Fırat Üniversitesi Müh. Bil. Dergisi
34(2), 769-778, 2022

Araştırma Makalesi
https://doi.org/10.35234/fumbd.1146700

Uzaktan Hasta Takibi İçin Mikroservis Mimarisi Kullanan Bir Uç Sistem Tasarımı

Sinan TAŞLI[1*], Güngör YILDIRIM[2]
[1] Bilgisayar Mühendisliği, Mühendislik Mimarlık Fakültesi, Batman Üniversitesi, Batman, Türkiye
[2] Bilgisayar Mühendisliği, Mühendislik Fakültesi, Fırat Üniversitesi, Elazığ, Türkiye
[*1] sinan.tasli@batman.edu.tr, [2] gungor.yildirim@firat.edu.tr

(Geliş/Received: 21/07/2022; Kabul/Accepted: 24/08/2022)

Öz: Gecikme ve sistem yükü, bulut sistemler için iki kritik konudur. Bu konular akıllı şehir, akıllı sağlık sistemleri gibi büyük projelerde daha da önem kazanmaktadır. Son yıllarda kenar/uç ve sis gibi bulut teknolojileri bu iki kritik konuda önemli kazanımlar sağlamayı başarmıştır. Ancak bununla birlikte bu tür sistemlerde veri iletişimi, analiz ve değerlendirme gibi işlemlerin nasıl sağlanacağı da iyi planlanmalıdır. Bu tip bulut teknolojilerinde monolotik yazılım mimarileri yerine mikroservis temelli mimarilerin tercih edilmesi daha esnek çözümler sağlayabilmektedir. Bu çalışmada mikroservis temelli uç bulut teknolojisi alt yapısı kullanan bir hasta takip sistemi önerilmektedir. Önerilen sistem sayesinde, kritik durumlu hastaların takip edilmesi ve hastada gerçekleşebilecek acil durumlar hiyerarşik bir şekilde değerlendirilebilmektedir. Önerilen sistemin özgün yanı, kullanılan sistemin devre kesici mekanizması kullanan mikroservis yazılım mimarisini kullanması ve tüm mikroservisleri konteyner alt yapısı ile kontrol edebilmesidir. Bu özellikleri sayesinde sistem yükü ve cevap gecikmesinde önemli iyileştirmeler elde edilmiştir.

Anahtar kelimeler: Mikroservis, bulut teknolojisi, uç sistem, sağlık.

An Edge System Design Using Microservice Architecture for Remote Patient Monitoring

Abstract: Latency and system load are two critical issues for cloud systems. These issues become more important in large projects such as smart cities and smart health systems. In recent years, cloud technologies such as edge and fog (fog) systems have achieved significant gains in these two critical issues. However, it should be well planned how to provide data communication, analysis, and evaluation in such systems. In these types of cloud technologies, choosing microservice-based software architectures instead of monolithic architectures can provide more flexible solutions. In this study, a patient monitoring system using edge cloud technology infrastructure based on microservice is proposed. Thanks to the proposed system, patients with critical conditions can be followed up and emergency situations that may occur in the patient can be evaluated in a hierarchical manner. The originality of the proposed system is that it uses microservice software architecture using a circuit breaker mechanism and can control all microservices with container infrastructure. With these features, significant improvements in system load and response delay have been achieved.

Key words: Microservice, Cloud technology, edge system, health.

1. Giriş

Nesnelerin İnterneti (Nİ/IoT) teknolojileri [1] sayesinde günümüzde birçok büyük ölçekli akıllı sistem (akıllı şehir, akıllı sağlık, akıllı enerji sistemleri, vb.) insanlığa hizmet vermektedir. Bu büyük ölçekli IoT sistemler, çok fazla sayıda heterojen sayısal aygıtın haberleşmesini ve ortaya çıkabilecek muazzam büyüklükte verinin işlenmesini gerçekleştirmektedir. Bulut teknolojileri [2] bu bakımdan bu tip büyük ölçekli IoT projelerin önemli oranda yükünü almaktadır. Ancak öte yandan bulut sistemlerin son kullanıcıya uzak olması, kamuya açık ticari versiyonlarında (public cloud) kullandığın kadar öde sistemi, bulut kaynaklarının yüklenme problemi gibi etkenler hem bulut sağlayıcı hem de son kullanıcı açısından kritik olabilmektedir. Son yıllarda kullanılan uç (edge) ve sis (fog) bulut teknolojileri bu tür kritik problemlerin çözümünde ciddi performans artışı sağlamayı başarmıştır[3,4]. Bu sistemler son kullanıcı ile bulut sitem arasında yer alırlar ve hangi işlemlerin bulutta hangi işlemlerin kendi üzerlerinde yapılacağına karar verirler. Filtreleme, kısıtlı kaynaklar ile yapılabilen algoritmik analizler bu sistemler üzerinde yürütülebilir. Bu çalışmanın da esas aldığı uç sistemlerin sis sistemlerden temel farkı, uç sistemlerin nispeten daha kısıtlı kaynaklara sahip olması ve son kullanıcıya daha yakın konumda bulunmasıdır. Özellikle son kullanıcıya daha yakın olması cevap sürelerinin daha kısa olmasına olanak sağlamaktadır. Uç sistemlerde kaynakların sınırlı olması bir şekilde sis çözüme göre onları daha da ekonomik kılmaktadır.

[*] Sorumlu yazar: sinan.tasli@batman.edu.tr. Yazarların ORCID Numarası: [1] 0000-0002-3760-7872, [2] 0000-0002-4096-4838

Uç sistemlerin sahip olduğu kısıtlı kaynak yapısı, doğal olarak, kullanılacak yazılım mimarisinin de doğru seçilmesini gerektirmektedir. Günümüzde Monolit Mimari ve Servis Odaklı Mimari (SOA) olmak üzere iki yaygın yazılım mimarisi bulunmaktadır [5]. Monolit mimaride yazılım bileşenleri tek proses içinde çalışır ve genellikle tek bir platformda geliştirilirler. Günümüzde monolit mimariler, hizmet olarak yazılım (software as a service-SaaS) projeleri, java WAR dosyaları gibi sık kullanılan uygulamaların temelini oluştururlar. Bir monolit yazılımın bileşenleri arasında iletişim basittir, tek proses içerisinde çalışmalarından dolayı paylaşımlı hafıza maliyeti düşüktür ve bir ağ iletişimine gerek yoktur. Ancak öte yandan yazılım bileşenlerinde meydana gelecek problem prosesin tüm çalışmasını etkileyecektir. Ayrıca platform bağımlı olması, kodların büyümesi ile karmaşıklığın artması, güncelleme problemleri monolit uygulamaların geliştirme süreçlerinin maliyetini arttırmaktadır. Proses temelli olmalarından dolayı diğer uygulama prosesleri ile prosesler arası(inter-process) iletişime ihtiyaç duyarlar. Bu dezavantajlarından dolayı monolit mimari, genellikle internet temelli dağıtık sistem uygulamaları için çok uygun değildirler. Kullanılacak yazılımın bileşenlerinin birbirleri ile internet üzerinden haberleşebilen daha küçük modüllere ayrılması onları dağıtık uygulamalar için daha verimli yapar. Böylece her bir yazılım bileşeni farklı platformlarda geliştirilebilir ve bu bileşenler yeniden kullanılabilir. Bu tip yazılım mimari yaklaşımı SOA'nın temel felsefesini oluşturmaktadır. SOA çözümler heterojen uygulamalara, yatay ölçeklemeye ve servis paylaşımına olanak sağlayabilir. Aslında SOA eski bir fikir olsa da temel mimarilerden biri olmayı 2000'li yılların başlarında başarmıştır. SOA'da yazılım bileşenleri bir endpoint üzerinden hizmet sunar ve bu hizmetlere bir ESB (Enterprise Service Bus) üzerinden ulaşılır [6,7]. SOA uygulamalarda ESB'ler ana omurgayı meydana getirmektedir. ESB'ler yetkilendirme ve yönlendirme gibi temel fonksiyonları yürütür. SOA uygulamalarının altyapısını SOAP, HTTP protokolleri ve WSDL(Web Services Description Language) teknolojisi oluşturmaktadır. Sağladığı esnekliğin yanı sıra SOA'nın bazı dezavantajları da bulunmaktadır. Bu dezavantajlarının başında ESB'lerin geliştirilmesinin ve sürdürülebilirliklerinin oldukça maliyetli olması gelmektedir. Bu nedenle ESB'ler genellikle büyük ölçekli yazılım firmaları (IBM, Oracle etc.) tarafından geliştirilir ve yürütülür. SOA uygulamalarda ESB'den dolayı gecikme süresi fazladır ve ayrıca kısıtlı geliştirme araçları geliştiriciler açısından bir sorundur.

Mikroservis yazılım mimarileri monolit ve SOA mimarinin güçlü yönlerini bir araya getirir[8]. Mikroservis mimariler aslında bir SOA tipi olarak değerlendirilir. SOA'dan en önemli farkı ise bir ESB mekanizmasına ihtiyaç duymamasıdır. Mikroservis temelli uygulamalarda servis iletişimleri, REST API [9], GraphQL [10], ve gRPC [11] gibi günümüzün yaygın iletişim protokolleri ile sağlanabilmektedir. Mikroservis mimaride senkron ve asenkron iletişim teknikleri kullanılabilir ve geliştirilen mikroservisler konteyner teknolojileri [12] ile yönetilebilir. Mikroservisler nispeten küçük yazılımlar olduğu için kısıtlı kaynaklı cihazlar üzerinde çalıştırılmaları kolaydır. Bu özellikleri sayesinde uç sistemler için mikroservis temelli yazılımların kullanılması pek çok yönden esneklik sağlayacaktır. Öte yandan mikroservis mimarisinin tipi, iletişim tekniği, sürdürülebilirliği gibi kritik planlamalar, problem tipine göre tasarım aşamasında iyi düşünülmelidir. Bu çalışma, son yılların önemli IoT uygulamalarından olan uzaktan hasta takip projeleri için mikroservis temelli bir uç sistem tasarımını önermektedir. Önerilen sistem özellikle aile hekimlikleri ve onların sorumluluklarında olan hastaların izlenmesine yöneliktir. Aile hekimlikleri aynı zamanda genel sağlık sistemi (bakanlık gibi) ile bulut üzerinden hizmet alabilmektedir. Aile hekimliklerinde yapılamayan değerlendirmeler veya depolanamayan veriler için bulut servisleri kullanılmaktadır. Uç cihazlar ve mikroservisler aile hekimliklerinde ve hasta lokasyonunda bulunmaktadır. Mikroservis kurulum, yürütüm ve kontrolleri konteyner teknolojileri ile sağlanmaktadır. İletişim için REST API teknolojilerini kullanan bu sistem aynı zamanda hayati fonksiyonlara sahip mikroservis yürütümlerinin sekteye uğramaması için devre-kesici mekanizmalarda kullanmaktadır. Yazarların bildiği kadarıyla, sağlık alanında devre kesici mekanizması ve konteyner-bulut teknolojilerini birlikte kullanabilen bu tip bir sistem daha önce önerilmemiştir.

Makalenin bundan sonraki bölümleri şu şekildedir; İkinci bölüm problem tanımını ve literatür özetini içermektedir. Önerilen sistemin detayları üçüncü bölümde verilmektedir. Dördüncü bölümde yapılan deneyler ve değerlendirmeler paylaşılmış ve sonuçlar ise beşinci bölümde tartışılmıştır.

2. Literatür Özeti ve Problemin Tanımı

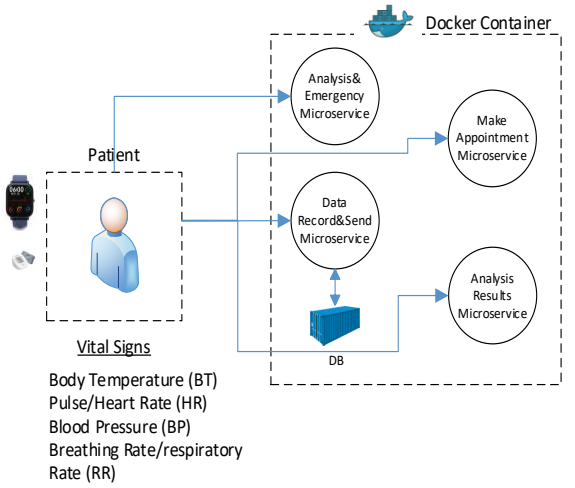
IoT temelli sağlık uygulamaları akademik ve endüstri dünyasının ilgi çekici konuları arasındandır. Kumar vd. tarafından kısıtlı kaynaklı SoC aygıtlar ile geliştirilen IoT tabanlı hasta izleme sistemi bu tür çalışmaların ilkleri arasındadır [13]. Punit vd. yoğun bakım üniteleri gibi acil tıbbi hizmetlere destek sağlamak amacıyla IoT tabanlı hasta izleme sistemini önermiştir [14]. Önerilen sistemle, hasta verileri gerçek zamanlı toplanarak analiz edilirken ayrıca bazı kronik/yaşlı hastalarda bilinç kaybı veya kriz durumlarına bağlı olarak gerçekleşen düşmelerin tespitleri de yapılabilmiştir. Düşme tespiti veya bunun sonucunda oluşabilecek hayati durumların izlenmesine yönelik sistem tasarımları ayrıca [15] ve [16]'daki çalışmalarda da önerilmiştir. Sağlık uygulamalarında Elektronik

Sağlık Kayıtları (ESK\EHR)'nın önemi kritiktir [17]. Dünya genelinde veya kitlesel olarak yaşanan salgın hastalıklar, hasta mahremiyeti ve istatistikleri ESK'nin gerekliliğini ortaya koymaktadır. Son yıllarda genotip bilgilerini ve hasta mahremiyeti detaylarını gözeten IoT sağlık sistemleri tanıtılmıştır [17,18]. Ayrıca, giyilebilir ve diğer IoT cihazları, Kişisel Sağlık Kayıtları (KSK\PHR) ile ilgili olanlar da dahil olmak üzere ilgili bilgileri toplayabilir ve ESK sistemlerine yükleyebilir [19]. Bununla birlikte, kurumlar arasında ESK verilerinin birlikte çalışabilirliğinin olmaması, sağlık ve klinik araştırma verilerinin bütünleştirilmesini ve paylaşılmasını zorlaştırır ve bu durum etkili ve verimli işbirliğini engeller [20]. Sağlık kuruluşlarının ortak bir veri standardı kullanması bu durumun önüne geçebilir. FHIR(Fast Healthcare Interoperability Resources), HL7 (Health Level Seven) tarafından sağlık alanında kullanılmak için geliştirilmiş bir veri standardıdır [21]. HL7, elliden fazla ülkede üyesi olan uluslararası bir kuruluştur ve sağlıkta birlikte çalışabilirliği sağlamak amacıyla FHIR projesini geliştirmektedir. Mevcut FHIR çözümleri, tam anlamıyla sağlık alanındaki birlikte çalışabilirlik problemlerini ortadan kaldırmasa bile önemli katkılar sağlamaktadır. Mikroservis temelli sağlık sistemleri de son yıllarda pek çok sistemde kendini göstermeye başlamıştır. [22] yaşlı hastaları takibi için mikroservis temelli bir IoT sistemini tasarlamış, [23] ise mikroservis alt yapısı kullanan bir bioenformatik platformunu tanıtmıştır. Bununla birlikte Aile Hekimlikleri gibi alt sağlık birimlerinin hasta takibini ve ana sağlık sistemi ile entegrasyonu mikroservis temelli ve bulut altyapısı üzerinden sağlayan net bir sistem tasarımı yazarların bildiği kadarıyla henüz önerilmemiştir.

Sağlık kuruluşları işletim sürecinde meydana gelen aksamaları azaltmak, hasta katılımlarını iyileştirmek ve hızlı karar mekanizmaları geliştirmek için sürdürülebilir IT alt yapılarına ihtiyaç duyarlar. Uzun yıllardır kullanılan büyük bir sağlık sisteminin doğrudan işletim modelini değiştirmek beraberinde öngörülemez problemleri getirebilir. Bu nedenle kısmi özerk alt birimlerin (aile hekimlikleri ve onların kapsama alanları gibi) IoT alt yapılarının iyileştirilmesi ve bulut teknolojileri aracılığı ile ana sağlık sisteme dâhil edilmesi daha sorunsuz bir geçiş süreci sağlayabilir. Ancak bununla birlikte bu tür IoT alt sistemleri, iki temel kriteri sağlamak durumundadır. Bunlardan birincisi bulut yükünü çok arttırmamalı ikincisi ise ekonomik olmalıdır. Her iki kriteri yerine getirmede mikroservis temelli uç sistemler önemli avantajlara sahip olacaktır. Konteyner altyapısı ile geliştirilen alt birim mikroservisleri ana sağlık sistemi tarafından kolaylıkla yönetilebilir. Dahası bu mikroservisler kısıtlı kaynaklı uç aygıtlarda (Raspberry Pi, Odroid N2, etc.) optimum maliyetle çalıştırılabilir.

3. Önerilen Sistem Mimarisi

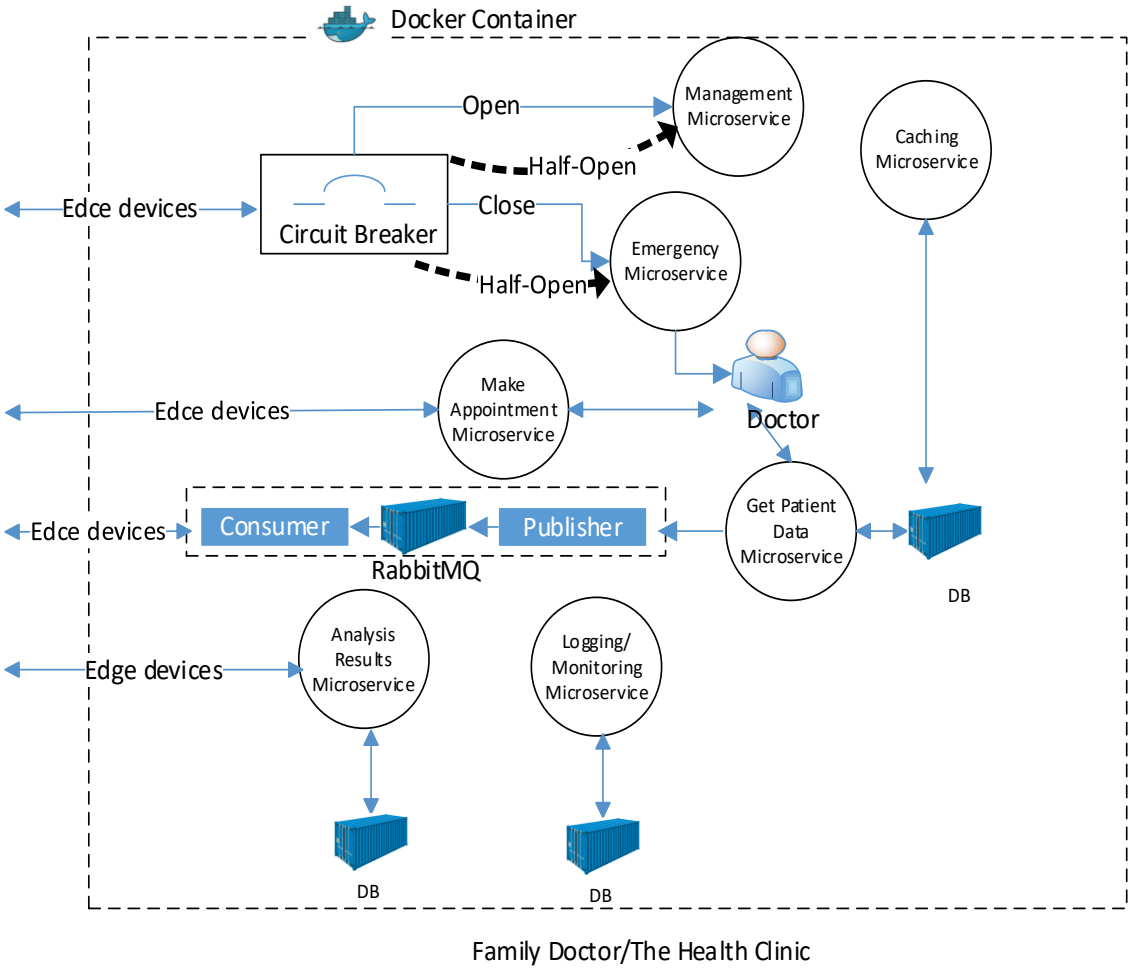
Önerilen sistem, aile hekimlikleri kapsamında bulunan acil durumlu hastaların uzaktan takip edilebilmesine yönelik tasarlanmıştır [24]. Sistem üç ana üniteden meydana gelmektedir. Bunlar, Hasta Uç Ünitesi (HUÜ), Aile Hekimliği Uç Ünitesi (AHUÜ) ve Bulut Ünitesidir (BÜ). HUÜ, gözlemlenecek olan hasta lokasyonunda bulunur ve kısıtlı kaynaklı bir donanım üzerinde çalışır. Şekil 1'den görüleceği üzere, HUÜ'de toplamda 4 adet mikroservis çalışmaktadır. Bu mikroservisler, Analysis&Emergency (Tahlil ve Acil Vaka), Data Record&Send (Veri Kayıt ve Gönderme), Analysis Result (Tahlil Sonuçları) ve Make Appointment (Randevu) mikroservisleridir. Hastadan alınan veriler Analysis&Emergency ve Data Record&Send mikroservislerine gönderilmektedir. Data Record&Send mikroservisinin görevi hastadan alınan vital bulgu (Ateş, Nabız, Tansiyon ve Solunum Sayısı) verilerini veri tabanına kaydetmek ve aile hekiminin gerek duyduğu zamanlarda verileri paylaşmaktır. Analysis&Emergency mikroservisine gelen veriler hastanın yaşı ve durumu itibariyle olması gereken normal değerler ile karşılaştırılmaktadır. Hastanın vital bulgu değerlerinden herhangi birinin hasta üzerinde risk oluşturabilecek duruma ulaşması halinde, hastanın aile hekimine ve gerekli sağlık kuruluşlarına bildirim gönderilmektedir. HUÜ'de ayrıca hastanın tahlil sonuçları, randevu alma ve bunları sorgulama isteklerine cevap verecek mikroservisler de bulunmaktadır. Hastanın bu isteklerini karşılamak için HUÜ üzerinde Make Appointment adında bir randevu alma mikroservisi ve Analysis Result adında da tahlil sonuçlarını görebileceği bir mikroservis bulunmaktadır. Hasta randevu almak istediği tarihi ve saati Make Appointment mikroservisi aracılığıyla aile hekimliğine bildirmektedir. Aile hekimliği tarafındaki aynı isimdeki mikroservis hastanın randevu için istediği tarih ve saati kontrol ederek randevunun uygun olup olmadığı bilgisini hastaya dönmektedir. Tahlil sonuçlarına, hastanın TC bilgisine göre, ilgili mikroservis üzerinden ulaşılabilmektedir. Mikroservislerin kurulumu, sürdümü ve yönetimi için Docker konteyner teknolojisi kullanılmaktadır. Konteyner kontrol ve koordinasyonu için Docker Swarm kullanılmıştır. Docker Swarm sayesinde mikroservislerin izlenmesi ve kontrolü sağlanmaktadır. Herhangi bir mikroservise ait konteynerin çalışamaz duruma gelmesi halinde Docker Swarm çalışamaz durumdaki konteynerden yeni bir tane oluşturarak sistemin devamlılığı konusunda büyük kolaylıklar sağlamaktadır. Bu tip sistemlerde asenkron çalışma için kullanılacak temel mekanizmalar, Gelişmiş Mesaj Kuyruklama Protokolü (Advanced Message Queuing Protocol-AMQP) veya Veri Dağıtım Servisleri (Data Distribution Service-DDS) temelli teknolojiler ile sağlanabilir. Önerilen sistemde uç cihaz konfigürasyonlarının yeterli olmasından dolayı bir AMQP implementasyonu olan RabitMQ tercih edilmiştir.




```python
@app.route('/select/<int:tc>', methods=['GET'])
def get_users(tc):
    sorgu='SELECT * FROM Analysis_Results WHERE tc = %s',(tc,)
    sonuc = baglanti.execute(sorgu)
    kullanicilar = baglanti.fetchall()
    for i in kullanicilar:
        print(i)
    if (sonuc==1):
        return jsonify(kullanicilar)
    else:
        return 'Sonucunuz henuz cıkmadı !'
```

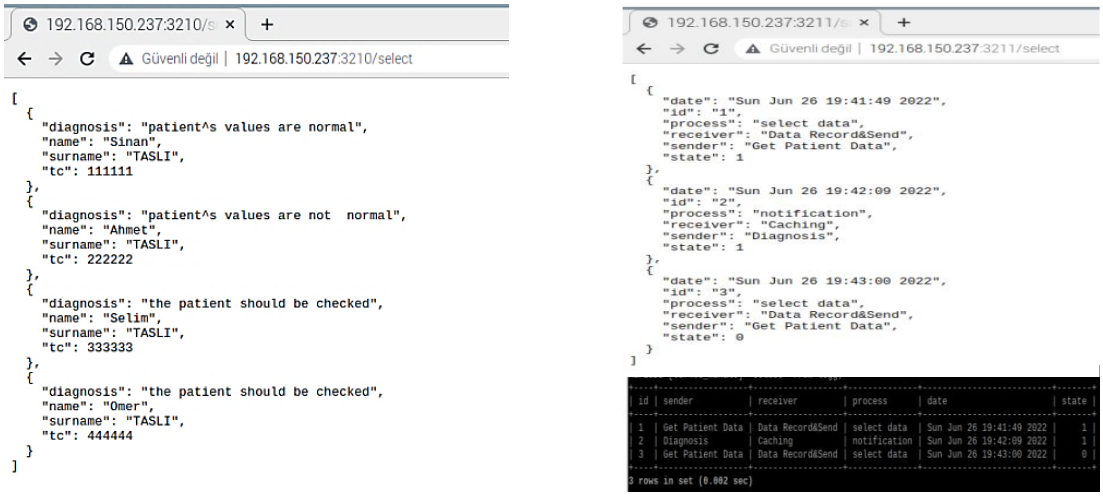
Şekil 1. HUÜ uç cihaz yapısı

Önerilen sistemin ikinci önemli ünitesi, bileşenleri Şekil 2'de gösterilen AHUÜ'dir. AHUÜ'de toplam 7 adet mikroservis çalışmaktadır. Bu mikroservislerden Emergency mikroservisi hastalardan gelen acil durum bilgilerini takip etmektedir. Bu mikroservis, sistem için kritik bir öneme sahiptir. Hastalardan gelen acil durum bilgilerinin alınamaması veya bu serviste oluşacak bir aksaklık, sistemi genel olarak başarısız kılabilir. Bu sebepten dolayı bu mikroservisin bir devre kesici (circuit-breaker) yardımıyla takip edilmesi ve güvenliğinin sağlanması gereklidir. Devre kesici (circuit breaker), Acil (Emergency) mikroservisinin cevap süresini izler ve mevcut senaryoya göre Şekil 4-a'da gösterilen üç farklı durumdan birinde (open, half-open ve closed) bulunur. Eğer servis cevap verme süresi belirlenen eşik değerini aşar ise Emergency mikroservisini olası kötü senaryolardan korumak için circuit breaker anahtarı açık (open) konumuna getirerek devre kesiciyi açar ve gelen istekleri Management (Yönetim) mikroservisine yönlendirir. Bundan sonra gelecek 5 isteğin tümünü direkt olarak Management mikroservisi alır. Open durumu, ön tanımlı eşik sayısı kadar yeni talep için bir değişiklik göstermez. Open durumunda yapılan talep sayısı eşik değeri aştıktan sonra anahtar yarı açık (half-open) konumuna geçer ve gelen ilk 5 istek Management ile birlikte Emergency mikroservisine de iletilir. Burada Emergency mikroservisinin düzgün çalışıp çalışmadığı kontrol edilir. Emergency mikroservisi gelen isteklere istenen düzeyde cevap verirse anahtar kapalı (close) konumuna geçer ve yeni gelecek olan isteklerin tümü sadece Emergency mikroservisine iletilir. Aksi durumda ise anahtar tekrar açık konumuna geçerek yeni gelecek isteklerin Management mikroservisine gönderilmesi sağlanır.



Şekil 2. AHUÜ uç cihaz yapısı

Aile hekimliği takip ettiği hastaların verilerini rutin aralıklarla veya talep üzerine görmek isteyebilir. Bu servis, AHUÜ'de yürütülen GetPatientData (Hasta Verileri) mikroservisi ile sağlanmaktadır. Bu mikroservis planlı veya talep üzerine hasta raporlarını almakla görevlidir, asenkron iletişim modunda çalışır ve kendine ait veritabanı bulunmaktadır. Bu veritabanında hastadan alınan vital bulgular ve doktorun hastalar için yaptığı teşhis bilgileri bulunmaktadır. Gerek uç sistem içinde gerekse bulut-uç iletişiminde bazı bilgiler veya sorgular daha sık işlenebilir. Benzer işlem/sorgu sonuçlarının hızlı bir şekilde sunulması için bir ön bellek(caching) mekanizmasının kullanılması bu tür sistemlerde ciddi performans artışı sağlamaktadır. Bu doğrultuda önerilen sistem bir caching mikroservisini de yapısında bulundurmaktadır. Performans kriterleri açısından Caching mekanizması sadece GetPatientData ve bulut mikroservisleri ile etkileşimlidir. GetPatientData mikroservisi yeni teşhis bilgisinin eklenmesi durumunda Caching(önbellekleme) mikroservisine bilgi vermektedir. Bu sayede Caching mikroservisi yeni verilere ulaşabilmektedir. Bu mekanizma bulut tarafından yapılan isteklere daha hızlı cevap verebilmek için kendinde her zaman en güncel hasta teşhis bilgilerini bulundurur. Bu durum sayesinde buluttan gelen isteklerde veritabanına sorgulama yapılmasına gerek duyulmadan istekler cevaplandırılmaktadır. Caching mikroservisinin kendinde barındırdığı hasta teşhis verilerine ait görsel Şekil 3-a'da bulunmaktadır. Mikroservis mimaride tüm servislerin ve servis sorgularının izlenmesi sürdürülebilirlik açısından önemlidir. Bu sebeple bu tür sistemlerde bir loglama servisi sıklıkla kullanılır. Önerilen sistemde, Logging/Monitoring (Olay Kayıt ve İzleme) mikroservisi yapılan işlem sonuçlarını arka planda takip etmektedir. Her bir sorgu Şekil 3-b'de verilen format ile Logging/Monitoring servisine iletilmektedir. Bu formatta, sorguyu gönderen ve alan mikroservis bilgisi, sorgunun içeriğine ait bilgi, sorgunun yapıldığı zaman ve sorgunun başarılı olup olmadığını bildiren veriler bulunmaktadır. Sorgular bir ID ile takip edilmektedir ve böylece sorguların işlem süreci, başarısız servislerin takibi daha rahat bir şekilde yapılabilmektedir.

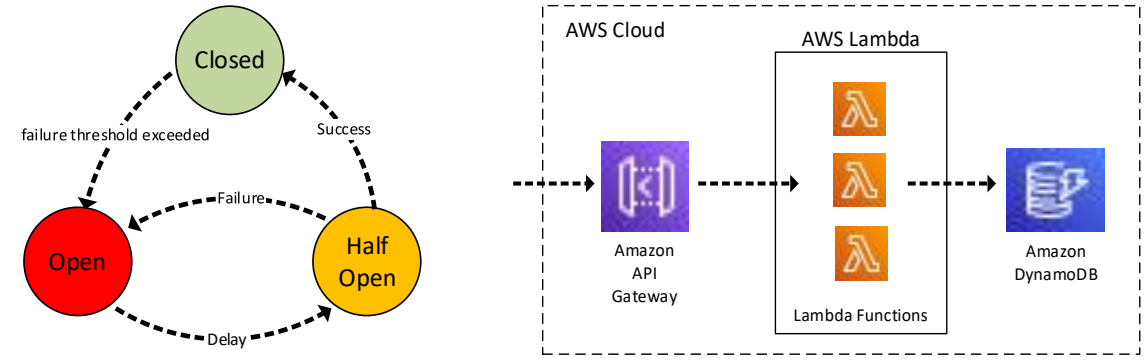


a) Caching sorgusu

b) Logging formatı

Şekil 3. Önbellek (Caching) ve Olay kayıt ve izleme (Logging) servis formatları

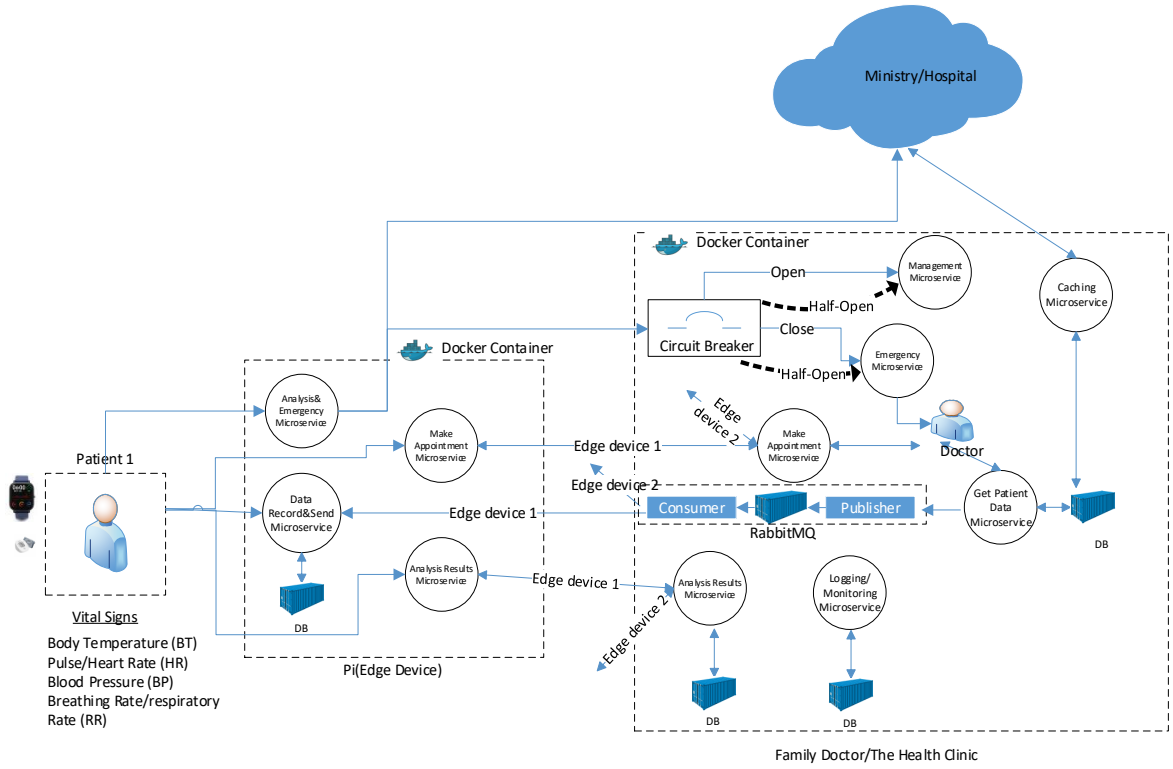
Mikroservisler kendi aralarında ve bulut ile senkron ve asenkron olarak haberleşmektedir. Asenkron haberleşme için kuyruk yapısı kullanılmaktadır. Uç cihazlar kaynak yönünden kısıtlı olduklarından dolayı veri tabanları, kuyruk yapıları ve mikroservisler konteyner olarak çalıştırılmaktadır. Hasta takip sistemindeki tüm konteynerlerin kontrolü ve yönetimi Docker Swarm tarafından yapılmaktadır. Docker Swarm sayesinde konterynerler üzerinde yük dengeleme, izleme ve konteynerlerden herhangi birinin çalışamaz duruma gelmesi halinde aynı konteynerden yenisinin oluşturulması yapılabilmektedir. Çalışmada, bulut servisler ve entegrasyonları için Amazon Web Servisileri (AWS) kullanılmıştır. Aslında bulut servis üzerinde birçok farklı servis bulunmakla birlikte bu çalışmada bulut servislerinin aile hekimliklerinden aldıkları sorguları depolama işlemleri dikkate alınmıştır. Şekil 4-b'de, kullanılan AWS bulut servisleri ve yapısı gösterilmektedir. Buluta gelen istekler Api Gateway üzerinden gerekli Lambda fonksiyonlarına iletilmektedir. Bulut üzerinde gerek duyulan verilerin kaydedilmesi için Amazon DynamoDB kullanılmaktadır. Önerilen sistemin buraya kadar bahsedilen tüm mekanizmalarını içeren genel yapısı Şekil 5'te görülebilir.



a) Devre kesici (Circuit breaker) durum diyagramı

b) Bulut yapısı

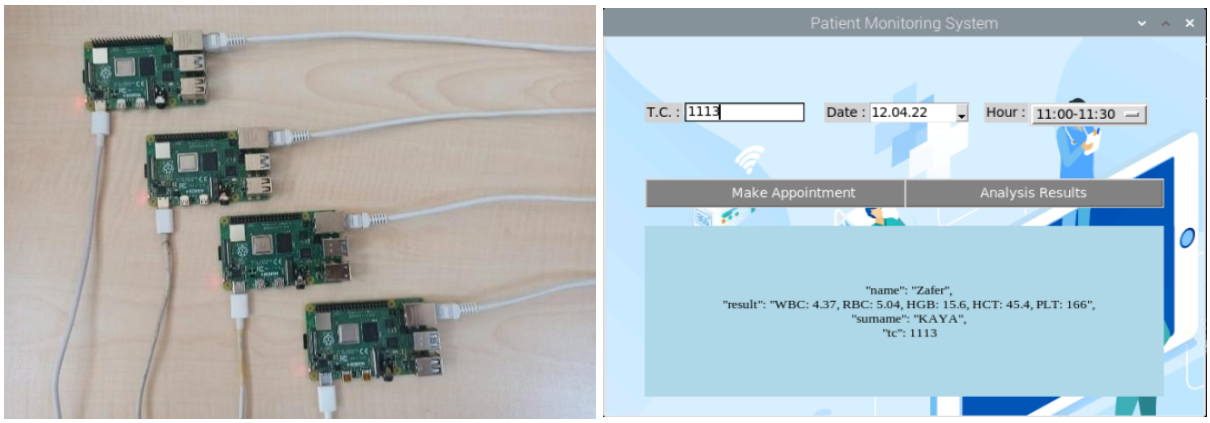
Şekil 3. Devre Kesici (Circuit breaker) durum diyagramı ve bulut yapısı



Şekil 4. Hasta takip sistemi genel yapısı [24]

4. Deneyler ve Bulgular

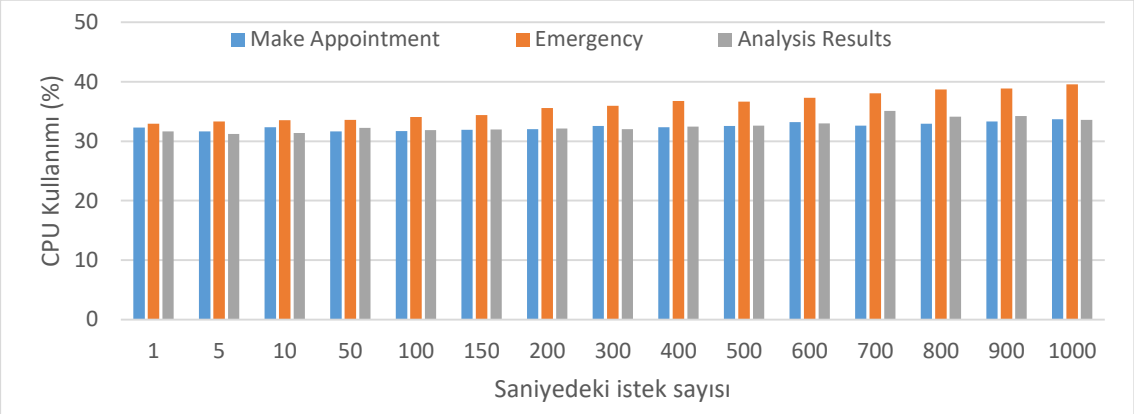
Önerilen sistemin fiziksel gerçekleştirmesi için kısıtlı kaynaklı SoC aygıtlar ve ticari bulut servisleri kullanılmıştır. SoC aygıt olarak Şekil 6-a'da gösterilen Raspberry Pi 4 Model B tercih edilmiş ve bulut için AWS servisleri alınmıştır. Şekil 6-b'de HUÜ'de kullanılmak üzere tasarlanan arayüz bulunmaktadır. Mikroservisler Python Flask platformunda kodlanmış, ilişkisel veri tabanı olarak MariaDB ve kuyruk yapısı için ise RabbitMQ kullanılmıştır. Sistem için istekler Apache JMeter yardımıyla oluşturulmuş ve ağ trafiği senaryoları Tcpreplay üzerinden kontrol edilmiştir. Uç cihazların CPU kullanımı, hafıza kullanımı ve mikroservislerin isteklere verdikleri cevap süreleri gibi performans metrikleri Zabbix yazılımı sayesinde takip edilmiştir.



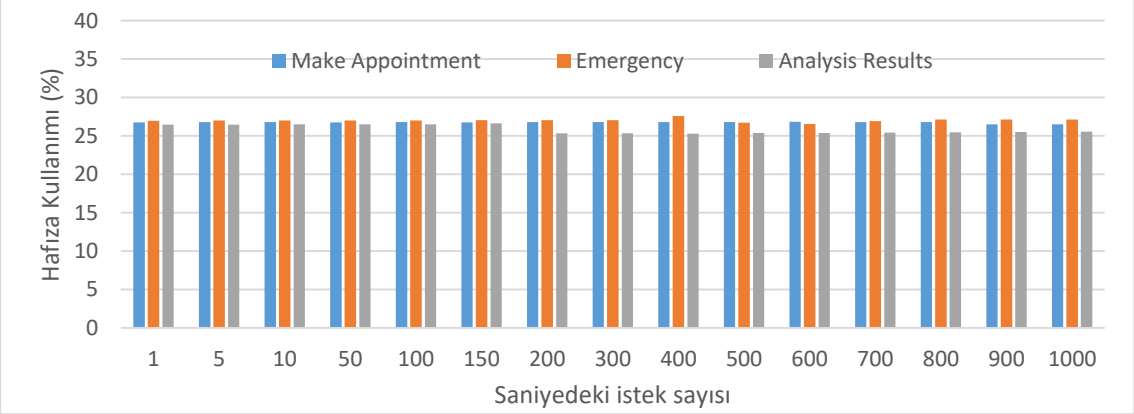
a) Raspberry Pi b) Hasta için tasarlanan kullanıcı arayüzü

Şekil 6. Kullanılan Raspberry piler ve hasta kullanıcı arayüzü

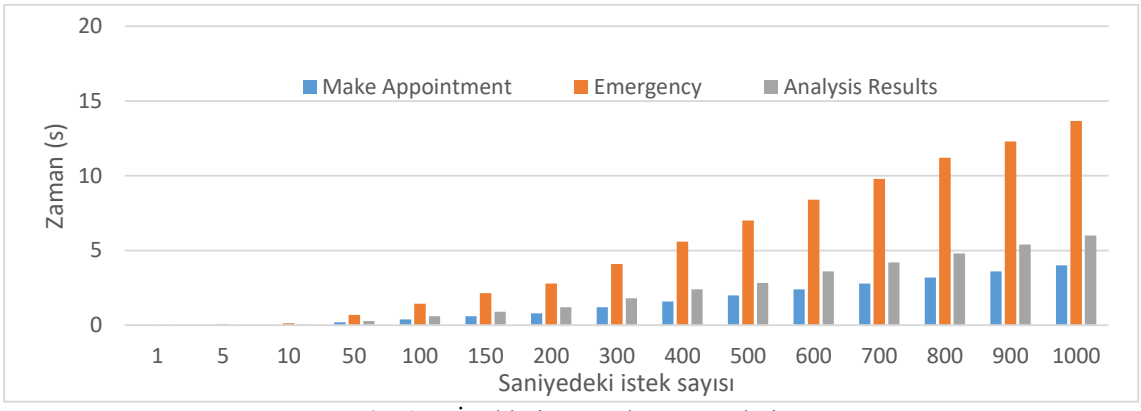
Yapılan ilk deneylerde kritik servislerin durumları izlenmiştir. Bunun için ilk olarak Emergency mikroservisi performansı gözlemlenmiştir. Bu servise yapılan isteklerde özellikle saniyede 100 istek ve üzerindeki durumlarda circuit breakerin açıldığı görülmüştür. Ölçümler esnasında circuit breaker sayesinde Emergency mikroservisinin hiçbir zaman için çalışamaz duruma gelmediği gözlenmiştir. Bu sonuç, önerilen sistemin kararlı çalıştığını göstermektedir. Ayrıca Caching mikroservisinin kullanımı sayesinde buluttan gelen isteklerin cevap sürelerinin %40 oranında azaldığı da deneylerde görülmüştür. Aile hekimliği tarafındaki uç cihazda çalışan Analysis Result, Emergency ve Make Appointment mikroservisleri hastalardan gelecek olan istekleri karşıladıkları için ölçümler bu mikroservisler üzerinde yapılmıştır. Ölçümler alınırken, ölçüm değerleri saniyede 1000 isteğe kadar test edilmiş ve tasarlanan sistemin problemsiz çalıştığı gözlenmiştir. Şekil 7'de aile hekimliği tarafındaki uç cihazın CPU kullanımı, Şekil 8'de hafıza kullanımı ve Şekil 9'da da isteklerin cevap süreleri gösterilmektedir. Şekil 7- 9'dan da görüleceği üzere aile hekimliğine yapılan istekler için en çok CPU kaynak kullanımı Emergency mikroservisi isteklerinde gerçekleşmiştir. Bunun ana nedeni Emergency mikroservisinin bir circuit breakera sahip olmasıdır. Emergency mikroservisine gelen isteklerde diğer isteklerin aksine circuit breaker, Emergency mikroservisini takip etmek için fazladan işlem yapmaktadır. Bu durum işlemci kullanımını ve isteğin cevaplanma süresini artırmaktadır. Uç cihaz (AHUÜ) üzerinde yapılan isteklerde hafıza kullanım oranının her durum için yaklaşık %26 olduğu görülmektedir. Aynı istekler buluta da yapılmış ve cevap süreleri ölçülmüştür. Şekil 10'da buluta yapılan isteklerin cevap süreleri gösterilmektedir. Uç cihaz (AHUÜ) üzerinde yapılan ölçümlerde mikroservislerin isteklere cevap verme süreleri ölçülmüş ve en yüksek değerin Emergency mikroservisine ait olduğu anlaşılmıştır. Bu değer, bulutun isteklere verdiği cevap süreleri ile karşılaştırıldığında çok büyük farklar olduğu ve isteklerin bulut yerine uç cihazlarda işlenmesinin sistemi ne kadar hızlandırdığını açıkça göstermektedir.



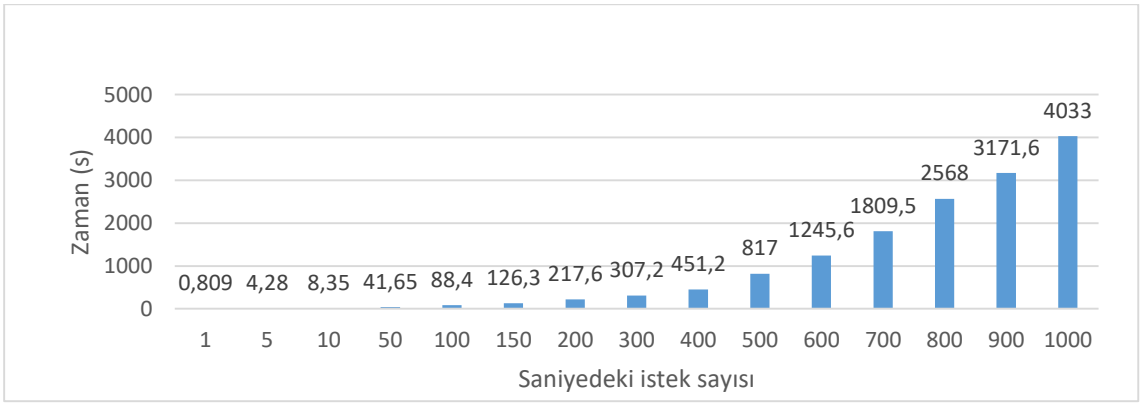
Şekil 5. CPU kullanımı



Şekil 6. Hafıza kullanımı



Şekil 7. İsteklerin cevaplanma süreleri



Şekil 8. Buluta yapılan isteklerin cevap süreleri

5. Sonuçlar

Bu çalışmada bulut teknolojisi kullanan hasta takip hizmetleri için mikroservis temelli bir uç sistem tasarımı önerilmiştir. Özellikle aile hekimliği sorumluluk alanlarındaki hasta takibi için tasarlanan bu sistemde, hasta verileri öncelikle uç sistem üzerinde işlenmiştir. Bu sayede bulut iletişim yükü ve performansı önemli ölçüde iyileştirilmiştir. Önerilen sistem üç ana alt üniteden meydana gelmektedir. Bunlar hasta taraflı uç sistem, aile hekimliği taraflı uç sistem ve bulut sistemdir. Her ünite faklı özellikte ve sayıda mikroservis çalıştırmaktadır. Mikroservislerin kurulumu, kontrolü ve sürdürülebilirlikleri konteyner teknolojileri ile sağlanmıştır. Asenkron ve senkron çalışma modlarına sahip mikroservis yürütümlerinde ayrıca kritik hizmetler için devre kesici (circuit breaker) servisi de bulunmaktadır. Bu şekli ile önerilen sistem, kısıtlı kaynaklı uç cihazlar üzerinde kolaylıkla çalıştırılabilir ve sürdürülebilir niteliklere sahiptir. Sistemin başarımı, gerçek SoC uç cihazları ve bulut servisleri ile ispatlanmıştır. Yazarların bundan sonraki çalışmaları, önerilen uç-bulut sistemi üzerinden yapay zekâ temelli teşhis algoritmalarının denenmesi olacaktır.

Açıklama

Bu makale, yazarlardan Sinan TAŞLI'nın Bulut Teknolojisi Kullanan Hasta Takip Hizmetlerinde Mikroservis Temelli Uç Sistem Tasarımı ve Geliştirilmesi isimli yüksek lisans tezinden türetilmiştir.

Kaynaklar

[1] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (IOT): A literature review," Journal of Computer and Communications, vol. 03, no. 05, pp. 164–173, 2015. Duman M, Gürbüz AC. 3D imaging for ground-penetrating radars via dictionarydimension reduction. Turk J Elec Eng & Comp Sci 2015; 23(5): 1242-1256.

[2] N. R. Tadapaneni, "Cloud computing: Opportunities and challenges," SSRN Electronic Journal, 2018, doi: 10.2139/ssrn.3563342.
[3] P. P. Ray, D. Dash, and D. De, "Intelligent internet of things enabled Edge System for Smart Healthcare," National Academy Science Letters, vol. 44, no. 4, pp. 325–330, 2020, doi: 10.1007/s40009-020-01003-0.
[4] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for Edge/Cloud Integration," IEEE Cloud Computing, vol. 3, no. 6, pp. 76–83, 2016, doi: 10.1109/mcc.2016.124.
[5] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari, and A. R. Hussin, "Understanding service-oriented architecture (SOA): A systematic literature review and directions for further investigation," Information Systems, vol. 91, p. 101491, 2020, doi: 10.1016/j.is.2020.101491.
[6] M. Keen, Patterns implementing an SOA using an enterprise service bus. Research Triangle Park, NC: IBM, International Technical Support Organization, 2004.
[7] Yildirim G. Tatar Y. (2019), Remote user supported IoT-WSN Laboratory and testbed platform: FiratWSN, Journal Of The Faculty Of Engineering And Architecture Of Gazi University, Vol.34, Iss.4,pp: 1831-1846, doi: 10.17341/gazimmfd.571588.
[8] I. Nadareishvili and I. Nadareishvili, Microservice architecture: Aligning principles, practices, and culture. Beijing, China: O'Reilly, 2016.
[9] M. Massé, REST API design rulebook designing consistent restful web service interfaces. Beijing u.a.: O'Reilly, 2012.
[10] G. Brito, T. Mombach, and M. T. Valente, "Migrating to graphql: A practical assessment," 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019,doi: 10.1109/SANER.2019.8667986.
[11] R. Vilalta *et al*., "GRPC-based SDN control and telemetry for soft-failure detection of spectral/spacial superchannels," *45th European Conference on Optical Communication (ECOC 2019)*, 2019, pp. 1-4, doi: 10.1049/cp.2019.0874.
[12] T. Siddiqui, S. A. Siddiqui, and N. A. Khan, "Comprehensive Analysis of Container Technology," 2019 4th International Conference on Information Systems and Computer Networks (ISCON), 2019, doi: 10.1109/iscon47742.2019.9036238.
[13] R. Kumar and M. P. Rajasekaran, "An IOT based patient monitoring system using Raspberry Pi," 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16), 2016, doi: 10.1109/icctide.2016.7725378.
[14] P. Gupta, D. Agrawal, J. Chhabra, and P. K. Dhir, "IOT based Smart Healthcare Kit," 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), 2016, doi: 10.1109/icctict.2016.7514585.
[15] S. Greene, H. Thapliyal, and D. Carpenter, "IOT-based fall detection for Smart Home Environments," 2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS), 2016, doi: 10.1109/inis.2016.017.
[16] X. Luo, T. Liu, J. Liu, X. Guo, and G. Wang, "Design and implementation of a distributed fall detection system based on wireless sensor networks," EURASIP Journal on Wireless Communications and Networking, vol. 2012, no. 1, 2012, doi: 10.1186/1687-1499-2012-118.
[17] G. Devi and S. A. M. Rizvi, "Integration of Genomic Data with EHR Using IoT," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2020, pp. 545-549, doi: 10.1109/ICACCCN51052.2020.9362968.
[18] A. Alamri, Ontology Middleware for Integration of IoT Healthcare Information Systems in EHR Systems, 2018, Computers, 7(4), 51, doi: 10.3390/computers7040051.
[19] S. Shi, D. He, L. Li, N. Kumar, M. K. Khan, and K.-K. R. Choo, "Applications of blockchain in ensuring the security and privacy of Electronic Health Record Systems: A survey," Computers & Security, vol. 97, p. 101966, 2020, doi: 10.1016/j.cose.2020.101966.
[20] N. Hong, A. Wen, F. Shen, S. Sohn, C. Wang, H. Liu, and G. Jiang, "Developing a scalable FHIR-based clinical data normalization pipeline for standardizing and integrating unstructured and Structured Electronic Health Record Data," JAMIA Open, vol. 2, no. 4, pp. 570–579, 2019, doi: 10.1093/jamiaopen/ooz056.
[21] J. Hong, P. Morris and J. Seo, "Interconnected Personal Health Record Ecosystem Using IoT Cloud Platform and HL7 FHIR," *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, 2017, pp. 362-367, doi: 10.1109/ICHI.2017.82.
[22] Ngankam H.K. et al. (2019) An IoT Architecture of Microservices for Ambient Assisted Living Environments to Promote Aging in Smart Cities. In: Pagán J., Mokhtari M., Aloulou H., Abdulrazak B., Cabrera M. (eds) How AI Impacts Urban Living and Public Health. ICOST 2019. Lecture Notes in Computer Science, vol 11862. Springer, Cham. https://doi.org/10.1007/978-3-030-32785-9_14.
[23] S. Rajapaksa, A. Wickramarachchi et al., A Scalable Bioinformatics Analysis Platform based on Microservices Architecture, 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, doi: 10.23919/SCSE.2019.8842809.
[24] S. Taşlı, Bulut Teknolojisi Kullanan Hasta Takip Hizmetlerinde Mikroservis Temelli Uç Sistem Tasarımı ve Geliştirilmesi, Yüksek Lisans Tezi, 2022, Fen Bilimleri Enstitüsü, Fırat Üniversitesi, Elazığ.