vd. 2020).

ÜÇÜNCÜ BÖLÜM
ARAŞTIRMA YÖNTEMİ

Bu tez çalışmasında, literatürde var olan iki farklı mikroservis örüntüsü olan API Geçidi örüntüsü ve Olay Odaklı örüntü; üç farklı bulut bilişim sağlayıcısı (AWS, GCP, Azure) üzerinde oluşturulmuştur. Bu örüntüler kullanılan yük testi aracı ve betikler yardımıyla yük altında kalmaya zorlanmışlardır. Örüntülerin üç farklı sağlayıcıda yük altındayken oluşturdukları ölçütler site güvenirliği kapsamında incelenmiş ve tez çalışmasının sonraki bölümünde ise yorumlanmıştır.

Aşağıdaki tabloda seçilen mimarilerin gerçekleştirimlerinin yapıldığı üç farklı sağlayıcıdaki hizmet bölgeleri, kullanılan izleme araçları, mesajlaşmada kullandıkları servisler, kullanılan veri tabanları, sanal makine türleri ve sanal ağ türleri verilmiştir.

Tablo 1

Altyapı tercihleri

Bulut Servis Sağlayıcısı	Amazon (AWS)	Google (GCP)	Microsoft Azure
Hizmet Bölgesi	Frankfurt	Frankfurt	Frankfurt
İzleme Aracı	CloudWatch	GCP Monitoring	Azure Monitor
Mesajlaşma	SQS	PUB/SUB	Service Bus
Veri Tabanı	RDS for Postgresql	Cloud SQL for Postgresql	SQL Database for Postgresql
Sanal Makine	EC2	GCE	Azure VM
Sanal Ağ	AWS VPC	GCP VPC	Azure Virtual Network
İşletim Sistemleri	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS
İşlemci Sayısı	1 Sanal İşlemci	1 Sanal İşlemci	1 Sanal İşlemci
Bellek Boyutu	2 Gigabayt	2 Gigabayt	2 Gigabayt

Bu servislerin kullanımı ile birlikte altyapıların konum olarak nerede yerleşim sağladığı da sistemlerin kurgulanmasında büyük öneme sahiptir. Çünkü karşılaştırmaların

yapıldığı bölgeler arasında oluşacak gecikmeler sonuçları anlamsız hale getirebilmektedir. Bunu önlemek adına sağlayıcıların ülkemize en yakın ortak hizmet bölgesi olan Frankfurt seçilmiştir.

Kaynaklar üzerinde yapılan incelemelerde görüldüğü üzere genişlemeye ve karmaşıklaşmaya devam eden bu ortamların güvenilir, sağlam ve ulaşılabilir olması adına en önemli kavramlardan biri izleme olmaktadır. Bu çalışmada incelenecek Amazon Web Servisleri (AWS), Google Bulut Platformu (GCP) ve Microsoft Azure sistemlerinin kendilerine özel sağladığı Servis Olarak Platform (PAAS) izleme çözümleri olan CloudWatch, GCP Monitoring, Azure Monitor'ün kullanılmasının sebebi de servis sağlayıcılara ait çözümlerin birbirleriyle karşılaştırılarak, okuyuculara servis sağlayıcısı seçimlerinde veri sağlamaya çalışmaktır.

Bulut servis sağlayıcıları üzerinden kullanılan veri tabanı, mesajlaşma servisleri gibi servislerin tasarımsal olarak tutarlılığı ve kararlılığı bulut servis sağlayıcıları tarafından karşılanmaktadır. Deneylerde kullanılan web adresleri X.X.X.X şeklinde örnek olarak gösterilmektedir. Kullanılan veri tabanı tabloları deneylerin başında boş olarak gelmekte sonrasında veriler ile doldurularak deneyler yapılmaktadır.

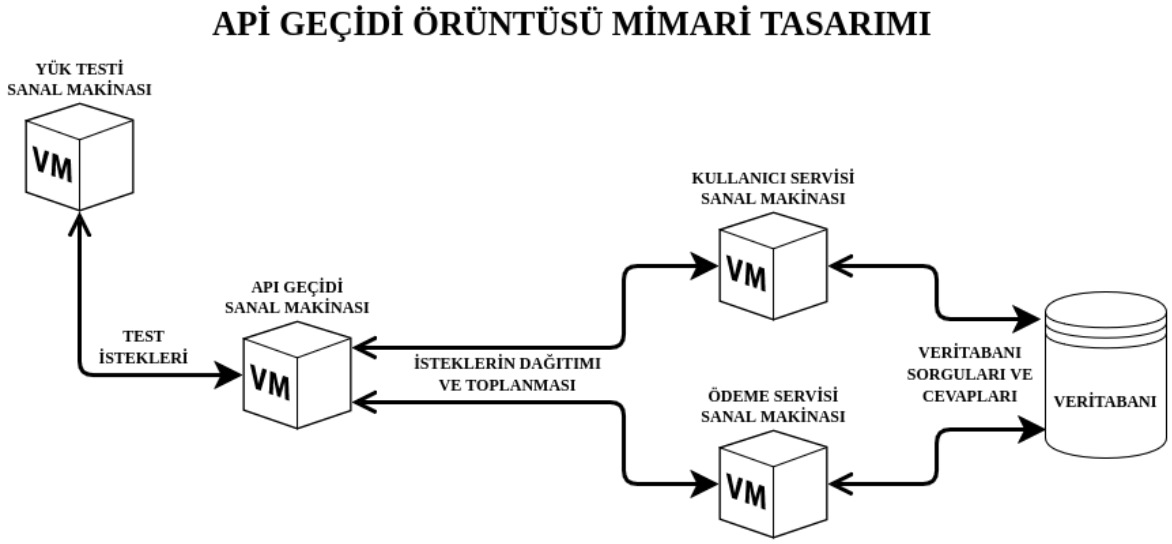
3.1 . Uygulama Programlama Arayüzü Geçidi Örüntüsü

API Geçidi, istemciler (mobil uygulamalar, web tarayıcıları vb.) ile arka uç servisleri (mikroservisler, sunucusuz işlevler veya veri tabanları) arasında bir köprü görevi gören bir ara katmandır. Çağdaş bulut tabanlı uygulamalarda, özellikle mikroservis mimarilerinde ve sunucusuz çözümlerde API çağrılarının merkezi bir noktadan yönetilmesi ve güvence altına alınması için kullanılır. Bu mimari hem performans hem de güvenlik açısından kazanımlar sunar.

Bir API Geçidi, istemcilerden gelen istekleri karşılar ve uygun şekilde yönlendirir. Örneğin, gelen API çağrılarını analiz ederek doğru mikroservise iletir, ardından mikroservisten gelen yanıtı işleyerek istemciye döner. Bu süreçte, istek ve yanıtları format açısından dönüştürme gibi görevleri de yerine getirir. Bu işlevsellik, API Geçidinin

sistemdeki iletişim yükünü azaltmasını sağlar.

Şekil 2'de seçilen ve her üç servis sağlayıcı üzerinde gerçeklenen API Geçidi tasarımının mimari görünümü yer almaktadır.



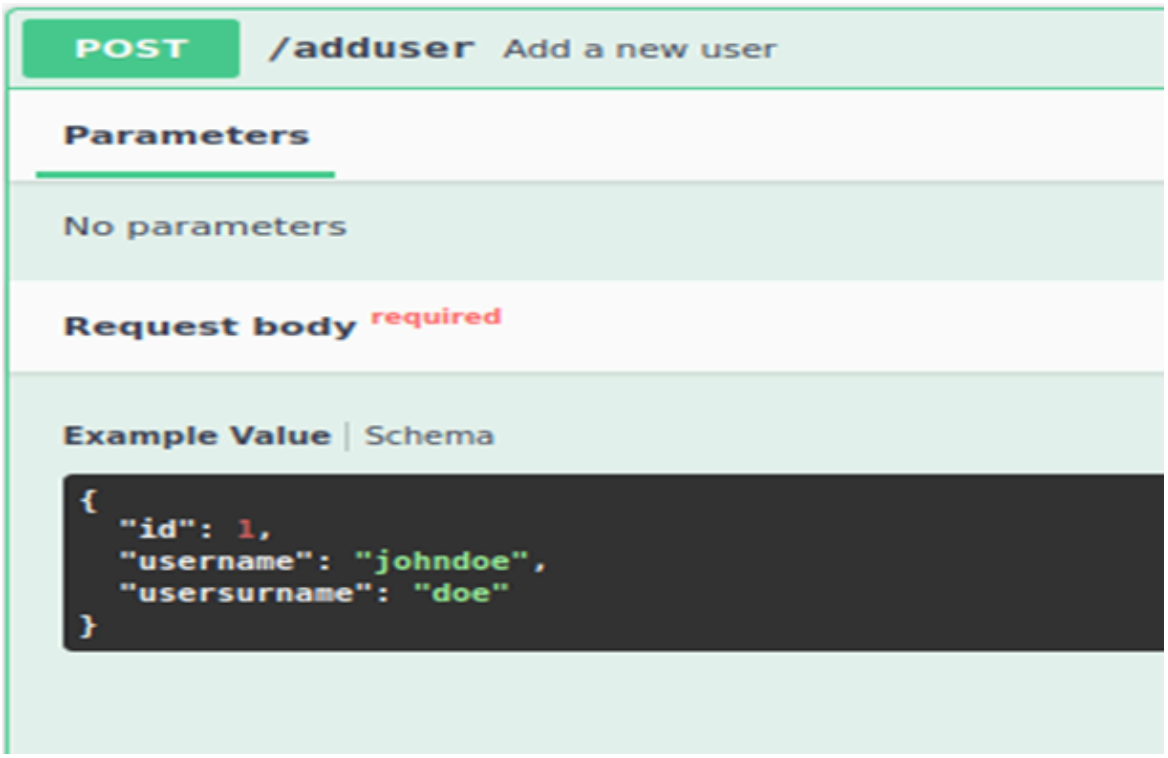
Şekil 2. API Geçidi Örüntüsü Mimari tasarım.

Şekil 2'de ifade edilen ilk örüntü olan API Geçidi için REST iletişimi kullanılmıştır. Bu protokol için tanımlanan uç noktalar OpenAPI standardı ile Swagger aracı kullanılarak görselleştirilmiştir. Tasarımda yer alan API Geçidi servisi gelen istekleri iki başlık altında işlemektedir. İlk istek türü, yeni kullanıcıların veri tabanına eklenmesi ve/veya daha önceden eklenen kullanıcıların veri tabanından getirilmesidir. Diğeri ise kullanıcıların yaptığı ödemelerin benzer şekilde veri tabanına eklenmesi ve/veya ödemelerin sorgulanarak getirilmesidir.

3.1.1. Kullanıcı Ekleme

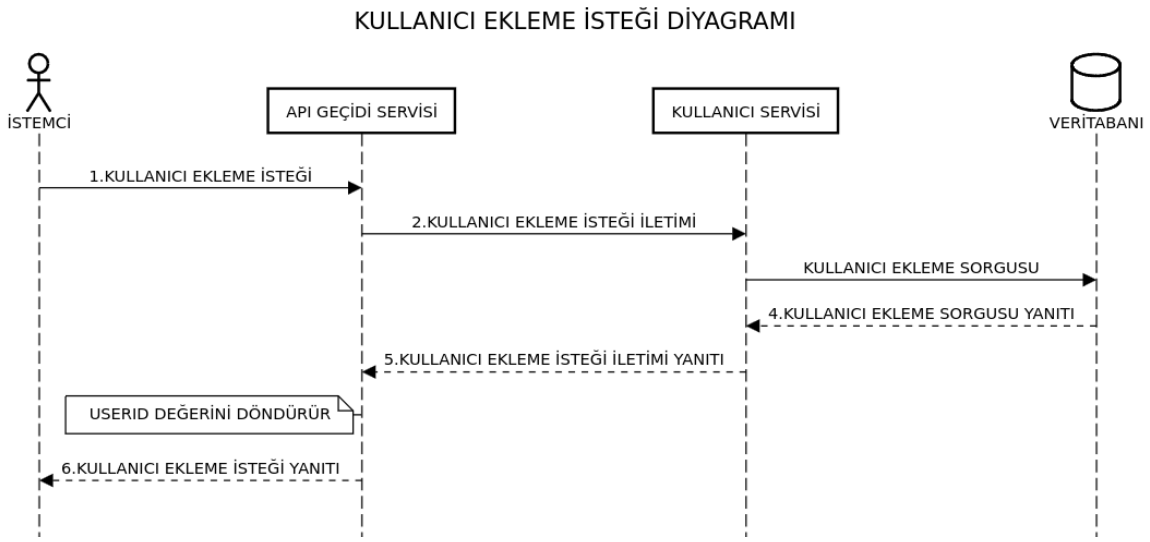
Sisteme yeni bir kullanıcı eklemek istendiğinde API Geçidi servisine bir Hiper Metin Aktarma Protokolü (HMAP) (Hyper Text Transfer Protocol - HTTP) POST isteği gönderilir. Gönderilen bu istek JavaScript Nesne Gösterimi (JNG) (Javascript Object Notation - JSON) formatında olup "http://X.X.X.X:8080/adduser" örnek adresine

gönderilmiştir. İstek Şekil 3'de görselleştirilmiştir.



Şekil 3. Kullanıcı ekleme ucu swagger görseli.

Şekil 4'te açıklandığı üzere istek API Geçidi birimine geldikten sonra kullanıcı birimine aktarılır, kullanıcı birimi gelen veriler ile veri tabanına sorgu yapar, kullanıcı birimi sorgunun cevabının olumlu olması durumunda veri tabanından gelen cevabı API Geçidine iletir, API Geçidi servisi de kullanıcının id'sini sorguyu yapan kullanıcıya geri döndürür.



Şekil 4. Kullanıcı ekleme sıralama diyagramı.

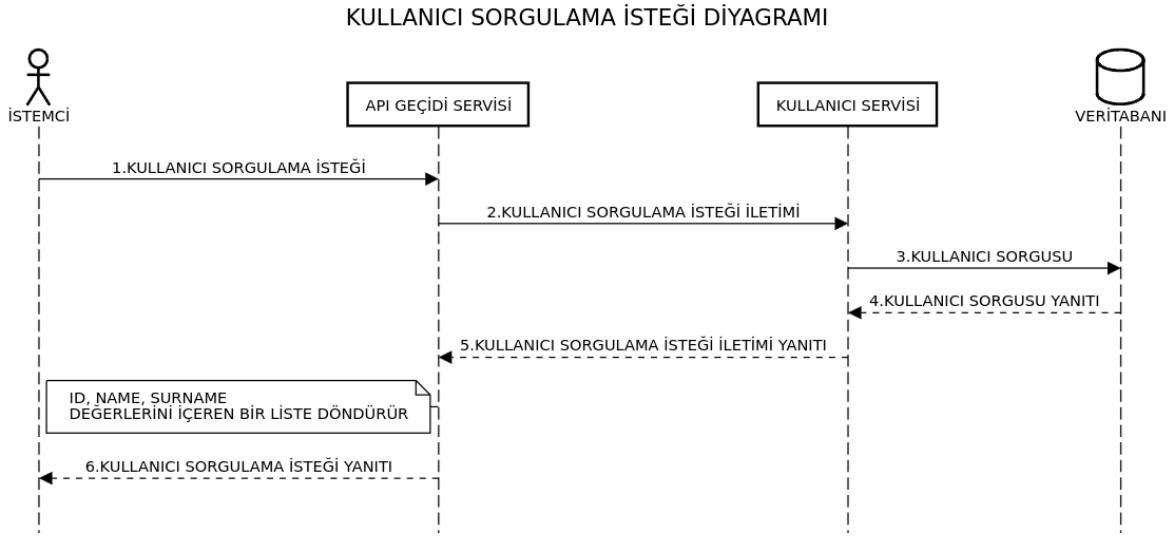
3.1.2. Kullanıcı Sorgulama

Veri tabanında var olan bir kullanıcıyı getirmek istediğinde API Geçidi servisine bir http GET isteği gönderilir. İsteğin "name" sorgulama değişkenine kullanıcının adı eklenir. İstek "http://X.X.X.X:8080/getuser?name=test" örnek adresine yapılmış olup Şekil 5'te görselleştirilmiştir.



Şekil 5. Kullanıcı sorgulama ucu swagger örneği.

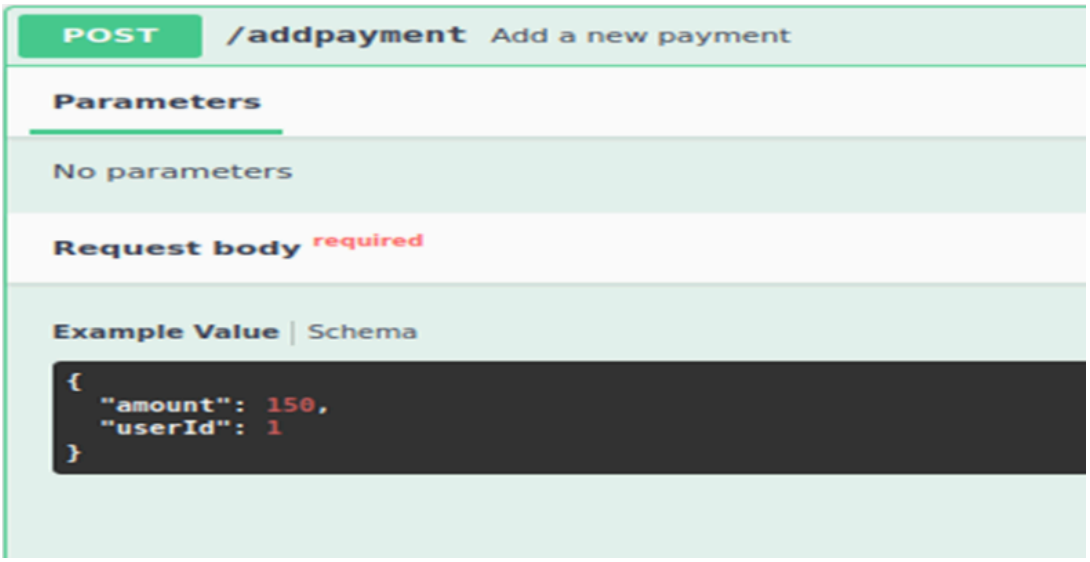
Şekil 6'da açıklandığı üzere, sorgu API Geçidi birimine geldikten sonra kullanıcı birimine aktarılır. Ardından kullanıcı birimi veri tabanına gelen isim değişkeni ile sorgu yapar, kullanıcı birimi sorgunun cevabının olumlu olması durumunda veri tabanından gelen cevabı API Geçidi birimine, API Geçidi birimi de kullanıcının idsi, ismi ve soyismi olan bir listeyi sorguyu yapan kullanıcıya geri döndürür.



Şekil 6. Kullanıcı sorgulama sıralama diyagramı.

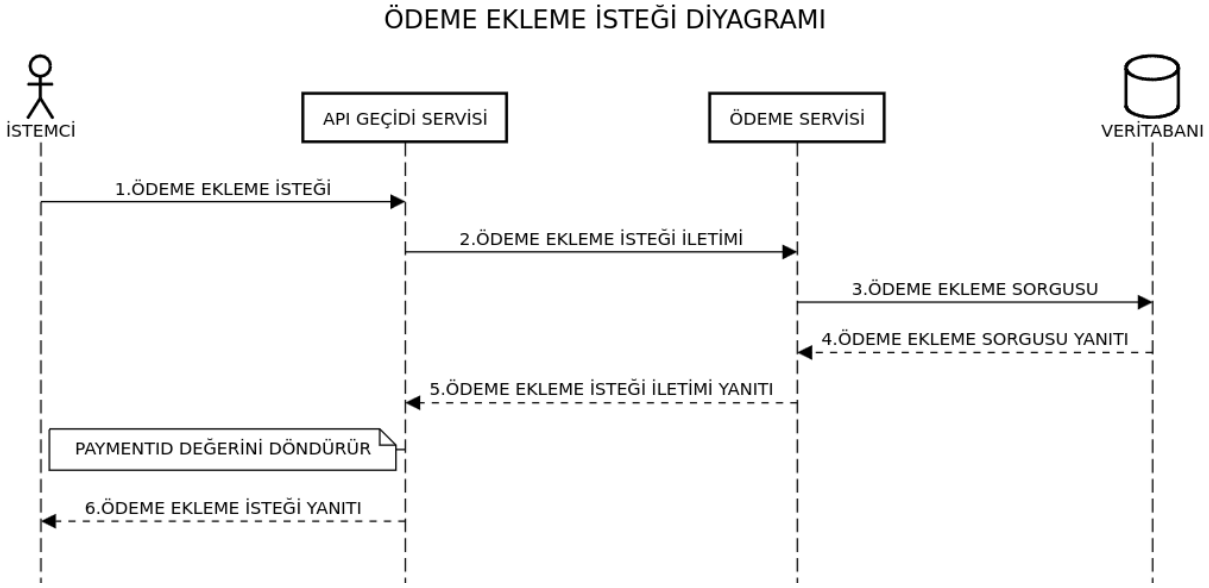
3.1.3. Ödeme Ekleme

API Geçidi servisi yeni kullanıcıların yanı sıra yapılan ödeme işlerini de ele almaktadır. Bu bağlamda kullanıcı veri tabanına yeni bir ödeme eklemek istenildiğinde "http://X.X.X.X:8080/addpayment" örnek adresinden yapacağı istekle sisteme yeni bir ödeme kaydı oluşturabilir. Yapılacak ödeme JSON formatında tanımlanmış olup görünümü Şekil 7'de sunulmuştur.



Şekil 7. Ödeme ekleme ucu swagger örneği.

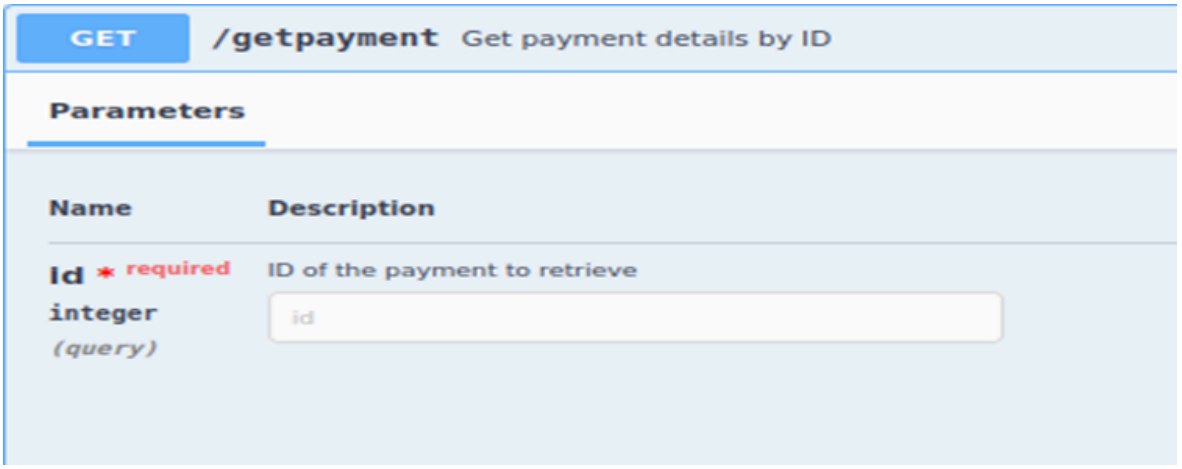
Şekil 8'de açıklandığı üzere istek, API Geçidi birimine geldikten sonra ödeme birimine aktarılır. Ödeme birimi veri tabanına gelen veriler ile sorgu yapar. Ödeme birimi sorgunun cevabının olumlu olması durumunda veri tabanından gelen cevabı API Geçidine iletir. API Geçidi servisi de ödemenin id'sini sorguyu yapan kullanıcıya geri döndürür.



Şekil 8. Ödeme ekleme sıralama diyagramı.

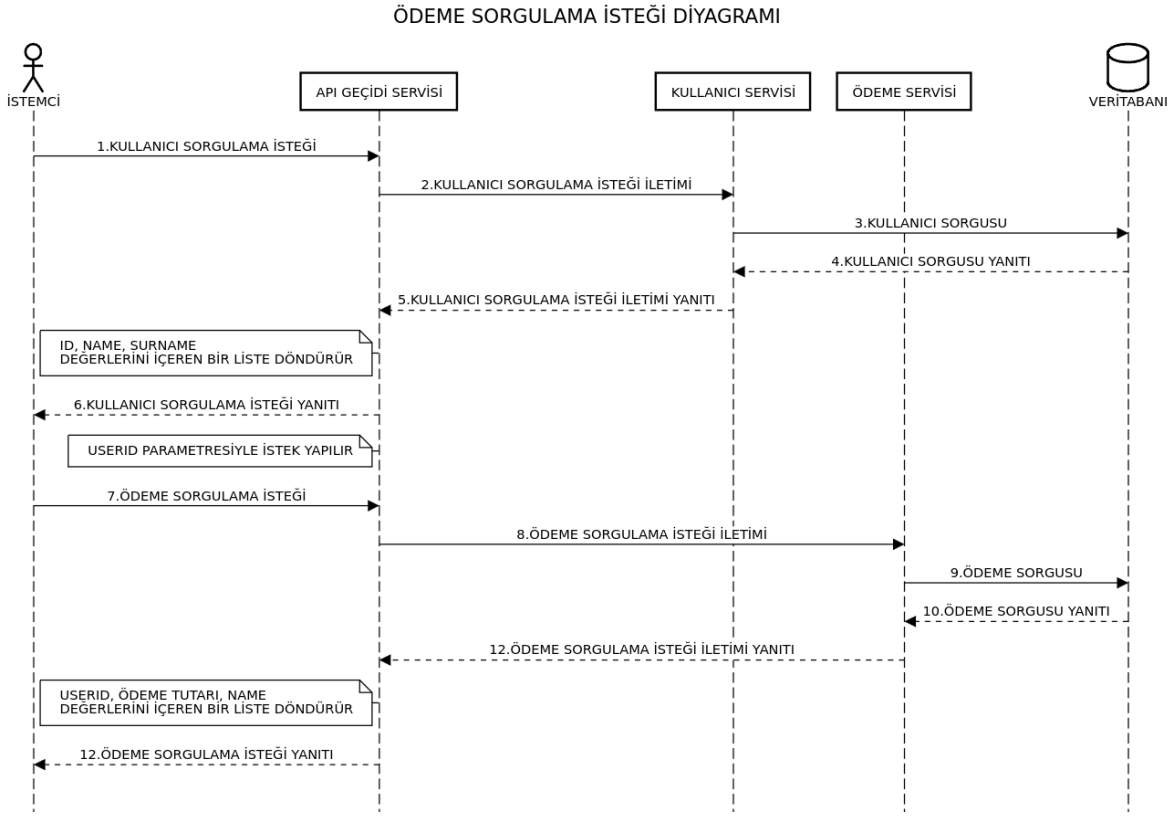
3.1.4. Ödeme Sorgulama

İşlem kullanıcının API Geçidi birimine Şekil 5'te açıklanan "getuser" sorgusunun sonucunu almasıyla başlar. Sonrasında kullanıcı kullanıcı id'sini kullanarak API Geçidi birimine "getpayment" ucundan Şekil 9'da Swagger görseli olan GET isteğini yapmasıyla devam eder. İsteğin "id" sorgulama değişkenine kullanıcının adı eklenir. İsteğin yapıldığı örnek adres "http://X.X.X.X:8080/getpayment?id=111" 'dir.



Şekil 9. Ödeme sorgulama ucu swagger örneği.

Sorgu Şekil 10'da gösterildiği gibi API Geçidi birimine geldikten sonra ödeme birimine aktarılır, ödeme birimi veri tabanına gelen id değişkeni ile sorgu yapar, ödeme birimi sorgunun cevabının olumlu olması durumunda veri tabanından gelen cevabı API Geçidine iletir, API Geçidi servisi de kullanıcı id'sini, ödemenin tutarını ve kullanıcın ismini bir liste halinde sorguyu yapan kullanıcıya geri döndürür.



Şekil 10. Ödeme sorgulama sıralama diyagramı.

3.2. Olay Güdümlü Örüntü

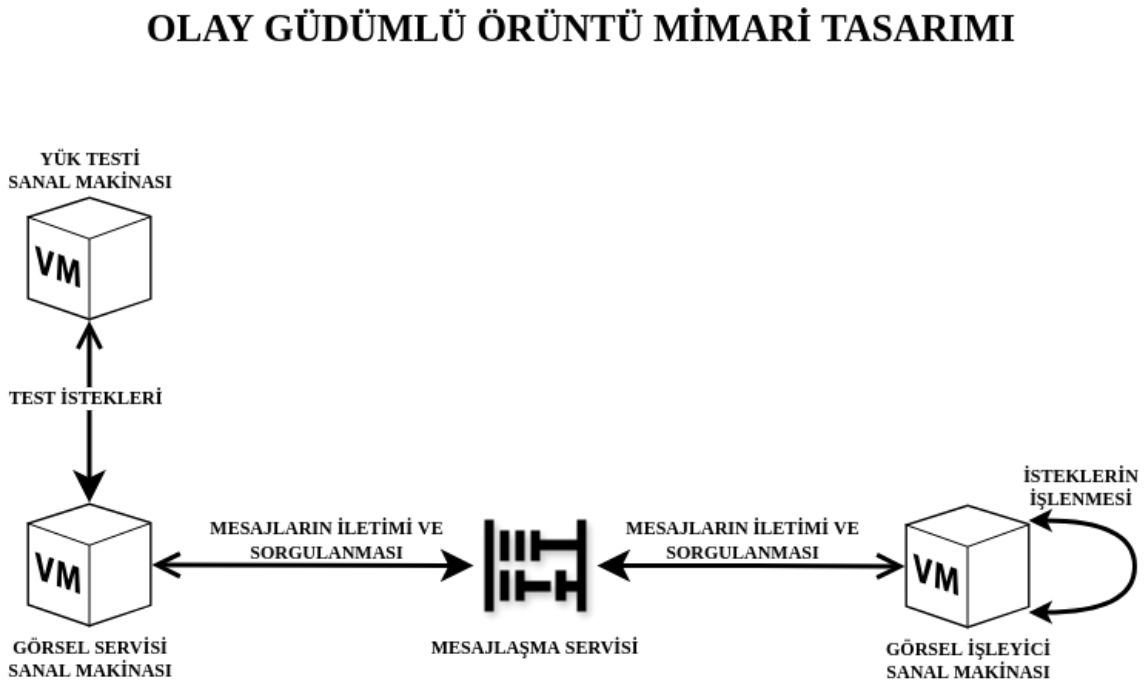
Olay Güdümlü örüntü bir sistemdeki bileşenlerin olaylara tepki vererek çalıştığı bir mikroservis örüntüsüdür. Bu örüntü, olayların (events) sistemdeki diğer bileşenler veya uygulamalar tarafından tetiklenmesi ve bu olaylara göre işlem yapılması üzerine kuruludur. Bu mimaride dört farklı kavram yer almaktadır. Bunlar sırasıyla olay (event), üretici (producer), tüketici (consumer) ve olay yöneticisidir (event broker).

Olay, sistem içinde meydana gelen bir durumu veya değişikliği ifade eder. Örneğin, bir kullanıcı kayıt olduğunda, bir dosya yüklendiğinde veya bir ödeme tamamlandığında bir olay oluşur. Üretici, olayı oluşturan bileşendir. Örneğin, bir e-ticaret sitesinde sipariş verildiğinde bu bileşen olayın üreticisi konumundadır. Tüketici, olayı alan ve ona tepki veren bileşendir. Örneğin, bir ödeme tamamlandığında fatura oluşturan servis bu türdendir. Olay yöneticisi ise, olayların üreticilerden tüketicilere iletilmesini sağlayan aracı sistemdir.

Böyle bir örüntüde bileşenler arasında sıkı bağımlılık yoktur. Bu durum kurulan yapının ölçeklenebilirliğini arttırır. Diğer yandan olaylar gerçek zamanlı işlenir ve yüksek performans sağlama gibi avantajları bünyesinde barındırır. Bu sayede büyük veri ve mikroservis tabanlı mimarilerde etkili bir çözüm sunar. Ayrıca esnek ve modüler olup mimari bileşenlere yeni özellikler eklendiğinde diğer bileşenler etkilenmez.

Bütün bu avantajlarına karşın olayların izlenmesinde ve hataların ayıklanmasındaki artan zorluklar, olay işlenmesi anında yaşanan gecikmeler ve bileşenlerin durumların bilgilerini tutmasındaki zorluklar bu örüntünün zorlukları olarak belirtilebilir.

Çalışmada seçilen bu örüntü REST iletişimi ve eş anlı mesajlaşma yapısını kullanmıştır. İletişim için tanımlanan uç noktalar OpenAPI standardı ile Swagger aracı kullanılarak görselleştirilmiştir. Gerçeklenen sistem görsel servisi birimi, görsel işleyici birimi ve mesajlaşma servisi ile Şekil 11'de görüldüğü gibi oluşturulmuştur.

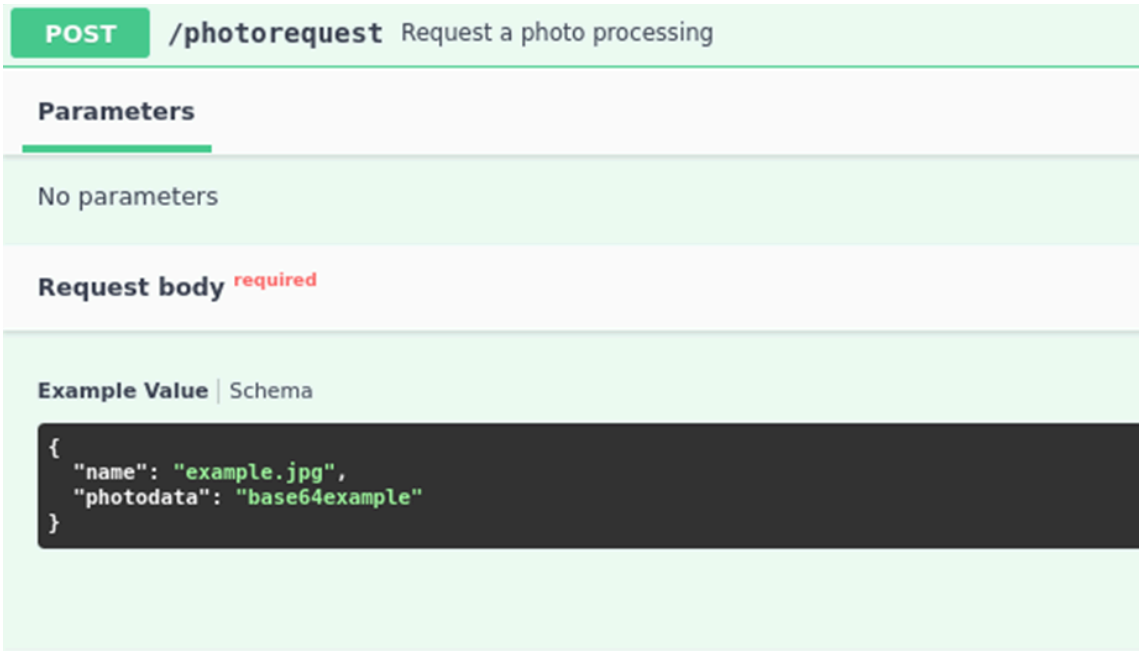


Şekil 11. Olay Güdümlü Örüntü Mimari tasarım.

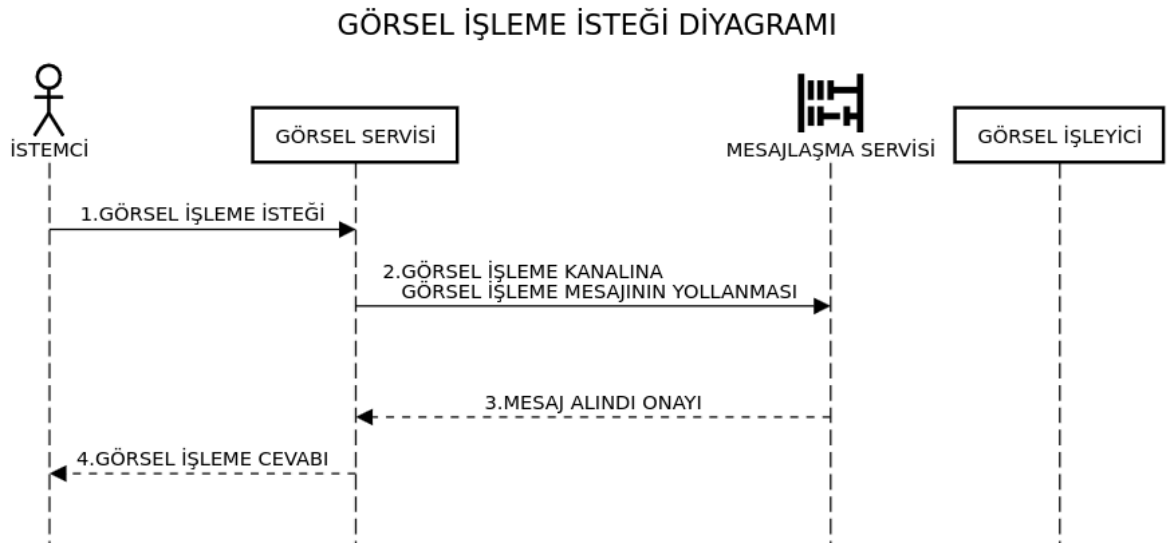
3.2.1. Görsel İşleme

Görsel işleme servisinin kullanıcı ile etkileşimde bulunduğu sıralama diyagramı Şekil 13'de belirtilmiştir. İşlem kullanıcının görsel servisine "photorequest" ucundan Şekil 12'de Swagger görseli olan POST isteğini yapmasıyla başlamaktadır. İsteğin veri kısmına görülen veri JSON formatında eklenir. İsteğin gönderildiği örnek adres "http://X.X.X.X:8080/photorequest"'dir.

İstek servis görsel servisine geldikten sonra mesajlaşma servisinin "görsel işleme" kanalına aktarılır. Mesajlaşma servisi kanala mesajın ulaştığı ile ilgili bilgiyi görsel servisine geri döndürür. İlgili birim de kullanıcıya işlemin başladığıyla ilgili bilgiyi döndürür.



Şekil 12. Görsel işleme ucu swagger örneği.

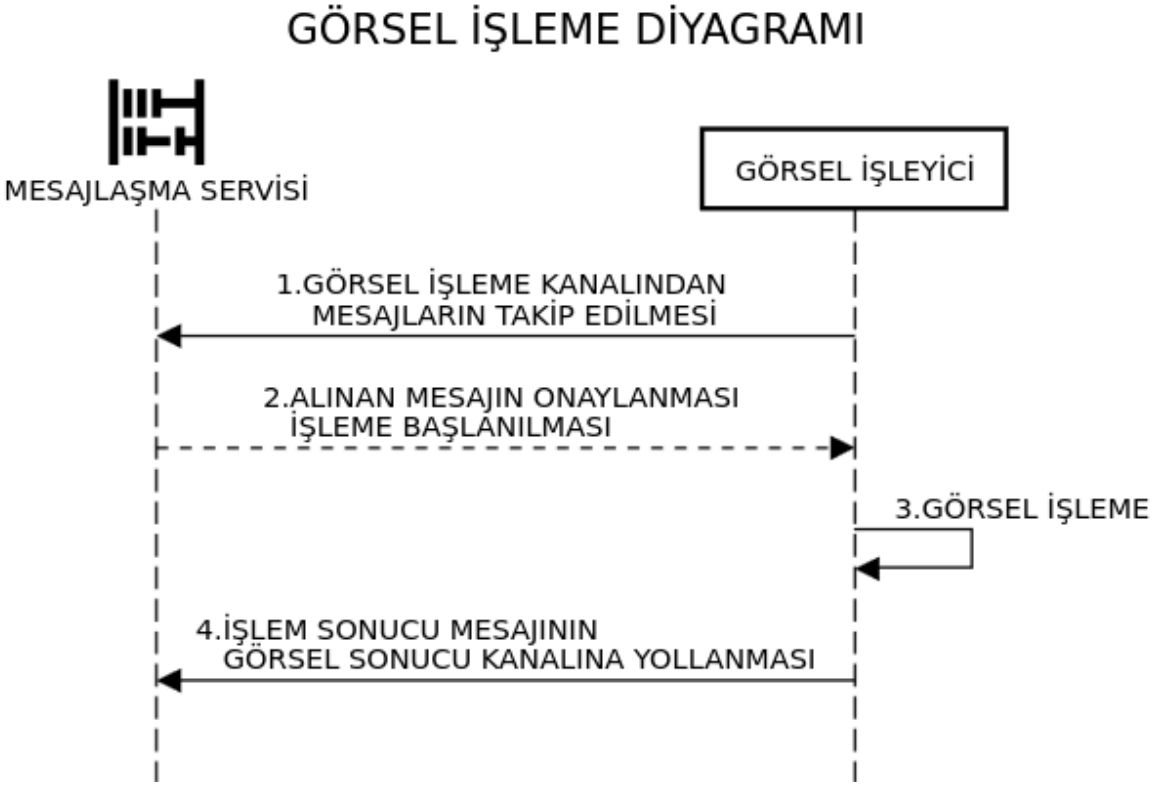


Şekil 13. Görsel işleme isteği sıralama diyagramı.

Şekil 14'te "görsel işleme" ve "işlem sonucu" isminde iki kanal bulunmaktadır. Olay Güdümlü mimaride kanal, genellikle mesajların bir üreticiden (producer) tüketiciye (consumer) aktarılmasını sağlayan bir iletişim hattıdır. Bu kavram, olayları taşıyan bir yol veya mekanizma olarak düşünülebilir. Kanallar, üreticinin ürettiği olayları tüketicilere aktarmak için kullandıkları bir araçtır. Gevşek bağlılık sağlayarak tarafların birbirlerinin varlığından haberdar olmadan çalışmalarını mümkün kılar. Ayrıca mesajların güvenli bir şekilde aktarılmasını ve gerektiği sırayla alınmasını da destekler.

Bunları sağlamak adına bulut servis sağlayıcıları mesajlaşma servislerinde kendilerinin oluşturduğu ya da RabbitMQ ve Apache Kafka gibi mesajlaşma kuyruğu uygulamalarının kullandığı protokoller kullanmaktadırlar. RabbitMQ Gelişmiş Mesajlaşma Kuyruğu Protokolü (Advanced Messaging Queue Protocol - AMQP) ve Apache Kafka İletim Kontrol Protokolü (Transmission Control Protocol - TCP) üzerinden ikili aktarımı (Binary over TCP) protokollerini kullanmaktadır (Apache, t.y.; RabbitMQ, t.y.). Microsoft Azure Service Bus servisi belgelendirmelerinde AMQP kullandığını belirtmektedir fakat AWS SQS ve GCP PUB/SUB servislerinde böyle bir ifade görülmemektedir (AWS, 2025a; GCP, 2025a; Microsoft Azure, 2025a).

Görsel işleyici birim belirli aralıklarla “görsel işleme” kanalına denetim mesajı atar. Eğer kanalda bir istek var ise işleme başlar. Bu işlem uygulamada gerçek bir işlemi taklit edecek şekilde alınır ve on saniye bekletildikten sonra “işlem sonucu” kanalına işlemin sonucu iletilir. Bu işlemler aşağıdaki diyagramda gösterilmiştir.



Şekil 14. Görsel işleme sıralama diyagramı.

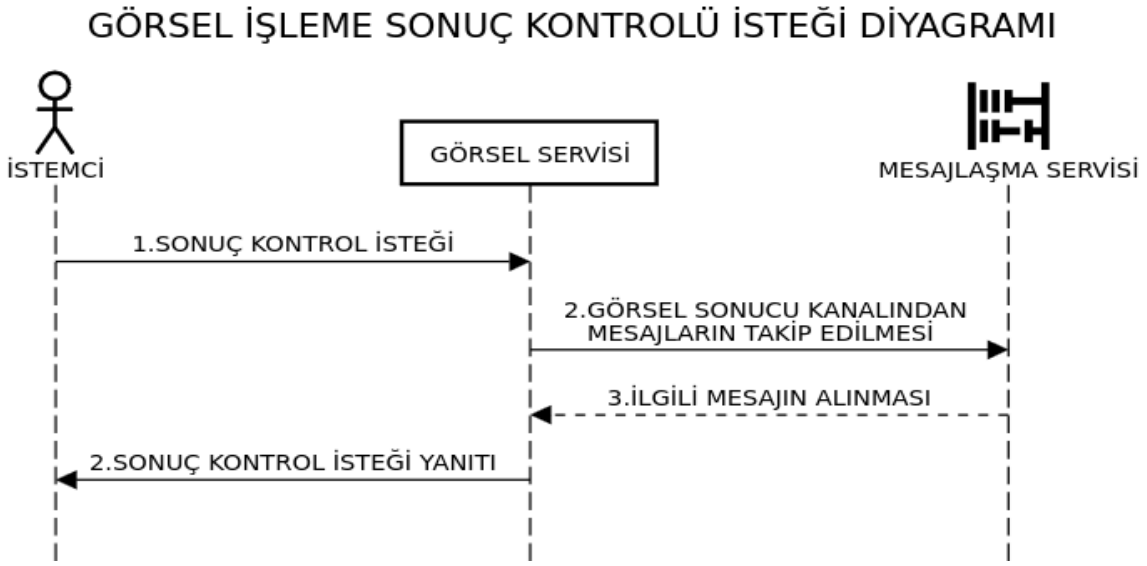
3.2.2. Görsel İşlem Sonucu Sorgulama

İşlem kullanıcının görsel servisi birimine “getresult” ucundan Şekil 15’de Swagger görseli olan GET isteğini yapmasıyla başlamaktadır. İsteğin “name” sorgulama değişkenine görselin adı eklenir ve örnek adresi “http://X.X.X.X:8080/getresult?name=test” dir.



Şekil 15. Görsel işleme sonucu sorgulama ucu swagger örneği.

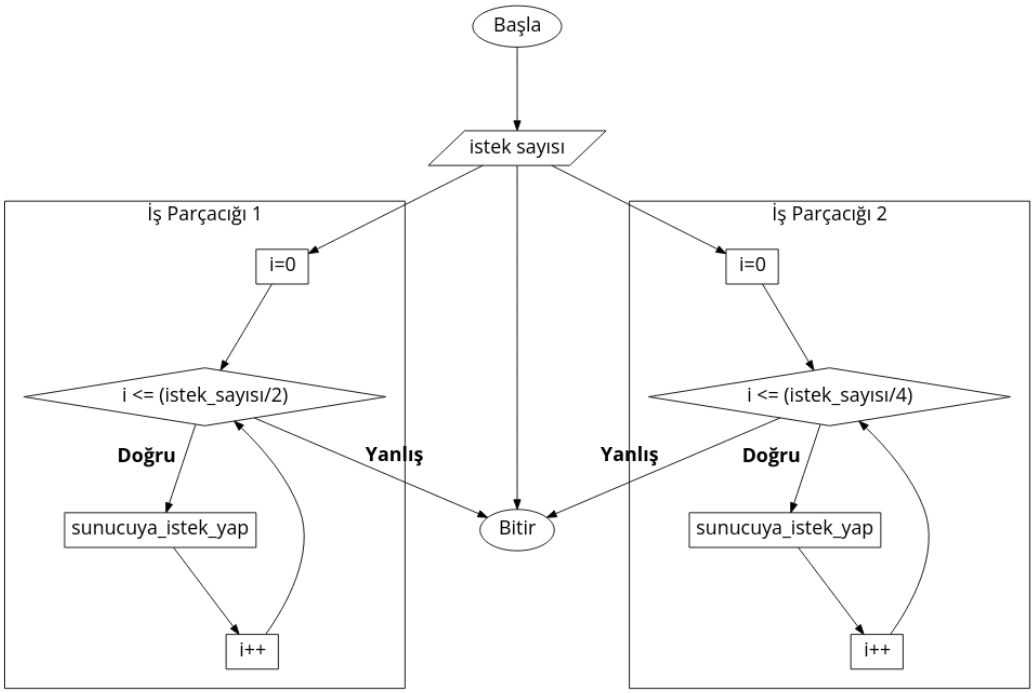
Sorgu Şekil 16'da gösterildiği gibi görsel servisi birimine geldikten sonra servis ilgili kanalı kontrol eder. Kontrol öncelikle servisin üstünde barındırdığı aralıklı sorgu sonuçlarından oluşturulan veri yapısında yapılır. Eğer burada yoksa mesajlaşma servisine tekrar sorgu yapılır gelen sonuç istemciye aktarılır.



Şekil 16. Görsel işleme sonuç kontrolü sıralama diyagramı.

3.3 İzleme Uygulaması

Örnek olarak oluşturduğumuz iki örüntünün izlenmesi için bulut servis sağlayıcılarında izlemenin gerçekleştirimi için kullanıma sunulan Tablo 1'deki araçlar kullanılmıştır. Bu uygulamalar için Şekil 17'deki örnek algoritmaya benzer bir algoritma kullanılarak test sanal makinesi üzerinden Vegeta test aracı kullanılarak yapılmıştır (Scaleway, t.y.).



Şekil 17. Yük testi algoritma akış şeması.

Şekil 17'de akış şeması verilen çok iş parçacıklı algoritma Vegeta yük testi tarafından kullanılan algoritma örneğidir. Algoritmada dışarıdan alınan istek sayısı kadar istek, iş parçacığı sayısına bölünerek toplamda istenilen saniye içerisinde istenilen konuma gönderilir. Bu konumda bulunan servise gönderilen istekler ile yük oluşturulur ve vegeta aracı tarafından görselleştirilir. Bu izlemenin sonrasında sanal makineler, mesajlaşma servisleri ve veri tabanlarında seçilen ölçütler için örnek alarmlar oluşturulmuş ve bu alarmların tetiklenmesi sağlanmıştır.

3.4 Nesnel Değerlendirme ve Ölçütler

Çalışma boyunca işlenen ölçütler bu bölümde ele alınmıştır. Ele alınan ölçütler sanal makineler, veri tabanları ve mesajlaşma servisleri için hesaplanma denklemleri ile alt başlıklar şeklinde paylaşılmıştır. Ölçütlerin hesaplanma denklemleri literatür incelenmelerinden ve servis sağlayıcılarının belgelendirmelerinden oluşturulmuştur. Bulut servis sağlayıcılarının kapalı kaynak sistemler olması ve net bir denklem sağlamamaları nedeniyle bu yöntem izlenmiştir. Denklemlerde süreler bulut servis sağlayıcıların izleme araçları için sunduğu işleme hassasiyet aralıkları olarak belirtilmektedir. Bu çalışmada hassasiyet 1 saniye olarak ayarlanmıştır.

3.4.1. Sanal Makine Ölçütleri

Sanal makinenin güvenirliği ve çalışan uygulamanın durumunu izlemek için kullanılan ölçütler bu bölümde işlenmiştir.

Sanal makinenin kullandığı işlemci durumunu ifade eden işlemci kullanımı ölçütüdür. Bu ölçüt, sanal makinenin sahip olduğu işlemcilerin ne kadar yük altında olduğunu bizlere göstermesi sebebiyle sistemin sağlığı açısından önem taşımaktadır. Denklem 3.1'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Tanenbaum ve Bos, 2014; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b). Denklemde p bir sürecin girdi ve çıktı işlemlerinin tamamlanması için bekleme olasılığını, n bellekte bulunan bütün süreçleri ifade etmektedir.

$$\text{İşlemci Kullanımı (\%)} = (1 - p^n) \times 100 \tag{3.1}$$

Bellek kullanımı ölçütü makinenin sahip olduğu bellekten ne kadarını kullandığını ifade etmektedir. Bu ölçüt, belleğin yeterli ve verimli kullanıldığının izlenebilmesi adına önem taşımaktadır. Denklem 3.2'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından

oluşturulmuştur (Stallings, 2001; Silberschatz vd. 2018; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\textbf{Bellek Kullanımı (\%)} = \left(\frac{\textbf{Kullanılan Bellek Miktarı}}{\textbf{Toplam Bellek Miktarı}} \right) \times 100 \tag{3.2}$$

Depolama kullanımı ölçütü makinenin sahip olduğu depolamadan ne kadarını kullandığını ifade etmektedir. Bu ölçüt, depolamanın yeterli ve verimli kullanıldığının izlenebilmesi adına önem taşımaktadır. Denklem 3.3'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Tanenbaum ve Bos, 2014; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\textbf{Depolama Kullanımı (\%)} = \left(\frac{\textbf{Kullanılan Depolama Miktarı}}{\textbf{Toplam Depolama Miktarı}} \right) \times 100 \tag{3.3}$$

Ağdan makineye gelen veri ölçütü makinenin ağ arayüzlerine gelen veri akışını belirtmektedir. Bu ölçüt, makinenin ağdan ağ genişliğinin altında veri aldığının ve sağlıklı bir ağ iletişimine sahip olup olmadığının kontrolü için gereklidir. Denklem 3.4'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Julian, 2017; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\textbf{Ağdan Makineye Gelen Veri} = \frac{\textbf{Toplam Alınan Veri}}{\textbf{Geçen Süre}} \tag{3.4}$$

Ağa makineden çıkan veri ölçütü makinenin ağ arayüzlerine gönderdiği veri akışını belirtmektedir. Bu ölçüt makinenin ağa, ağ genişliğinin altında veri gönderdiği ve sağlıklı bir ağ iletişimine sahip olup olmadığının kontrolü için gereklidir. Denklem 3.5'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Julian, 2017; AWS, 2025b; GCP 2025b;

Microsoft Azure 2025b).

$$\text{Ağa Makineden Çıkan Veri} = \frac{\text{Toplam Gönderilen Veri}}{\text{Geçen Süre}} \tag{3.5}$$

3.4.2. Mesajlaşma Servisleri Ölçütleri

Bulut servis sağlayıcıları tarafından sağlanan mesajlaşma servislerinin beklenen şekilde davrandığını izlemek adına seçilen ölçütler bu bölümde işlenmiştir.

Gönderilen mesaj sayısı ölçütü mesajlaşma servisinin ilgili kanalına üretici tarafından gönderilen mesaj sayısını ifade etmektedir. Bu ölçüt, üreticiden gönderildiği bilinen mesaj sayısı ile servise ulaşan mesaj sayısı arasında tutarlılığı takip etmek adına önem taşımaktadır. Denklem 3.6'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Wittig vd., 2019; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\text{Gönderilen Mesaj Sayısı} = \frac{\text{Toplam Gönderilen Mesaj Sayısı}}{\text{Geçen Süre}} \tag{3.6}$$

Alınan mesaj sayısı ölçütü mesajlaşma servisinin ilgili kanalından tüketicinin aldığı mesaj sayısını ifade etmektedir. Bu ölçüt, üreticiden alındığı bilinen mesaj sayısı ile servisten gönderilen mesaj sayısı arasında tutarlılığı takip etmek adına önem taşımaktadır. Denklem 3.7'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Wittig vd., 2019; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\text{Alınan Mesaj Sayısı} = \frac{\text{Toplam Alınan Mesaj Sayısı}}{\text{Geçen Süre}} \tag{3.7}$$

Silinen mesaj sayısı ölçütü mesajlaşma servisinin ilgili kanalından uygulama tarafından silinen mesaj sayısını ifade etmektedir. Bu ölçüt, uygulama tarafından silindiği bilinen mesaj sayısı ile servisten silinen mesaj sayısı arasında tutarlılığı takip etmek adına önem taşımaktadır. Denklem 3.8'de ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Wittig vd., 2019; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\textbf{Silinen Mesaj Sayısı} = \frac{\textbf{Toplam Silinen Mesaj Sayısı}}{\textbf{Geçen Süre}} \tag{3.8}$$

3.4.3. Veri Tabanı Servisleri Ölçütleri

Bulut servis sağlayıcıları tarafından sağlanan veri tabanı servislerinin beklenen şekilde davrandığını izlemek adına seçilen ölçütler bu bölümde işlenmiştir.

Okuma işlem sayısı ölçütü, veri tabanı servisinin sahip olduğu depolama biriminden yaptığı okuma işlemini ifade etmektedir. Bu ölçüt, veri tabanının yaptığı okuma sayısının depolamanın sınırlarından düşük olduğu ve sağlıklı şekilde işlem yapabildiğini izlemek adına önem sağlamaktadır. Denklem 3.9'da ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından oluşturulmuştur (Wittig vd., 2019; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\textbf{Okuma İşlem Sayısı} = \frac{\textbf{Toplam Okuma Sayısı}}{\textbf{Geçen Süre}} \tag{3.9}$$

Yazma işlem sayısı ölçütü, veri tabanı servisinin sahip olduğu depolama biriminden yaptığı okuma işlemini ifade etmektedir. Bu ölçüt, veri tabanının yaptığı yazma sayısının depolamanın sınırlarından düşük olduğu ve sağlıklı şekilde işlem yapabildiğini izlemek adına önem sağlamaktadır. Denklem 3.10'da ölçütün nasıl hesaplandığı paylaşılmıştır. Bu denklem bulut servis sağlayıcılarının belgelendirmelerinden ve literatür kaynaklarından

oluşturulmuştur (Wittig vd., 2019; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

$$\text{Yazma İşlem Sayısı} = \frac{\text{Toplam Yazma Sayısı}}{\text{Geçen Süre}} \tag{3.10}$$

Veri tabanı bağlantı sayısı ölçütü, veri tabanı servisinin üzerine yapılan bağlantı sayısını ifade etmektedir. Bu ölçüt, veri tabanına yapılan bağlantı sayısının limitlerin içerisinde olduğunu takip etmek adına önem taşımaktadır. Ölçüt anlık olarak veri tabanına yapılan bağlantı sayısını gösterdiğinden bir formülasyona sahip değildir (Wittig vd., 2019; AWS, 2025b; GCP 2025b; Microsoft Azure 2025b).

DÖRDÜNCÜ BÖLÜM
ARAŞTIRMA BULGULARI

Bu çalışmada bir önceki kısımda ele alınan iki farklı örüntü, üç farklı bulut servis sağlayıcısı üzerinde gerçekleştirilmiş ve Şekil 18'de bahsedilen algoritma doğrultusunda Vegeta test aracı ile bu örüntüler üzerine saniyede 100, 250 ve 500 istek bir saatlik kesit içerisinde kademeli olarak yapılarak sistemlerin yük altındaki durumları Bölüm 3.4'te verilen ölçütler özelinde gözlemlenmiştir. Örüntülerin yük altında sergiledikleri durumlara ait sonuçlar bu bölümde paylaşılmaktadır.

4.1. Uygulama Programlama Arayüzü Geçidi Örüntüsüne Ait Bulgular

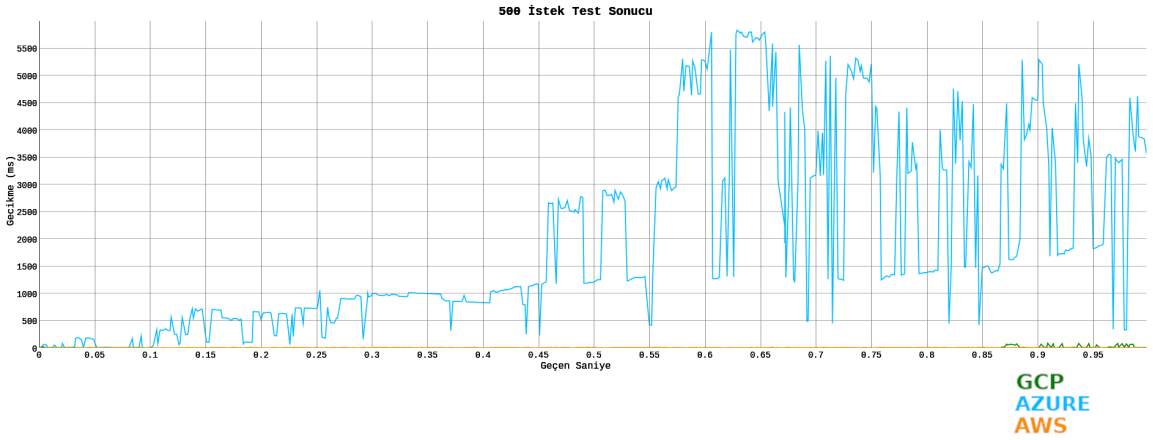
Bölüm 3.1'de açıklanmış olan API Geçidi Örüntüsü (Şekil 5) Amazon Web Servisi, Google Bulut Platformu ve Microsoft Azure'da oluşturulmuştur. Oluşturulan mimari sırasıyla kullanıcı ekleme, kullanıcının getirilmesi, ödeme ekleme ve ödemenin getirilmesi durumlarını ele almaktaydı. Sistemin uç birimlerinin üç farklı bulut servis sağlayıcısındaki tüm durumlarına ait yük altındaki gecikme performansları incelenmiştir.

Modelin API Geçidi, kullanıcı ve veri tabanı servislerine ait olan ve modelin üzerinde çalıştığı sanal makinelerin işlemci kullanımı, ağ trafiği, okuma/yazma ve veri tabanına yapılan bağlantı sayısı gibi durumları izlenerek elde edilen sonuçlar bu kısımda incelenmiştir.

4.1.1 Uç Noktalarındaki Yük Testi Raporları

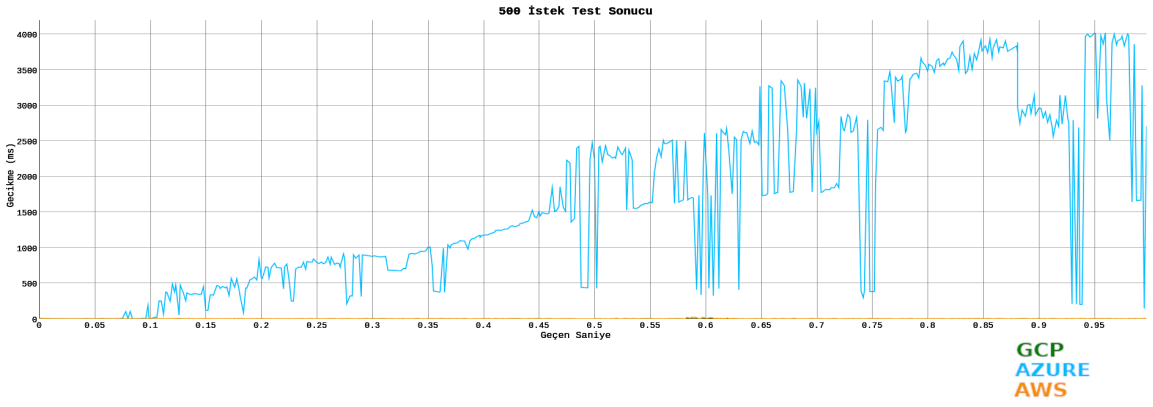
Kurgulanan sistemin uç noktalarına bahsi geçen Vegeta programı ile Şekil 17'de açıklanan algoritma doğrultusunda yük testi oluşturulmuştur. İstekler sırasıyla 100, 250 ve 500 adet olmak üzere bir saatlik süre içerisinde kademeli bir şekilde sisteme uygulanmıştır. Şekil 18'de kullanıcı ekleme ucunda yapılan, Şekil 19'da kullanıcı sorgulama ucunda yapılan, Şekil 20'de ödeme ekleme ucunda yapılan, Şekil 21'de ödeme sorgulama ucunda yapılan yük testlerinin raporları görselleştirilmiştir. Bu görseller 500 istek için yapılmış en yoğun test tipinin görselleridir. 100 ve 250 istek için yapılan testler EK-1 kısmında

sunulmuştur.



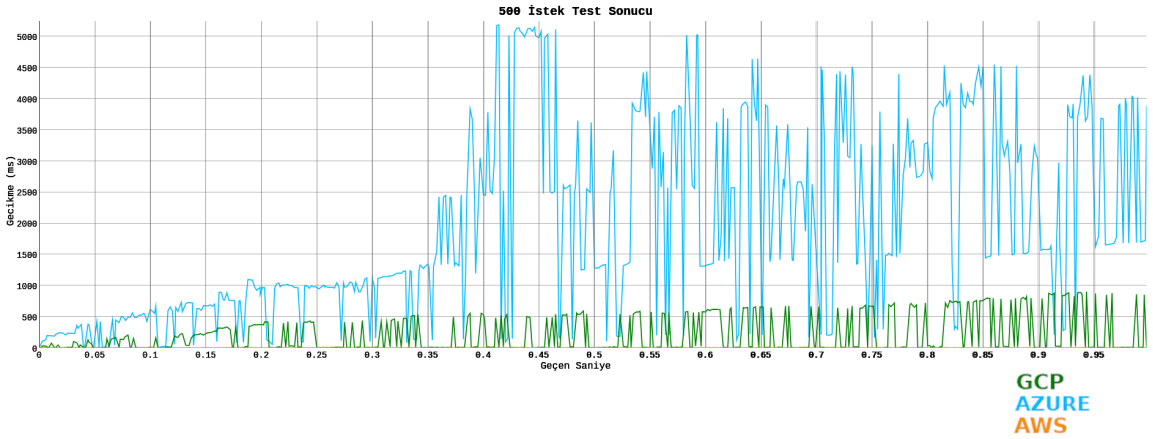
Şekil 18. Kullanıcı ekleme yük testi sonuç görseli.

Kullanıcı ekleme ucu için yapılan deneylerin çıktıları AWS, GCP ve Azure için şu şekilde detaylandırılmıştır; AWS üzerinde yapılan deneylerde cevap süreleri ortalama 2,917 ms, en düşük 1,3 ms ve en yüksek 14,56 ms olarak ölçülmüştür. GCP üzerinde yapılan deneylerde cevap süreleri ortalama 5,819 ms, en düşük 2,319 ms ve en yüksek 87,72 ms olarak ölçülmüştür. Azure üzerinde yapılan deneylerde cevap süreleri ortalama 1,912 s, en düşük 3,844 ms ve en yüksek 5,832 s olarak ölçülmüştür. Bu teste göre en iyi başırımı AWS senaryosu vermiştir oluşan farklılıkların sebebi veri tabanı ulaşım süreleri, ağ gecikmeleri gibi servis sağlayıcılara bağlı durumlar olabilmektedir. Bu sonuçların arasındaki farklılık,yapılan ölçüt panellerinde izlenememektedir.



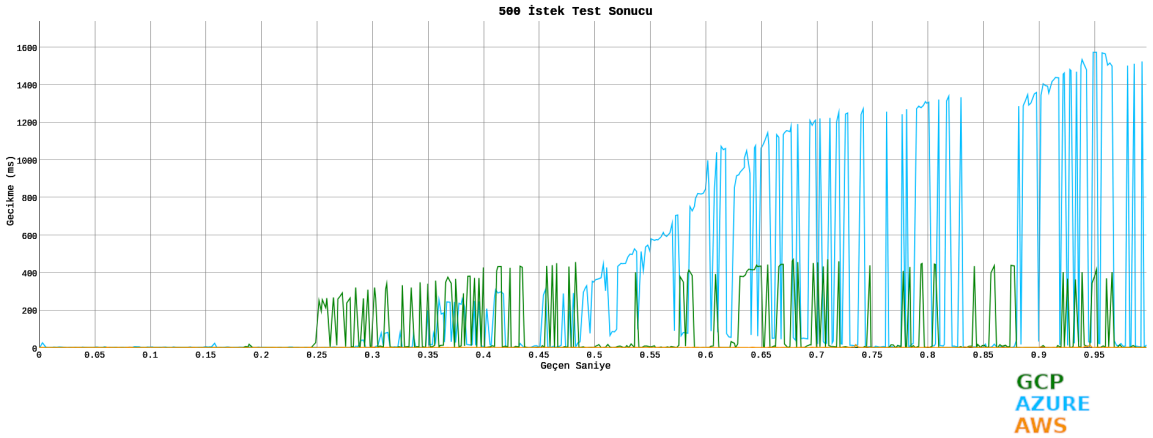
Şekil 19. Kullanıcı sorgulama yük testi sonuç görseli.

Kullanıcı sorgulama ucu için yapılan deneylerin çıktıları AWS, GCP ve Azure için şu şekilde detaylandırılmıştır; AWS üzerinde yapılan deneylerde cevap süreleri ortalama 2,5151 ms, en düşük 0,9209 ms ve en yüksek 26,2044 ms olarak ölçülmüştür. GCP üzerinde yapılan deneylerde cevap süreleri ortalama 1,9144 ms, en düşük 1,4773 ms ve en yüksek 5,0279 ms olarak ölçülmüştür. Azure üzerinde yapılan deneylerde cevap süreleri ortalama 1,664 s, en düşük 1,6825 ms ve en yüksek 4,021 s olarak ölçülmüştür. Bu teste göre en iyi başırımı AWS senaryosu vermiştir oluşan farklılıkların sebebi veri tabanı ulaşım süreleri, ağ gecikmeleri gibi servis sağlayıcılara bağlı durumlar olabilmektedir. Bu sonuçların arasındaki farklılık, yapılan ölçüt panellerinde izlenememektedir.



Şekil 20. Ödeme ekleme yük testi sonuç görseli.

Ödeme ekleme ucu için yapılan deneylerin çıktıları AWS, GCP ve Azure için şu şekilde detaylandırılmıştır; AWS üzerinde yapılan deneylerde cevap süreleri ortalama 2,5326 ms, en düşük 1,7877 ms ve en yüksek 1,7877 ms olarak ölçülmüştür. GCP üzerinde yapılan deneylerde cevap süreleri ortalama 213,25 ms, en düşük 2,5161 ms ve en yüksek 904,68 ms olarak ölçülmüştür. Azure üzerinde yapılan deneylerde cevap süreleri ortalama 1,931 s, en düşük 5,0507 ms ve en yüksek 5,187 s olarak ölçülmüştür. Bu teste göre en iyi başırımı AWS senaryosu vermiştir oluşan farklılıkların sebebi veri tabanı ulaşım süreleri, ağ gecikmeleri gibi servis sağlayıcılara bağlı durumlar olabilmektedir. Bu sonuçların arasındaki farklılık, yapılan ölçüt panellerinde izlenememektedir.



Şekil 21. Ödeme sorgulama yük testi sonuç görseli.

Ödeme sorgulama ucu için yapılan deneylerin çıktıları AWS, GCP ve Azure için şu şekilde detaylandırılmıştır; AWS üzerinde yapılan deneylerde cevap süreleri ortalama 2,3234 ms, en düşük 1,374 ms ve en yüksek 17,302 ms olarak ölçülmüştür. GCP üzerinde yapılan deneylerde cevap süreleri ortalama 76,8646 ms, en düşük 1,2752 ms ve en yüksek 471,97 ms olarak ölçülmüştür. Azure üzerinde yapılan deneylerde cevap süreleri ortalama 302,93 ms, en düşük 1,5855 ms ve en yüksek 1,573 s olarak ölçülmüştür. Bu teste göre en iyi başırımı AWS senaryosu vermiştir oluşan farklılıkların sebebi veri tabanı ulaşım süreleri, ağ gecikmeleri gibi servis sağlayıcılara bağlı durumlar olabilmektedir. Bu sonuçların arasındaki farklılık, yapılan ölçüt panellerinde izlenememektedir bu durum sonuç kısmında değerlendirilmiştir.

4.1.2 Servislere Ait Ölçüt Durumları

Örüntüye ait olan dört farklı senaryodaki (kullanıcının eklenmesi, kullanıcının sorgulanması, ödemenin eklenmesi ve ödemenin sorgulanması) bahsi geçen bulut sağlayıcılarındaki sanal makineler ve veri tabanlarının durumları incelenmiştir.

Amazon Web Servisleri

API Geçidi örüntüsü AWS bulut sağlayıcısı üzerinde gerçeklenmiştir. Tablo 2'de sırasıyla gerçekleştirilen yük testlerinde API Geçidi servisinin üzerinde çalıştığı sanal

makinenin dört farklı senaryodaki işlemci kullanımı, gelen ve giden ağ trafiğinde sergilediği durum görülmektedir. Kullanıcı ve veri tabanın servislerinin benzer şekilde bu dört farklı senaryodaki ölçütleri EK-3'de sunulmuştur.

Tablo 2

Api geçidi sanal makinesi AWS ölçüt durumları

Test Ucu	**İşlemci Kullanımı**		**Ağdan Gelen Veri**		**Ağa giden veri**	
Kullanıcı ekleme	En Az %4	En Çok %5	En Az 0,05 MiB/S	En Çok 0,168 MiB/S	En Az 0,03 MiB/S	En Çok 0,167 MiB/S
Kullanıcı Sorgulama	En Az %4,3	En Çok %4,93	En Az 0,03 MiB/S	En Çok 0,071 MiB/S	En Az 0,01 MiB/S	En Çok 0.067 MiB/S
Ödeme ekleme	En Az %4,81	En Çok %5,01	En Az 0.01 MiB/S	En Çok 0.207 MiB/S	En Az 0,01 MiB/S	En Çok 0.198 MiB/S
Ödeme sorgulama	En Az %4,23	En Çok %4,5	En Az 0,01 MiB/S	En Çok 0,072 MiB/S	En Az 0,01 MiB/S	En Çok 0,067 MiB/S

Google Bulut Platformu

API Geçidi örüntüsü GCP bulut sağlayıcısı üzerinde gerçekleştirilmiştir. Tablo 4'te sırasıyla gerçekleştirilen yük testlerinde API Geçidi servisinin üzerinde çalıştığı sanal makinenin dört farklı senaryodaki işlemci kullanımı, bellek kullanımı, depolama kullanımı, gelen ve giden ağ trafiğinde sergilediği durum görülmektedir. Kullanıcı ve veri tabanın servislerinin benzer şekilde bu dört farklı senaryodaki ölçütleri EK-3'de sunulmuştur.

Tablo 3

Api geçidi sanal makinesi GCP ölçüt durumları

Test Ucu	İşlemci Kullanımı		Bellek Kullanımı		Depolama Kullanımı		Ağdan Gelen Veri		Ağa giden Veri	
Kullanıcı ekleme	En Az %3	En Çok %42	En Az %10	En Çok %17	En Az %20	En Çok %20	En Az 1 KiB/S	En Çok 5 KiB/S	En Az 5 KiB/S	En Çok 8,7 KiB/S
Kullanıcı Sorgulama	En Az %4,3	En Çok %45	En Az %17	En Çok %20	En Az %20	En Çok %20	En Az 1 KiB/S	En Çok 4,4 KiB/S	En Az 6 KiB/S	En Çok 9 KiB/S
Ödeme Ekleme	En Az %2	En Çok %35	En Az %16	En Çok %20	En Az %20	En Çok %20	En Az 1 KiB/S	En Çok 4 KiB/S	En Az 5 KiB/S	En Çok 9 KiB/S
Ödeme sorgulama	En Az %5	En Çok %48	En Az %17	En Çok %20	En Az %20	En Çok %20	En Az 2 KiB/S	En Çok 15 KiB/S	En Az 6 KiB/S	En Çok 17 KiB/S

Microsoft Azure

API Geçidi modeli Microsoft Azure bulut sağlayıcısı üzerinde gerçekleştirilmiştir. Tablo 4'te sırasıyla gerçekleştirilen yük testlerinde API Geçidi servisinin üzerinde çalıştığı sanal makinenin dört farklı senaryodaki işlemci kullanımı, gelen ve giden ağ trafiğinde sergilediği durum görülmektedir. Kullanıcı ve veri tabanın servislerinin benzer şekilde bu dört farklı senaryodaki ölçütleri EK-3'de sunulmuştur.

Tablo 4

API Geçidi sanal makinesi Microsoft Azure ölçüt durumları

Test Ucu	İşlemci Kullanımı		Ağdan Gelen Veri		Ağa giden veri	
Kullanıcı ekleme	En Az %0,6	En Çok %0,9	En Az 70 KiB/S	En Çok 885 KiB/S	En Az 48,83 KiB/S	En Çok 488,28 KiB/S
Kullanıcı Sorgulama	En Az %0,6	En Çok %1,6	En Az 88 KiB/S	En Çok 878,91 KiB/S	En Az 48,87 KiB/S	En Çok 439,45 KiB/S
Ödeme ekleme	En Az %0,6	En Çok %1,6	En Az 47 KiB/S	En Çok 488,28 KiB/S	En Az 49 KiB/S	En Çok 488,28 KiB/S
Ödeme sorgulama	En Az %0,6	En Çok %1,35	En Az 46 KiB/S	En Çok 305 KiB/S	En Az 48,83 KiB/S	En Çok 303 KiB/S

Yük testlerinde beklenildiği gibi sanal makinalara gelen ve makinalardan çıkan veri girişlerinde yükselmeler görülmüştür. Aynı zamanda makinaların işlemci kullanımlarındaki artış da normal seyretmektedir. Bu bulgular incelendiğinde ölçütlerde anormal bir duruma rastlanılmamıştır. Yük testlerinde ortaya çıkan gecikme farklılıkları ve ölçüt panellerinin olağan durumu sonuç kısmında değerlendirilmiştir.

4.2 Olay Güdümlü Örüntüye Ait Bulgular

Bölüm 3.2'de açıklanmış olan olay güdümlü örüntü (Şekil 11) Amazon Web Servisi, Google Bulut Platformu ve Microsoft Azure'da oluşturulmuştur. Oluşturulan mimari sırasıyla görsel oluşturma ve oluşturulan görsellerin kullanıcı tarafından getirilmesi durumlarını ele almaktaydı. Modelin uç birimlerinin üç farklı bulut servis sağlayıcısındaki

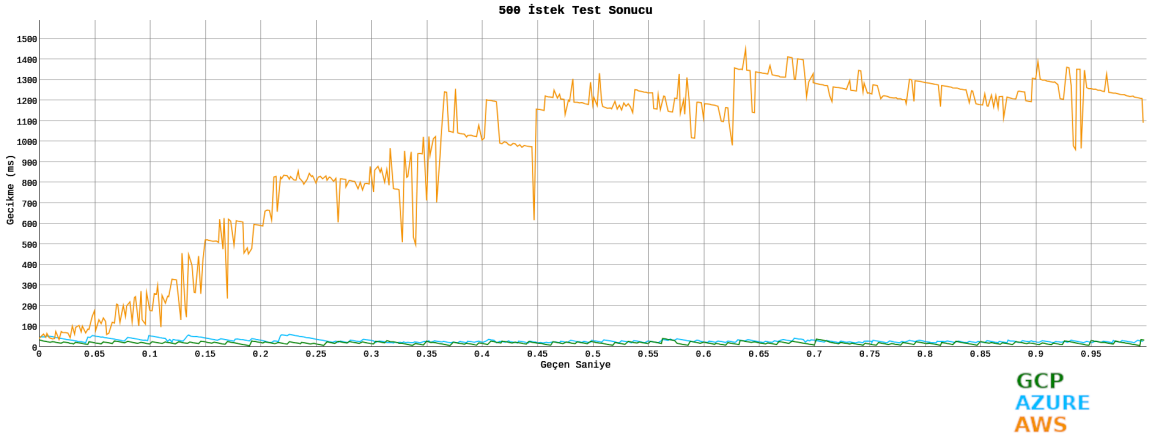
tüm durumlarına ait yük altındaki gecikme performansları incelenmiştir.

Modelin, API Geçidi, kullanıcı ve veri tabanı servislerine ait olan ve modelin üzerinde çalıştığı sanal makinelerin işlemci kullanımı, ağ trafiği, okuma/yazma ve veri tabanına yapılan bağlantı sayısı gibi durumları izlenerek elde edilen sonuçlar bu kısımda incelenmiştir.

4.2.1 Uç Noktalarındaki Yük Raporları

Oluşturulan bu mimarinin uç noktalarına bahsi geçen Vegeta programı ile Şekil 18'de açıklanan algoritma doğrultusunda yük testi oluşturulmuştur. İstekler sırasıyla 100, 250 ve 500 adet olmak üzere bir saatlik süre içerisinde kademeli bir şekilde modele uygulanmıştır.

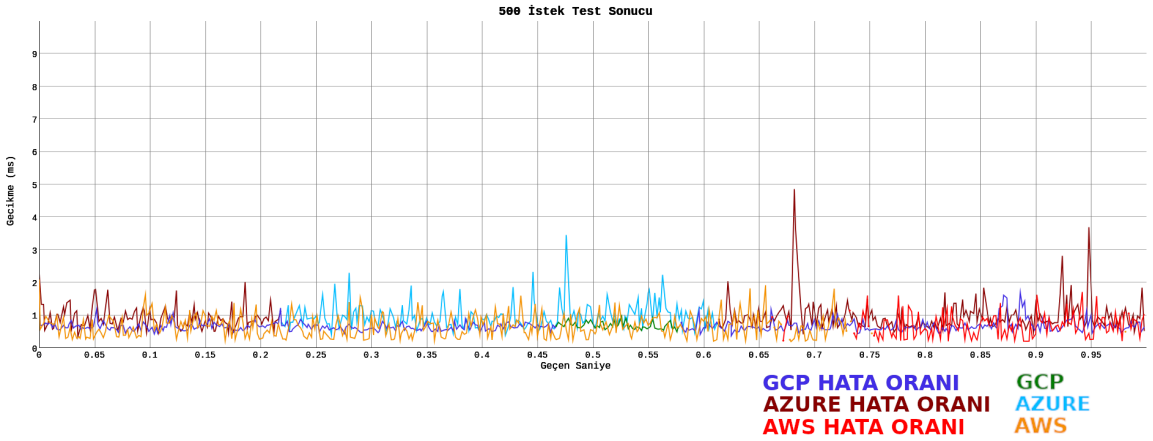
Şekil 22'de görsel işleme ucuna yapılan, Şekil 23'de görsel işlem sonucu sorgulama ucunda yapılan yük testlerinin raporları görselleştirilmiştir. Bu görseller 500 istek için yapılmış en yoğun test tipinin görselleridir 100 ve 250 istek için yapılan testler EK-2 kısmında sunulmuştur.



Şekil 22. Görsel işleme yük testi sonuç görseli.

Görsel işleme ucu için yapılan deneylerin çıktıları AWS, GCP ve Azure için şu şekilde detaylandırılmıştır; AWS üzerinde yapılan deneylerde cevap süreleri ortalama

953,60 ms, en düşük 36,4505 ms ve en yüksek 17,302 ms olarak ölçülmüştür. GCP üzerinde yapılan deneylerde cevap süreleri ortalama 19,3276 ms, en düşük 5,8261 ms ve en yüksek 41,0855 ms olarak ölçülmüştür. Azure üzerinde yapılan deneylerde cevap süreleri ortalama 30,1369 ms, en düşük 19,3019 ms ve en yüksek 61,3912 ms olarak ölçülmüştür. Bu teste göre en iyi başırımı GCP senaryosu vermiştir oluşan farklılıkların sebebi mesajlaşma servislerinin tekrarlı mesaj işleme kısıtı, mesajların ulaşım ve sıralama hızı, ağ gecikmeleri gibi servis sağlayıcılara bağlı durumlar olabilmektedir. Bu sonuçların arasındaki farklılık, yapılan ölçüt panellerinde alınan ve gönderilen mesaj sayıları arasındaki fark şeklinde gözükebilmektedir. Bu durum sonuç kısmında değerlendirilmiştir.



Şekil 23. Görsel işlem sonucu sorgulama yük testi sonuç görseli.

Görsel işlem sonucu sorgulama ucu için yapılan deneylerin çıktıları AWS, GCP ve Azure için şu şekilde detaylandırılmıştır; AWS üzerinde yapılan deneylerde cevap süreleri ortalama 66,061 ms, en düşük 19,477 ms ve en yüksek 209,227 ms olarak ölçülmüştür. GCP üzerinde yapılan deneylerde cevap süreleri ortalama 66,415 ms, en düşük 37,193 ms ve en yüksek 170,170 ms olarak ölçülmüştür. Azure üzerinde yapılan deneylerde cevap süreleri ortalama 96,577 ms, en düşük 34,813 ms ve en yüksek 485,759ms olarak ölçülmüştür. Ayrıca hata oranları başta verilen sıraya göre %27, %88,6, %60,5 olarak ölçülmüştür. Bu teste göre en iyi başırımı hata oranı ve ulaşım hızı olarak AWS senaryosu vermiştir. Oluşan farklılıkların sebebi mesajlaşma servislerinin tekrarlı mesaj isteme kısıtı, mesajların ulaşım ve sıralama hızı, ağ gecikmeleri gibi servis sağlayıcılara bağlı durumlar olabilmektedir. Hataların sebebi ise mesajların yerine ulaşmadan sorgu yapılması, yapılan sorguların sonuçlardan daha hızlı olması gibi sebepler olabilmektedir. Bu sonuçların

arasındaki farklılık, yapılan ölçüt panellerinde alınan ve gönderilen mesaj sayıları arasındaki fark şeklinde gözükebilmektedir. Bu durum sonuç kısmında değerlendirilmiştir.

4.2.2 Servislere Ait Ölçüt Durumları

Örüntüye ait olan iki farklı senaryodaki (görsel işleme ve getirme) her bir bulut sağlayıcısının (AWS,GCP ve Azure) mesaj servisleri izlenmiştir.

Amazon Web Servisleri

Olay Güdümlü örüntüyü kullanan sistem, AWS bulut sağlayıcısı üzerinde gerçekleştirilmiştir. Tablo 5'te sırasıyla gerçekleştirilen yük testlerinde AWS SQS servisinin iki farklı senaryodaki gönderilen, alınan ve silinen mesaj sayıları görülmektedir. Kullanıcı ve veri tabanın servislerinin benzer şekilde bu iki farklı senaryodaki ölçütleri EK-4'de sunulmuştur.

Tablo 5

AWS mesajlaşma servisi ölçüt durumları

Test Ucu	Alınan Mesaj Sayısı	Gönderilen Mesaj Sayısı	Silinen Mesaj Sayısı
Görsel İşleme	En Çok 333 / S	En Çok 500 / S	En Çok 329 / S
Görsel Sorgulama	En Çok 371 / S	En Çok 371 / S	En Çok 371 / S

Google Bulut Platformu

Olay Güdümlü örüntüyü kullanan sistem, GCP bulut sağlayıcısı üzerinde gerçekleştirilmiştir. Tablo 6'da sırasıyla gerçekleştirilen yük testlerinde GCP PUB/SUB servisinin iki farklı senaryodaki gönderilen, alınan ve silinen mesaj sayıları görülmektedir. Kullanıcı ve veri tabanın servislerinin benzer şekilde bu iki farklı senaryodaki ölçütleri EK-4'de sunulmuştur.

Tablo 6

GCP mesajlaşma servisi ölçüt durumları

Test Ucu	Alınan Mesaj Sayısı	Gönderilen Mesaj Sayısı	Silinen Mesaj Sayısı
Görsel İşleme	En Çok 500 / S	En Çok 276 / S	En Çok 500 / S
Görsel Sorgulama	En Çok 980 / S	En Çok 495 / S	En Çok 976 / S

Microsoft Azure

Olay Güdümlü örüntüyü kullanan sistem, Azure bulut sağlayıcısı üzerinde gerçekleştirilmiştir. Tablo 7'de sırasıyla gerçekleştirilen yük testlerinde Azure Service Bus servisinin iki farklı senaryodaki gönderilen, alınan ve silinen mesaj sayıları görülmektedir. Kullanıcı ve veri tabanın servislerinin benzer şekilde bu iki farklı senaryodaki ölçütleri EK-4'de sunulmuştur.

Tablo 7

Microsoft Azure mesajlaşma servisi ölçüt durumları

Test Ucu	**Alınan Mesaj Sayısı**	**Gönderilen Mesaj Sayısı**	**Silinen Mesaj Sayısı**
Görsel İşleme	En Çok 190 / S	En Çok 550 / S	En Çok 197 / S
Görsel Sorgulama	En Çok 200 / S	En Çok 557 / S	En Çok 200 / S

Yük testlerinde beklenildiği gibi bahsi geçen mesajlaşma kanallarına yollanan, bu kanallardan alınan ve silinen mesaj sayılarında yükselmeler görülmüştür. Fakat 500 istek gibi isteklerde bazı alınan ve gönderilen mesaj sayıları arasında tutarsızlıklar vardır. Bu bulgular incelendiğinde ölçütlerde durum anormal olarak adlandırılabilir. Yük testlerinde ortaya çıkan hatalar ve farklılıklar durumu sonuç kısmında değerlendirilmiştir.

BEŞİNCİ BÖLÜM
SONUÇ VE ÖNERİLER

Yapılan çalışmanın diğer çalışmalarla farklılığının ortaya konulabilmesi adına Tablo 8'de literatürde ele alınan çalışmaların farklı boyutları gösterilmiştir. Kullanılan örüntülerin eksikliği, bahsi geçen bulut servis sağlayıcılarının tamamının bir arada bulunmaması, SRE yaklaşımının vurgulanmaması farklılıkların en sık göze çarpanlarıdır. Bu bilgilerden yola çıkarak, yapılan çalışmada önceki çalışmalardan farklı bilimsel bakış açılarının ortaya konulduğu söylenebilir.

Tablo 8

Literatürde ele alınan çalışmaların farklı boyutları

Çalışmalar	**Örüntülerin Gerçeklenmesi**	**Kullanılan Bulut Servis Sağlayıcıları**	**İzleme Araçları**	**SRE Yaklaşımı**
Calero ve Aguado, 2015	YOK	AWS, AZURE	Varsayılan, 3. Taraf	YOK
Fatema vd., 2014	YOK	AWS, AZURE	Varsayılan, 3. Taraf	YOK
Absa vd., 2019	YOK	AWS	Varsayılan, 3. Taraf	YOK
Ward ve Barker, 2014	YOK	AWS	Varsayılan, 3. Taraf	VAR
Aceto vd. 2013	YOK	AWS, AZURE	Varsayılan, 3. Taraf	YOK

Tablo 8'in devamı

Grati vd., 2015	YOK	YOK	3. Taraf	YOK
Kumar vd., 2017	YOK	AWS, AZURE	Varsayılan, 3. Taraf	YOK
Birje ve Bulla, 2020	YOK	AWS, AZURE	Varsayılan, 3. Taraf	YOK
Kontsek vd. 2020	YOK	YOK	3. Taraf	VAR

Yapılan testlerden ve çalışma kapsamındaki bulut servis sağlayıcılarının sundukları belgelerden toplanan veriler çalışmada kullanılan servisler çerçevesinde Tablo 9'da gösterilmiştir. Çalışmada kullanılmayan Mysql, Sqlserver veri tabanları için olan ölçüt sayıları gibi kapsam dışında kalan veriler tabloya dahil edilmemiştir.

Tablo 9

Bulut servis sağlayıcılarının izleme araçlarının karşılaştırılması

Karşılaştırma Ölçütü	**AWS**	**GCP**	**Azure**
Sanal Makinalar İçin Sağlanan ölçüt sayısı	19	44	31
Mesajlaşma İçin Sağlanan ölçüt Sayısı	11	70	27
Veri Tabanı İçin Sağlanan ölçüt Sayısı	59	73	66
Uyarı Ulaşma Süresi	~1 Saniye	~1 Saniye	~1 Saniye
Ortalama Varsayılan Veri Toplama Frekansı	5 Dakika	1 Dakika	1 Dakika
Ortalama Özelleşmiş Veri Toplama Frekansı	1 Saniyeye Kadar	1 Saniyeye Kadar	1 Saniyeye Kadar

Tablo 9'un devamı

Maliyet	0.30 Dolar * Ölçüt / Ay	0.30 Dolar * Ölçüt / Ay	0.16 Dolar / 10 Milyon İşlem
Ölçüt Saklama Süresi	~15 Ay	~1.5 Ay	~3 Ay

Tabloda iki farklı mimaride kullanılan her parça için varsayılan ölçütler üzerinden karşılaştırmalar gösterilmiştir. Özelleşmiş ölçütleri kullanmak adına her sağlayıcıda farklı yaklaşımlar gösterildiğinden varsayılan ölçütler tercih edilmiş ve bu ölçütler üzerinden eşit bir karşılaştırma yapılmıştır. Bunun yanı sıra GCP için sağlayıcının doğrudan sağladığı ve kullanıcının ek bir kurulum yapmasına gerek olmayan etmen sistemi de kullanılmıştır. Buna benzer etmenlar AWS ve Azure kısmında kullanıcının ek bir kurulum yapmasını gerektirdiği için kapsam dışında olacağından karşılaştırmaya eklenmemiştir. Uyarı ulaşma süreleri, sağlayıcı üzerinde bir alarmın tetiklendiği andan mail olarak ulaştığı an arasında geçen süre olarak hesaplanmıştır. Veri toplama frekansı oluşturulan ölçüt panelleri üzerinde test edilmiştir ve sonuçlarda da görüldüğü şekilde işlenmiştir. Maliyet ise varsayılan servisler üzerinde sağlayıcıların belirttiği ücretsiz kısımdan sonra oluşan maliyet olarak işlenmiştir.

Daha önce yapılan çalışmalarda seçilen ana sağlayıcıların hepsinin bir arada bulunmaması ve seçilen karşılaştırma noktalarının SRE bakış açısıyla değerlendirilmemiş olması nedeni ile çalışma gerçekleştirilmiştir. Bahsi geçen bakış açısıyla bu verilere bakıldığında alınabilecek ölçüt sayısı bir SRE'nin oluşturması gereken servis düzeyleri adına daha detaylı ve daha etkili bir kurgu yapmasına olanak vermesi açısından önem sağlamaktadır. Yine aynı bakış açısından önetkin bir izleme yaklaşımı oluşturulması açısından uyarı ulaşım hızı çok büyük öneme sahiptir. Veri toplama frekansı ise detaylı bir izleme ortamı oluşturulması açısından ve bu verilerin olası bir hata için değerlendirilmesinde rol oynamaktadır. Maliyet kısmı ise her alanda olduğu gibi SRE alanında da sürdürülebilirlik açısından öneme sahiptir. Ölçüt saklama süreleri ise bir olay sonrasında geri dönülebilirlik ve olay toplantılarında çıkarımlar yapmak adına önem taşımaktadır.

Bu veriler göz önüne alındığında, izleme özelinde çalışmanın yapıldığı tarih itibari ile seçilen noktalarda GCP'nin daha tercih edilebilir olduğu söylenebilir. Ayrıca ölçüt panellerinin oluşturulması da izleme açısından önemli bir noktadır. Bu alanda da GCP hem kullanım kolaylığı açısından hem de Prometheus sorgu dili ile çoğu SRE'nin alıştığı şekilde panel oluşturabilmesi ile diğerlerinin önüne geçmektedir. Bununla beraber AWS de Azure'un önüne panel oluşturmada daha fazla özelleştirme seçeneği sunması nedeniyle geçmektedir.

Sonuçta yapılan çalışmalarda eğer yük testi raporları olmasa, alınan hataların seçilen izleme ölçütleri ile yakalanamayacağı görülmüştür. Raporlarda kırmızı gözüken ve hata alınan anlar ile izleme ölçüt panelleri karşılaştırıldığında panellerden hata alındığı gözükmemektedir. Bunun sebebi seçilen ölçütler içinde bir servisin uç noktalarından alınan hata oranının varsayılan mikroservis ölçütleri arasında bulunmamasıdır. Örneğin API Geçidi örüntüsü için veri tabanı ve API Geçidi servisi büyük önem taşımaktadır ama yük testinde seçilen ölçütler güvenli sınırlar arasında kalmıştır. Bu nedenle olası bir sorunu ortaya koyamamaktadır. Olay Güdümlü örüntü için mesajlaşma servisinde alınan ve gönderilen mesaj sayıları ölçütü hata alınan alanlarda göndermek istenilen mesaj sayısıyla uyum sağlamadığı için bir soruna işaret edebilmektedir.

Buradan yola çıkarak karşılaştırılan servis sağlayıcılarının verdiği servislerin sağlam ve yerine göre kullanışlı olduğu fakat tek başına yeterli olamayabileceği sonucuna ulaşılmıştır. Bu yetersizliği gidermek adına izleme kavramının yanında izlenebilirlik ve izlenebilir uygulama geliştirme kavramlarının da devreye girmesinin gerekliliğine yapılan testler neticesinde kanaat getirilmiştir. Bu değerlendirme yapılırken oluşturulan bu çalışmanın tamamen bağımsız olması adına maliyet, sadece varsayılan ölçütler kullanması ve oluşturulan yazılımın basit tutulması gibi kısıtları vardır.

İleriki çalışmalarda daha büyük ve tam mikroservis mimarileriyle beraber servis sağlayıcıların sunduğu sistemlerin yanına izleme alanındaki özgür yazılımlar, açık kaynaklı yazılımlar, sahipli yazılımlar eklenilmesi yine bahsi geçen noktalarda ve ilerisinde karşılaştırmaların yapılması alanın daha iyi anlaşılmasına ve kurgulanmasına katkı sağlayacağı düşünülmektedir.

KAYNAKÇA

Absa, S., Benedict, S., and Kumar, A. (2019). “Monitoring IaaS using various cloud monitors”. *Cluster Computing*, 22, 12459–12471. https://doi.org/10.1007/s10586-017-1657-y.

Aceto, G., Botta, A., Donato, W., and Pescapè, A. (2013). “Cloud monitoring: A survey”. *Computer Networks*, 57 (9), 2093-2115. https://doi.org/10.1016/j.comnet.2013.04.001.

Akbar, M. A., Mahmood, S., Alsalman, H., Razzaq, A., Gumaei, A. and Riaz, M. T. (2020). “Identification and prioritization of cloud based global software development best practices”. *IEEE Access*, 8, 191242-191262. https://doi.org/10.1109/ACCESS.2020.3031365.

Aksakalli, I. K., Celik, T., CAN, A. B., and Tekinerdogan, B. (2021). “Deployment and communication patterns in microservice architectures: A systematic literature review”. *Journal of Systems and Software*, 180, 0164-1212. https://doi.org/10.1016/j.js s.2021.11101

Apache, (t.y.). Apache Kafka. *Kafka protocol guide*. Erişim: 11 Ocak 2025, https://kafka.apache.org/protocol.

AWS., (2025a). AWS Docs. *Amazon SQS Documentation*. Erişim: 11 Ocak 2025, https://docs.aws.amazon.com/sqs.

AWS., (2025b). AWS Docs. *CloudWatch metrics for your instances.* Erişim: 11 Ocak 2025, https://docs.aws.amazon.com.

Birje, M. N. and Bulla, C. (2019). "Commercial and open source cloud monitoring tools: A review”. 2019 *International Conference on E-Business and Telecommunications*, 22–23 March 2019, Hyderabad, India, 480-490. https://doi.org/10.1007/978-3-030-24322-7_59.

Calero, J. M. A. and Aguado, J. G. (2015). “Comparative analysis of architectures for monitoring cloud computing infrastructures”. *Future Generation Computer Systems*, 47, 16-30. https://doi.org/10.1016/j.future.2014.12.008.

Dave, D. M. (2023). “Impact of site reliability engineering on manufacturing operations: Improving efficiency and reducing downtime”. I*nternational Journal of Scientific and Research Publications*, 13 (11), 136-138. https://doi.org/10.29322/IJSRP.13.11.2023.p14312.

Debroy, V., Mansoori, A., Haleblian, J., and Wilkens, M. (2020). “Challenges faced with application performance monitoring (APM) when migrating to the cloud”. *2020 IEEE International Symposium on Software Reliability Engineering Workshops,* 12-15 October 2020, Coimbra, Portugal, 153-154. https://doi.org/10.1109/ISSREW51248.2020.00046.

Dickerson, M. and Chen, T. Y. (2023). “Teaching site reliability engineering as a computer science elective”. *54. ACM Technical Symposium on Computer Science Education*, 15 - 18 March 2023, Toronto, Canada, 521–527. https://doi.org/10.1145/3545945.3569809.

European Commission, (2012). European Commission Policies. *Unleashing the potential of cloud computing in Europe*. Erişim: 16 Ekim 2024, https://www.consilium.europa.eu/en/policies/cloud-computing.

Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P. and Lynn, T. (2014). “A survey of cloud monitoring tools: Taxonomy, capabilities and objectives”. *Journal of Parallel and Distributed Computing*, 74, 2918-2933. https://doi.org/10.1016/j.jpdc.2014.06.007.

Fielding, R. (2000). Architectural styles and the design of network-based software architectures. Doctoral dissertation*,* University of California, United States.

Fowler, M. (t.y.). Martin Fowler Blog. *Microservices*. Erişim: 18 Ekim 2024, https://martinfowler.com/articles/microservices.html#MicroservicesAndSoa.

Gartner, (2022). Gartner Press Releases. *Gartner says more than half of enterprise IT spending in key market segments will shift to the cloud by 2025*. Erişim: 14 Ekim 2024, https://www.gartner.com/en/newsroom/press-releases/2022-02-09-gartner-says-more-than-half-of-enterprise-it-spending.

GCP., (2025a). Gcp Pub/Sub. *Architectural overview of Pub/Sub*. Erişim: 11 Ocak 2025, https://cloud.google.com/pubsub/architecture.

GCP., (2025b). GCP Monitoring. *Metrics list*. Erişim: 11 Ocak 2025, https://cloud.google.com/monitoring/api/metrics.

Gefen, D. (2002). "Customer loyalty in e-commerce". *Journal of the Association for Information Systems*, 3 (1), 27-51. https://doi.org/10.17705/1jais.00022.

Google (t.y.). Google Sre Book. *What is Site Reliability Engineering (SRE) ?*. Erişim: 3 Ekim 2024, https://sre.google.

Grati, R., Boukadi, K., and Ben-Abdallah, H. (2015). "Overview of IaaS monitoring tools". *12. International Conference of Computer Systems and Applications*, 17-20 November 2015, Marrakech, Morocco, 1-7. https://doi.org/10.1109/AICCSA.2015.7507146.

Jhingran, S., and Rakesh, N. (2023). "Performance analysis of microservices behavior in cloud vs containerized domain based on CPU utilization". *International Journal on Recent and Innovation Trends in Computing and Communication*, 11, 509-516. https://doi.org/10.17762/ijritcc.v11i6s.6959.

Julian, M. (2017). *Practical Monitoring: Effective Strategies for the Real World*. O'Reilly Media: United States.

Ker, D. (2021). "Measuring cloud services use by businesses". *OECD Digital Economy Papers 304*. https://doi.org/10.1787/71a0eb69-en.

Kontsek, M., Moravcik, M., Jurc, J., and Sterbak, M. (2020). "Survey of the monitoring tools suitable for CC environment". *18. International Conference on Emerging eLearning Technologies and Applications*, 12-13 November 2020, Kosice, Slovenia. 330-335. https://doi.org/10.1109/ICETA 51985.2020.9379152.

Kumar, P., Singh, P., Chopra, S., Sarna, J. S., and Rawat, K. (2017). “Inspection of cloud computing monitoring tools”. *2017 International Conference on Infocom Technologies and Unmanned Systems*, 18-20 December 2017, Dubai, United Arab Emirates. 361-365. https://doi.org/10.1109/ICTUS.2017.8286033.

Mell, P. and Grance, T. (2011). “The NIST definition of cloud computing”. *National Institute of Standards and Technology Special Publication 800-145*. https://doi.org/10.6028/NIST.SP.800-145.

Microsoft Azure, (2025a). Azure Service Bus Messaging. *AMQP 1.0 in Azure Service Bus and Event Hubs protocol guide.* Erişim: 11 Ocak 2025, https://learn.microsoft.com /en-us/azure/service-bus-messaging/service-bus-amqp-protocol-guide.

Microsoft Azure, (2025b). Azure Monitor. *Supported metrics with Azure Monitor.* Erişim: 11 Ocak 2025, https://learn.microsoft.com/en-us/azure/azure-monitor/refere nce/metrics-index.

Musa, J. D. (1980). “The measurement and management of software reliability”. *Proceedings of the IEEE*, 68 (9), 1131-1143. https://doi.org/10.1109/PROC.1980.11 812.

Musa, J. D. and Everett, W. W. (1990). “Software-reliability engineering. Technology for the 1990s”. *IEEE Software*, 7 (6), 36-43. https://doi.org/10.1109/52.60588.

Niknejad, N., Ismail, W., Ghani, I., Nazari, B., and Bahari, M. (2020). “Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation”. *Information Systems*, 91, 101491. https://doi.org/10.1016/j.is.2020.101491

Noviani, E. F., Kembara, B., Yudha Pratama, B. A., Permata Sari, D. A., Shiddiqi, A. M., and Santoso, B. J. (2022). “Performance analysis of AWS and GCP cloud providers”. *2022 IEEE International Conference on Cybernetics and Computational Intelligence*, 16-18 June 2022, Malang, Indonesia. 236-241. https://doi.org /10.1109/CyberneticsCom55287.2022.9865484.

Papadopoulos, A. V., Versluis, L., Bauer, A., Herbst, N., v. Kistowski, J., Ali-Eldin, A., Abad, C. L., Amaral, J. N., Tůma, P., Iosup, A. (2021). “Methodological principles for reproducible performance evaluation in cloud computing”. *IEEE Transactions on Software Engineering*, 47 (8), 1528-1543. https://doi.org/10.1109/TSE.2021.3051240.

RabbitMQ, (t.y.). RabbitMQ Docs. *Which protocols does RabbitMQ support ?*. Erişim: 11 Ocak 2025, https://www.rabbitmq.com/docs/protocols.

Saraswat, M., and Tripathi, R. C. (2020). “Cloud computing: Comparison and analysis of cloud service providers-AWS, Microsoft, and Google”. *9. International Conference System Modeling and Advancement in Research Trends*, 4-5 December 2020, Moradabad, India. 281-285. https://doi.org/10.1109/SMART50582.2020.9337100.

Scaleway, (t.y.). Scaleway Docs. *Load Testing with Vegeta*. Erişim: 11 Kasım 2024, https://www.scaleway.com/en/docs/tutorials/load-testing-vegeta.

Shamim, M. S. I., Gibson, J., Morrison, P., and Rahman, A. (2022). “Benefits, challenges, and research topics: A multi-vocal literature review of Kubernetes”. *ArXiv,* 2211, 07032. https://doi.org/10.48550/arXiv.2211.07032.

Silberschatz, A., Galvin, P. B., and Gagne, G. (2018). *Operating system concepts*. Wiley: United States.

Soldani, J., Tamburri, D., and Heuvel, W.-J. (2018). “The pains and gains of microservices: A systematic grey literature review”. *Journal of Systems and Software*, 146, 215-232. https://doi.org/10.1016/j.jss.2018.09.082.

Stallings, W. (2001). *Operating systems : internals and design principles*. Prentice Hall: United States.

Synergy Research Group (2023). *Cloud market spending growth report*. Erişim: 1 Ağustos 2024, https://www.srgresearch.com.

Tamburri, D., Miglierina, M., and Di Nitto, E. (2020). “Cloud applications monitoring: An industrial study”. *Information and Software Technology*, 127, 106376. https://doi.org/10.1016/j.infsof.2020.106376.

Tanadechopon, T., and Kasemsontitum, B. (2023). “Performance evaluation of programming languages as API services for cloud environments: A comparative study of PHP, Python, Node.js, and Golang”. *7. International Conference on Information Technology*, 16-17 November 2023, Thailand, 293-297. https://doi.org/10.1109/InCIT60207.2023.10413079

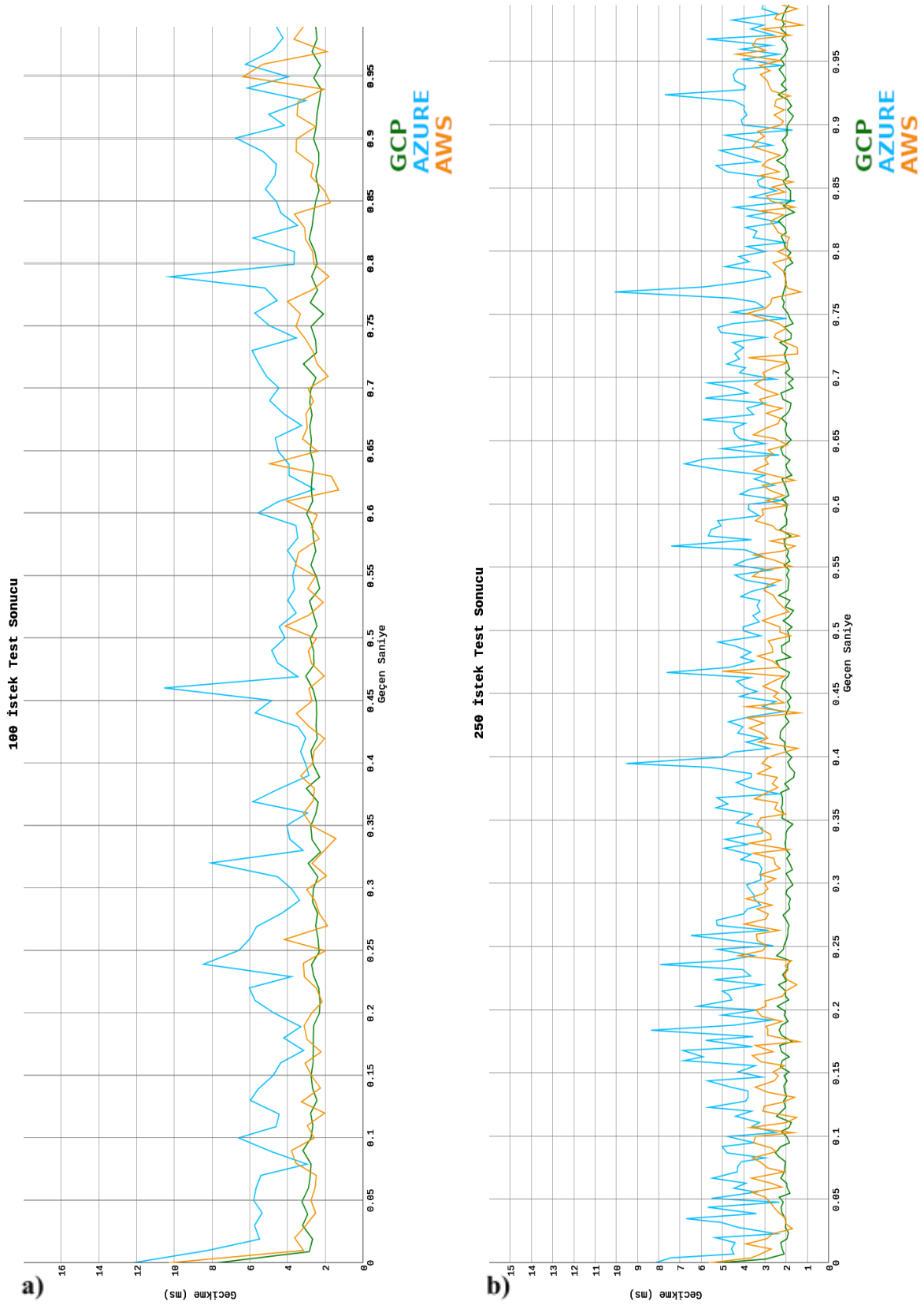
Tanenbaum, A. S. and Bos, H. J. (2015). *Modern Operating Systems.* Prentice Hall Press: United States.

Ward, J., and Barker, A. (2014). “Observing the clouds: A survey and taxonomy of cloud monitoring”. *Journal of Cloud Computing*, 3, 24. https://doi.org/10.1186/s13677-014-0024-2.

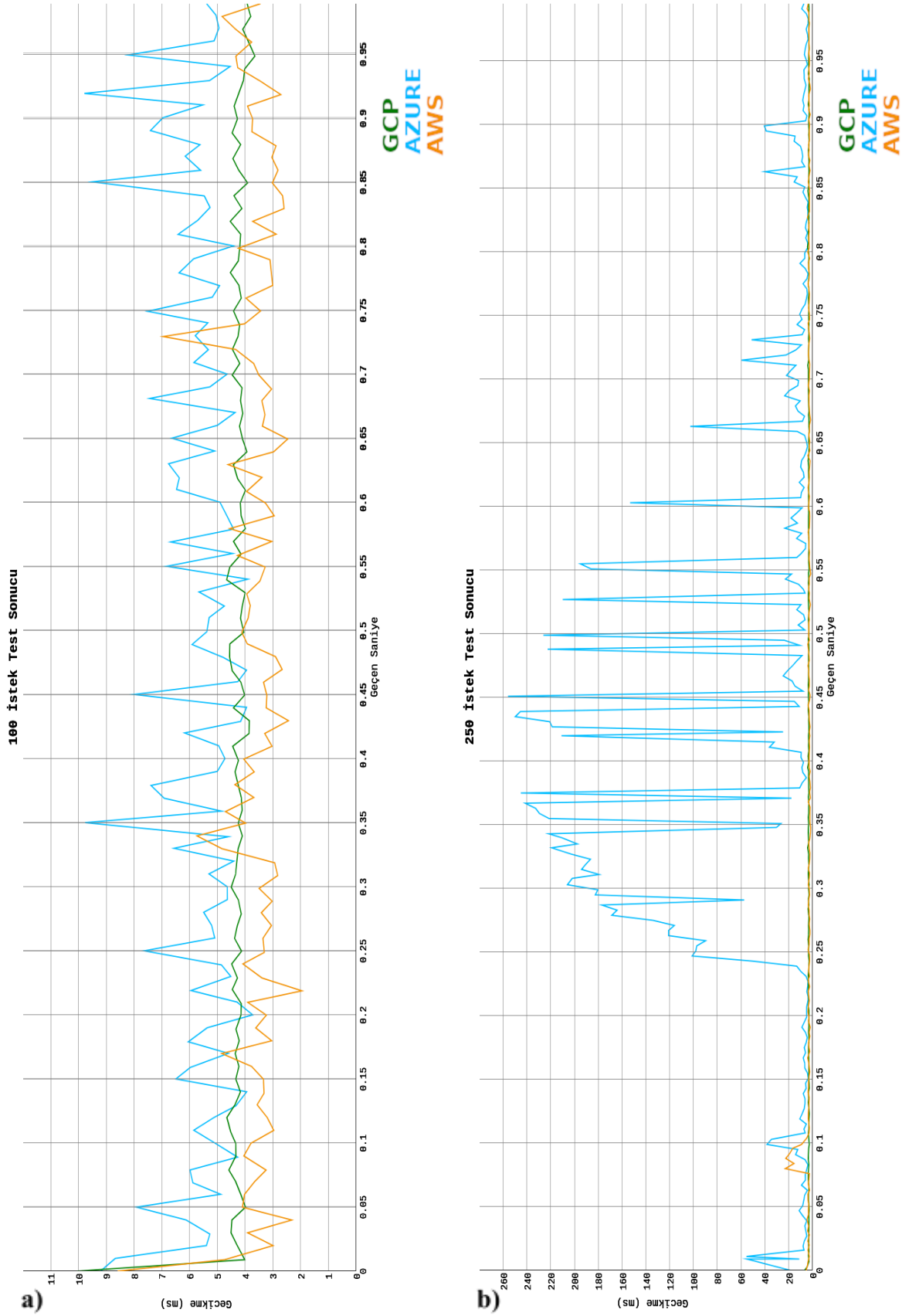
Wittig, M., Wittig, A., and Whaley, B. (2019). *Amazon Web Services in action*. Manning Publications: United States.

EKLER

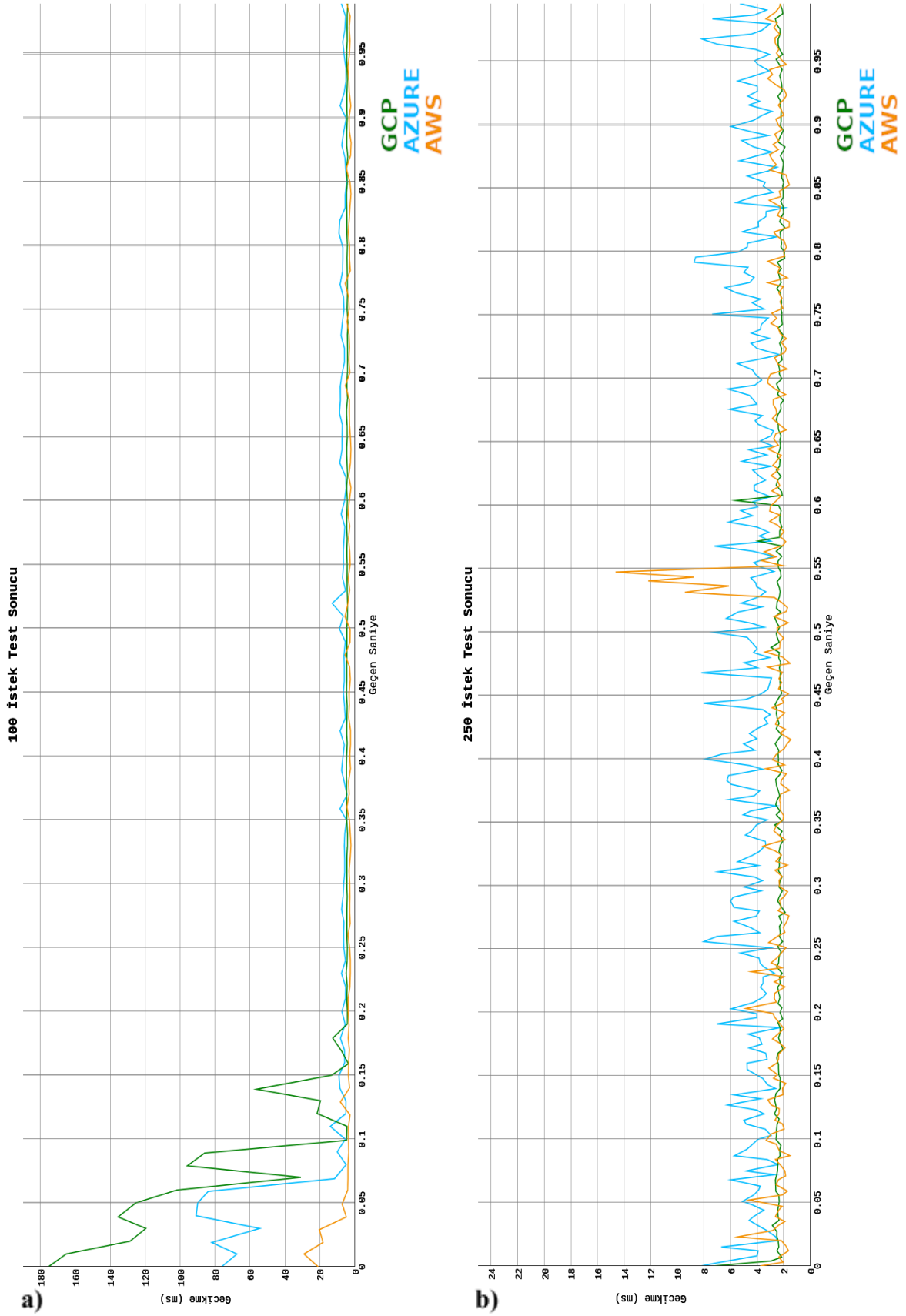
EK-1 API GEÇİDİ ÖRÜNTÜSÜ 100 VE 250 İSTEK YÜK TESTİ SONUÇLARI



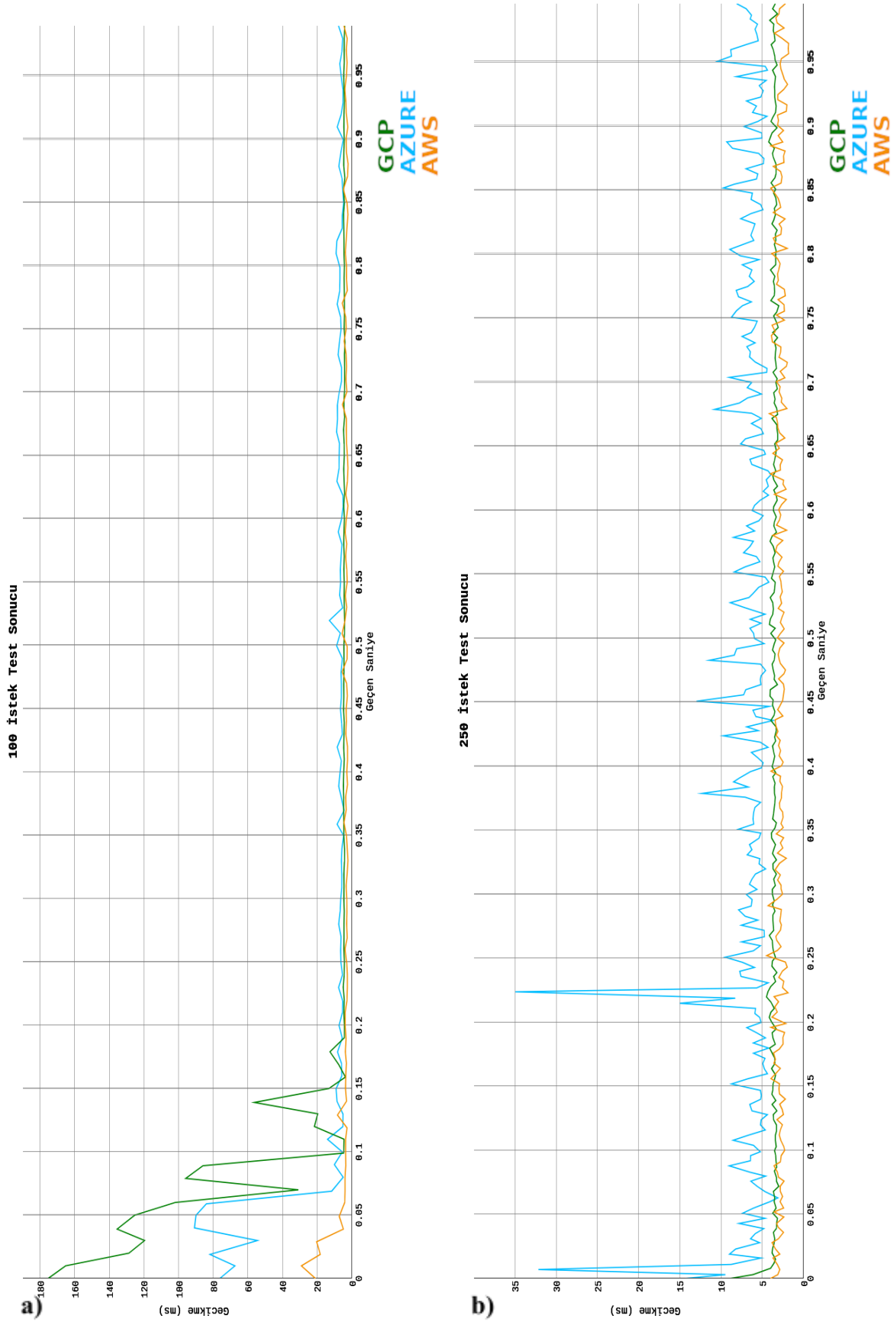
Ek Şekil 1. Kullanıcı sorgulama 100 istek (a) ve 250 istek (b) çıktıları.



Ek Şekil 2. Kullanıcı ekleme 100 istek (a) ve 250 istek (b) çıktıları.

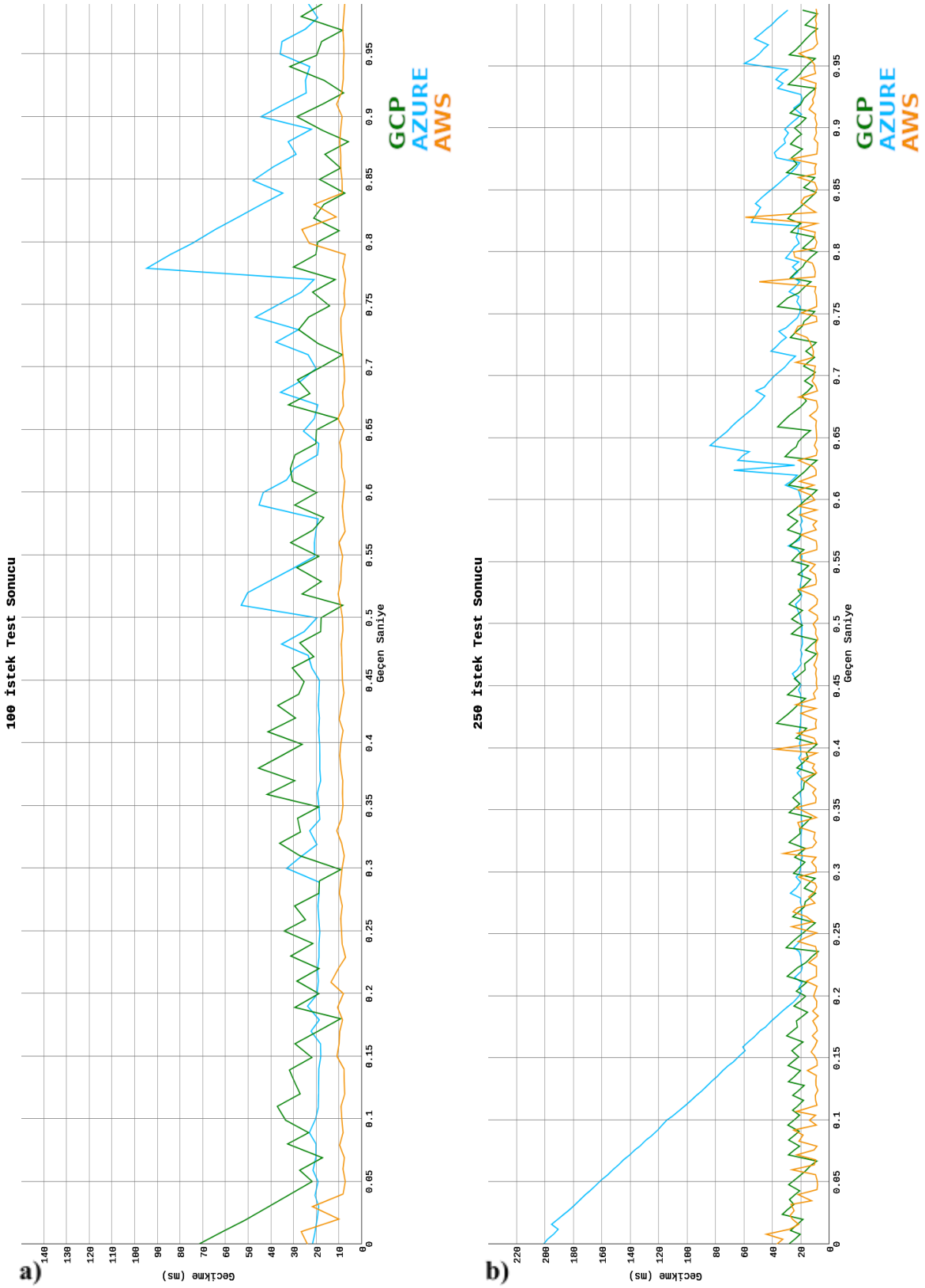


Ek Şekil 3. Ödeme sorgulama 100 istek (a) ve 250 istek (b) çıktıları.

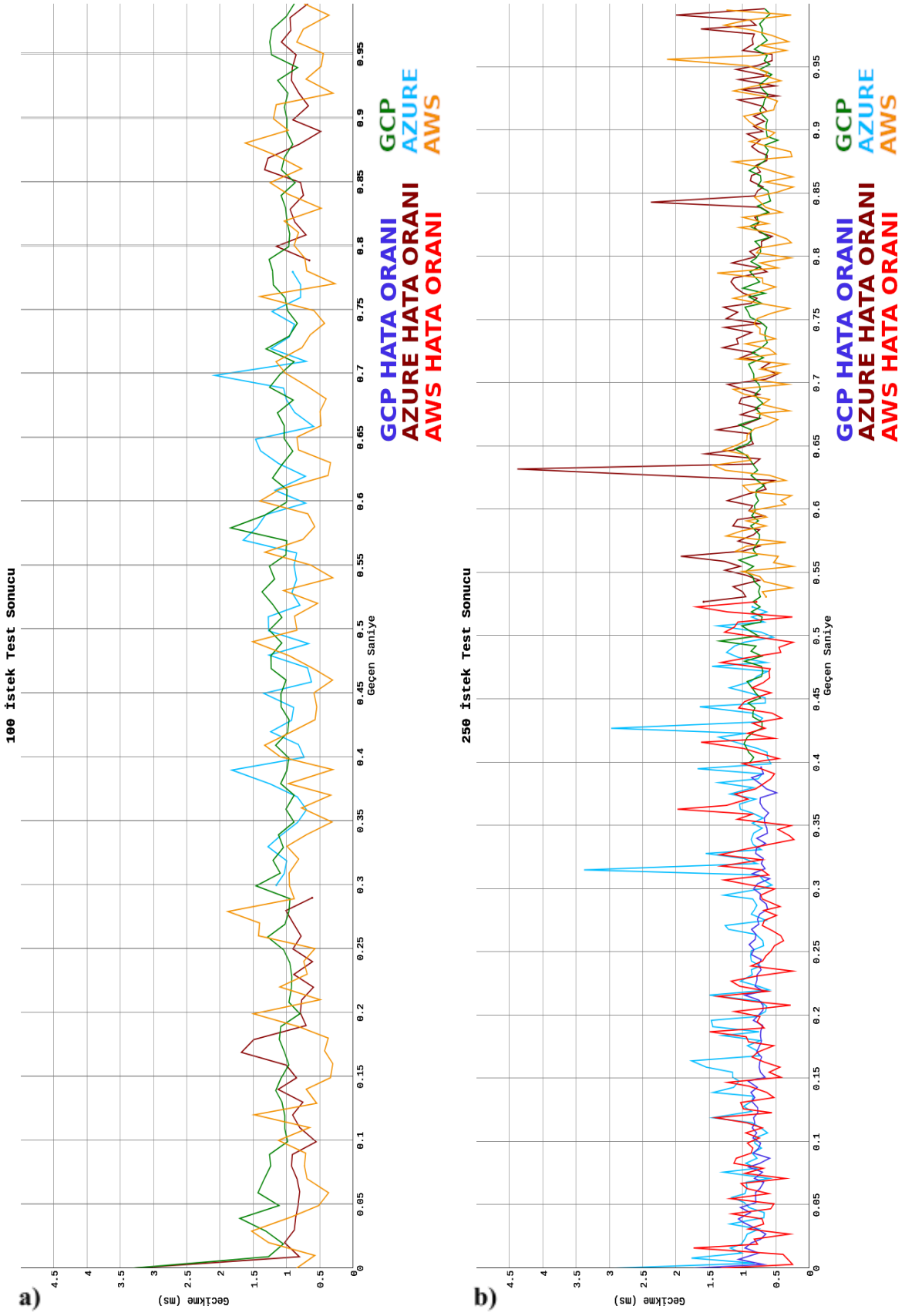


Ek Şekil 4. Ödeme ekleme 100 istek (a) ve 250 istek (b) çıktıları.

EK-2 OLAY GÜDÜMLÜ ÖRÜNTÜ 100 VE 250 İSTEK YÜK TESTİ SONUÇLARI



Ek Şekil 5. Görsel işleme 100 istek (a) ve 250 istek (b) çıktıları.



Ek Şekil 6. Görsel işlem sonucu sorgulama 100 istek (a) ve 250 istek (b) çıktıları.

EK-3 API GEÇİDİ ÖRÜNTÜSÜ ÖLÇÜT DURUMLARI

Ek Tablo 1

Kullanıcı servisi sanal makinesi ölçüt durumları

Test Ucu	İşlemci Kullanımı			Bellek Kullanımı			Depolama Kullanımı			Ağdan Gelen Veri			Ağa Giden Veri		
		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
Kullanıcı ekleme	AWS	%5	%6.73	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,05 MiB/S	0,178 MiB/S	AWS	0,05 MiB/S	0,164 MiB/S
	GCP	%5	%25	GCP	%10	%13	GCP	%25	%25	GCP	1 KiB/S	3 KiB/S	GCP	5 KiB/S	7 KiB/S
	AZR	%0,6	%1,6	AZR	YOK	YOK	AZR	YOK	YOK	AZR	48.83 KiB/S	890 KiB/S	AZR	45.33 KiB/S	785,27 KiB/S
Kullanıcı sorgulama		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
	AWS	%6	%6.36	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,05 MiB/S	0,084 MiB/S	AWS	0,03 MiB/S	0,079 MiB/S
	GCP	%10	%40	GCP	%10	%15	GCP	%25	%25	GCP	1 KiB/S	3,5 KiB/S	GCP	4 KiB/S	7,72 KiB/S
	AZR	%2	%6,5	AZR	YOK	YOK	AZR	YOK	YOK	AZR	46.83 KiB/S	878,91 KiB/S	AZR	43.33 KiB/S	781,52 KiB/S

Ek Tablo 2

Ödeme servisi sanal makinesi ölçüt durumları

Test Ucu	İşlemci Kullanımı			Bellek Kullanımı			Depolama Kullanımı			Ağdan Gelen Veri			Ağa Giden Veri		
		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
Ödeme ekleme	AWS	%0,1	%0.432	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,05 MiB/S	0,152 MiB/S	AWS	0,01 MiB/S	0,116 MiB/S
	GCP	%5	%50	GCP	%10	%17	GCP	%25	%25	GCP	1 KiB/S	8 KiB/S	GCP	5 KiB/S	9 KiB/S
	AZR	%0,5	%7	AZR	YOK	YOK	AZR	YOK	YOK	AZR	48.83 KiB/S	890 KiB/S	AZR	45.33 KiB/S	785,27 KiB/S
Ödeme sorgulama		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
	AWS	%0,1	%0,201	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,05 MiB/S	0,078 MiB/S	AWS	0,05 MiB/S	0,84 MiB/S
	GCP	%5	%45	GCP	%10	%20	GCP	%25	%25	GCP	1 KiB/S	15 KiB/S	GCP	5 KiB/S	17 KiB/S
	AZR	%0,6	%3,46	AZR	YOK	YOK	AZR	YOK	YOK	AZR	52.03 KiB/S	537,11 KiB/S	AZR	42.37 KiB/S	488,28 KiB/S

Ek Tablo 3

Veri tabanı servisleri ölçüt durumları

Test Ucu	Yazma İşlem Sayısı			Okuma İşlem Sayısı			Bağlantı Sayısı		
		En Az	En Çok		En Az	En Çok		En Az	En Çok
Kullanıcı ekleme	AWS	1/S	22/S	AWS	0/S	3/S	AWS	5	5
	GCP	1/S	25/S	GCP	0/S	1/S	GCP	5	8
	AZR	1/S	6/S	AZR	0/S	1/S	AZR	12	12
		En Az	En Çok		En Az	En Çok		En Az	En Çok
Kullanıcı sorgulama	AWS	1/S	10/S	AWS	1/S	1.3/S	AWS	5	5
	GCP	1/S	19/S	GCP	0/S	1/S	GCP	5	7
	AZR	1/S	1/S	AZR	0/S	1/S	AZR	11	12
		En Az	En Çok		En Az	En Çok		En Az	En Çok
Ödeme ekleme	AWS	1/S	20/S	AWS	1/S	1,7/S	AWS	5	6
	GCP	1/S	9/S	GCP	0/S	0,5/S	GCP	7	9
	AZR	1/S	1/S	AZR	0/S	0,5/S	AZR	12	17
		En Az	En Çok		En Az	En Çok		En Az	En Çok
Ödeme sorgulama	AWS	1/S	10/S	AWS	1/S	1.3/S	AWS	6	6
	GCP	1/S	2/S	GCP	0/S	0,5/S	GCP	10	14
	AZR	1/S	1/S	AZR	0/S	0,5S	AZR	12	13

EK-4 OLAY GÜDÜMLÜ ÖRÜNTÜ GÖSTERGE PANELİ DURUMLARI

Ek Tablo 4

Görüntü işleme servisi sanal makinesi ölçüt durumları

Test Ucu	İşlemci Kullanımı			Bellek Kullanımı			Depolama Kullanımı			Ağdan Gelen Veri			Ağa Giden Veri		
		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
Görsel İşleme	AWS	%4	%5,6	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,05 MiB/S	1,67 MiB/S	AWS	0,05 MiB/S	1,37 MiB/S
	GCP	%10	%35	GCP	%15	%20	GCP	%25	%25	GCP	1 KiB/S	9 KiB/S	GCP	10 KiB/S	17 KiB/S
	AZR	%0,6	%2,5	AZR	YOK	YOK	AZR	YOK	YOK	AZR	45.96 KiB/S	292,97 KiB/S	AZR	45.33 KiB/S	341,8 KiB/S
		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
Görsel İşlem Sonucu Sorgulama	AWS	%4	%6.06	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,05 MiB/S	1,72 MiB/S	AWS	0,05 MiB/S	0,164 MiB/S
	GCP	%5	%38	GCP	%15	%20	GCP	%25	%25	GCP	1 KiB/S	11 KiB/S	GCP	5 KiB/S	7 KiB/S
	AZR	%0,6	%2,4	AZR	YOK	YOK	AZR	YOK	YOK	AZR	48.83 KiB/S	291 KiB/S	AZR	46.77 KiB/S	340,27 KiB/S

Ek Tablo 5

Görüntü işleyici sanal makinesi ölçüt durumları

Test Ucu	İşlemci Kullanımı			Bellek Kullanımı			Depolama Kullanımı			Ağdan Gelen Veri			Ağa Giden Veri		
		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok		En Az	En Çok
Görsel İşleme	AWS	%5	%6.55	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,152 MiB/S	0,652 MiB/S	AWS	0,220 MiB/S	0,811 MiB/S
	GCP	%5	%50	GCP	%15	%22	GCP	%25	%25	GCP	1 KiB/S	4 KiB/S	GCP	5 KiB/S	15 KiB/S
	AZR	%0,5	%6,95	AZR	YOK	YOK	AZR	YOK	YOK	AZR	42.83 KiB/S	1,5 MiB/S	AZR	45.33 KiB/S	165,25 KiB/S
Görsel İşlem Sonucu Sorgulama	AWS	%5	%6.63	AWS	YOK	YOK	AWS	YOK	YOK	AWS	0,1 MiB/S	0,644 MiB/S	AWS	0,1 MiB/S	0,808 MiB/S
	GCP	%5	%50	GCP	%17	%20	GCP	%25	%25	GCP	1 KiB/S	6 KiB/S	GCP	5 KiB/S	18 KiB/S
	AZR	%0,7	%7	AZR	YOK	YOK	AZR	YOK	YOK	AZR	35 KiB/S	1,53 MiB/S	AZR	58.59 KiB/S	175,78 KiB/S