Avrupa Bilim ve Teknoloji Dergisi
Sayı 33, S. 274-279, Ocak 2022

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 33, pp. 274-279, January 2022

Research Article

Yoğun Hesaplama ve Zaman Gerektiren İşlemlerin Sunucularda Yapılması

Ramazan Akkurt[1*], M. Ferhat Tüysüz[2]

[1*] Mersin Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Mersin, Türkiye, (ORCID: 0000-0003-2319-9887), rakkurt@mersin.edu.tr
[2] Northumbria University, Computer and Information Sciences Department, Newcastle, England, (ORCID: 0000-0002-8955-9710), mehmet.tuysuz@northumbria.ac.uk

(İlk Geliş Tarihi 28 Mayıs 2021 ve Kabul Tarihi 25 Aralık 2021)

(**DOI:** 10.31590/ejosat.944342)

ATIF/REFERENCE: Akkurt, R. & Tüysüz, M. F. (2022). Yoğun Hesaplama ve Zaman Gerektiren İşlemlerin Sunucularda Yapılması. *Avrupa Bilim ve Teknoloji Dergisi*, (33), 274-279.

Öz

Günden güne artan mobil cihaz sayısı, birçok büyük veya küçük ölçekli kullanıcı isteklerinin mobil cihazlara kaymasına sebep olmuştur. Bu artış cihazların sahip olduğu iş yükünü de buna bağlı olarak arttırmıştır. Ancak sınırlı kaynakları olan mobil cihazların, bazı büyük işlemleri lokal olarak kendi bünyesinde çalıştırması bazen uzun bekleme sürelerine sebep olmakta, bazen de kullanıcı deneyimini kötü etkilemektedir. Mobil cihazların sahip olduğu bu tarz işlemlerin daha hızlı ve etkili bir şekilde çalıştırılması ve sonuçlarının cihaza tekrar döndürülmesini sağlayan mobil uç hesaplama (Mobile Edge Computing) sistemi günümüzde çeşitli ağ teknolojileri ile mümkün hale gelmiştir. Bu çalışma kapsamında yoğun şekilde CPU kullanımına ihtiyaç duyan ve uzun gecikme sürelerine sebep olan multi-thread yapılı bir mobil uygulama geliştirilmiş ve bu mobil uygulamanın lokal ve MEC sistemindeki performanslarının karşılaştırılması yapılmıştır.

Anahtar Kelimeler: Mobil Uç Hesaplama, Veri Boşaltma, Mobil Bulut Hesaplama, Hücresel Ağlar.

Performing CPU-intensive and Long Time Consuming Tasks on Servers

Abstract

The increasing number of mobile devices day by day has caused many large and small scale user requests to shift to mobile devices. This increase has raised the workload of devices accordingly. However, Mobile devices with limited resources sometimes evaluate some large processes locally, causing long waiting times and sometimes adversely affecting the user experience. The MEC (Mobile Edge Computing) system, which enables such processes to be evaluated faster and more effectively and the results are returned to the device, that mobile devices have, has been developed with various network technologies today. In this study, a multi-thread mobile application that requires heavy CPU usage and has long delay has been developed and the performance of this mobile application in local and MEC systems has been compared.

Keywords: Mobile Edge Computing, Offloading, Mobile Cloud Computing, Cellular Network.

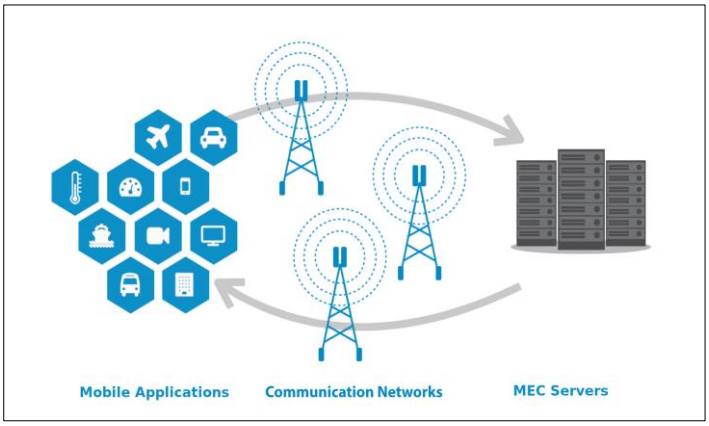
[1]Sorumlu Yazar: rakkurt@mersin.edu.tr

1. Giriş

Günümüz teknolojileri ile birlikte kullanıcı taleplerindeki artışa bağlı olarak, yüksek kaynağa ihtiyaç duyan mobil uygulamalar da yaygın hale gelmiştir. Artırılmış sanal gerçeklik (Agumented reality), video editing ve image processing gibi mobil cihazlara büyük bir yük getiren uygulamalar, cihazların kaynaklarını uzun süreler meşgul etmekte ve büyük miktarlarda enerji tüketimine sebep olmaktadır. Sınırlı kaynaklara sahip mobil cihazların bu şekilde ki uygulamaları kendi bünyesinde değerlendirmesi, uzun gecikme sürelerine neden olarak kullanıcı deneyimini kötü etkilemektedir. Bu tarz sorunları ortadan kaldırmak için çözüm önerisi olarak getirilen Mobile Cloud Computing (MCC), mobile cihazlara güçlü CPU kaynakları ve büyük miktarda depolama alanları gibi geniş kaynaklar sunmaktadır [1, 2]. Ancak, MCC kaynaklarının sabit ve merkezi olarak konumlandırılması, coğrafik koşullar göz önünde alındığında, mobil cihazlar ve MCC kaynakları arasındaki mesafeden kaynaklı olarak, uçtan uca gecikmenin uzun olması anlamına gelmektedir [3,4]. Uzun gecikme sürelerinin önüne geçmek için Cloud kaynaklarının kullanıcıya daha yakın olarak konumlandırılması ile Mobile Edge Computing (MEC) kavramı ortaya çıkmıştır [5-7]. Yüksek download-upload hızları ve geniş kapsama alanı sunan 5G ağ teknolojisi, mobil cihazlar ve MEC kaynakları arasındaki iletişim için gecikmeleri çok düşük seviyelere çekmektedir. Yüksek CPU kaynağı ve enerji tüketimi gerektiren uygulamalar MEC sistemine mobil cihaz adına yürütülmek üzere gönderilmektedir. MEC sistemi bu tarz büyük kapasiteli uygulamaları kendi bünyesinde yürüterek, sonuçları mobil cihaza göndermektedir. Bu şekildeki iş yükünün tamamının veya belirli parçalarının MEC tarafından üstlenilmesi, düşük cevap süresi gerektiren uygulamalar için hem düşük bir gecikme süresi hem de cihazlar için pil tasarrufu anlamına gelmektedir.

1.1. MEC Sistemi ve Yapısı

MEC sistemi güçlü cloud kaynaklarının dağıtık olarak konumlandırıldığı bir sistem olarak tasarlanmıştır. Mobil cihazların ilk olarak iletişim kurduğu erişim noktalarına veya baz istasyonlarına yakın olarak konumlandırılmışlardır [8, 9]. Bu yakınlık sayesinde uçtan uca gecikme sürelerinin çok kısa olması nedeniyle mobil cihazlara büyük avantajlar sağlamaktadır [10-12].



Şekil 1. MEC Sistemi ve Yapısı

Mobil cihazlar kendi lokal kaynaklarında çalıştıramadığı, yüksek pil gücü veya işlemci gereksinimi nedeniyle zorlanarak/verimsiz çalıştırabildiği uygulamaları MEC sisteminin kaynaklarından yararlanarak (MEC sunucularında değerlendirilen uygulama verilerinin tekrar mobil cihaza gönderilmesiyle) sağlayabilmektedirler. MEC sisteminin MCC'ye göre yakınlık, düşük gecikme süresi, yüksek bant genişliği, gerçek-zamanlı trafik yönetimi ve konum farkındalığı gibi temel avantajları da bulunmaktadır.

MEC sistemi veri boşaltma (Offloading), Kaynak Tahsisi (Resource Allocation) ve Hareketlilik Yönetimi (Mobility Management) gibi 3 temel yapıda çalışma alanı mevcuttur. Bu üç yapı için temel olarak MEC sistemine ne gibi işlevsellikler kattığına aşağıda kısaca değinilmiştir.

1.2. Veri Boşaltma

MEC kaynakları ve mobil cihazlar arasındaki iletişim ve veri transferi hücresel ağ trafiğini etkilemeden ayrı bir iletişim kanalı üzerinden boşaltma (offloading) olarak gerçekleştirilmektedir [13-15]. Offloading veri iletimi Küçük Hücre Ağları (SCN), Wifi offloading ve Ad-Hoc temelli offloading olarak üç şekilde gerçekleştirilmektedir.

Mobil cihazdaki uygulama verileri çalıştırılmak üzere yukarda bahsedilen iletim modelleri kullanılarak MEC sistemine aktarılmaktadır. Uygulamalar tamamen lokal cihazda çalışırılabilir veya tamamı MEC'te çalıştırılabilir (Tam Offloading) ya da belirli parçalar lokalde kalan diğer parçalar ise MEC'te çalıştırılabilir (Yarı-Offloading) şekilde 3 farklı iletim modeli olarak gerçekleştirmektedir [16]. Offload olarak iletim yapılacak uygulama parçalarına nasıl karar verilecek? Hangi durumlarda offload yapılmalı/yapılmamalı ya da en ideal offloading kararı nasıl verilmeli? gibi temel konular, offloading kararları (offloading decisions) konu başlığında irdelenen temel çalışma konularıdır [17, 18].

1.3. Kaynak Tahsisi

MEC sistemi birden fazla kullanıcıya aynı anda cevap verebilecek kaynak kapasitesine sahiptir. Birden fazla mobil cihaz MEC sisteminden aynı anda yararlanabilmektedir. MEC, en ideal ve uygun gecikme sürelerinde, kullanıcıların taleplerini karşılayabilmek için kaynaklarını belirli oranlarda tahsis eder [19, 20]. Kaynak tahsisi kullanıcıların talep ettiği iş yüküne göre farklılık göstermektedir. Hangi cihaza ne kadar kaynak tahsis edilecek ve bu kaynaklar ortalama ne kadar sürede meşgul edilecek gibi tasarım sorunları MEC sisteminde hala araştırma konuları arasında yer almaktadır [21]. MEC bazı durumlarda komşu MEC sistemler ile iş birliği içinde çalışarak kendi içinde de yük dengeleme yapabilme olanağına sahiptir. Aşırı yük binen bir MEC sistemi kendi yükünü komşu MEC sistemi ile paylaşabilmektedir.

1.4. Hareketlilik Yönetimi

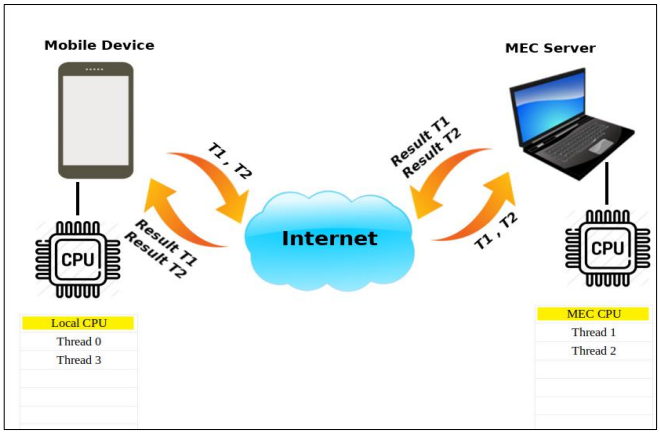
Hücresel ağ kullanarak iletişim sağlayan mobil cihazların, bulunduğu konumlardan sürekli olarak hareket halinde olması durumda ortaya çıkabilecek sorunlar ele alınması gereklidir. Kullanıcıların MEC kaynaklarını kullandıkları lokasyondan ayrılması veya uzaklaşması, MEC sistemi ile kullanıcı arasındaki mesafeden kaynaklı olarak gecikme sürelerinin artmasına sebep olacaktır. Geleneksel mobil iletişim teknolojilerinde kullanılan handover (el değiştirme) sistemi sayesinde servis aldığı hücresel ağın kapsama alanından uzaklaşan veya ayrılan kullanıcının, alınan servisin devamlılığı için yeni bulunduğu lokasyondaki en uygun ağa devredilmesi gereklidir [22, 23]. İletişimin en temel problemlerinden biri

kullanıcıların hareketli olmasından dolayı ortaya çıkan hareketliliğin yönetilmesinin güçlüğüdür [24]. Hareketlilik yönetiminin amacı ağların etki alanı içinde hareket halindeki birimler ile sağlıklı bir şekilde iletişim kurmak ve bu iletişimin devamlılığını sağlamaktır. Bu sayede gelen ve giden veriler kesintisiz bir şekilde hareket halindeki kullanıcılara ulaşabilmektedir. Yukarda bahsedilen handover senaryosu MEC sistemi içinde aynı anlama gelmektedir. Bir mobil cihaz MEC sisteminden yararlanırken, konumunu değiştirdiği anlarda MEC sistemleri arasında da servis migrasyonu yapılması gerekmektedir. Mobil cihazın mevcut MEC sistemden aldığı servisler ve tahsis ettiği kaynaklar, hiçbir kesinti olmadan bulunduğu konumdaki en ideal ve yakın MEC sistemine taşınır [25-27]. Bu tarz senaryolarda servis kalitesi bozulmadan ve uzun bekleme sürelerine neden olmadan servis migrasyonunun yapılması gerekmektedir. En ideal yeni MEC sisteminin seçilmesi temel tasarım sorunlarının başlıcalarındandır.

2. Materyal ve Metot

2.1. Sistem Modeli

Çalışmamızda kullanılmak üzere bir adet mobil cihaz ve MEC sistemini temsil edecek şekilde programlanmış yüksek CPU kaynağı olan bir adet dizüstü bilgisayar kullanılmıştır. Her iki platformda da (MEC sistemini temsil eden dizüstü bilgisayar ve mobil cihaz) kullanılmak üzere yüksek CPU kullanımı gerektiren bir mobil uygulama geliştirilmiştir. Geliştirilen uygulamanın tüm iş yükünü mobil cihaz tek başına kaldıracak kadar yeterli kaynağa sahip değildir. Mobil cihaz kendi kaynaklarını kullanarak uygulamayı yürüttüğünde büyük bir yürütme gecikmesi ile karşı karşıya kalmaktadır. Bu noktada MEC sistemi devreye girerek uygulamanın daha kısa sürelerde yürütülmesine olanak sağlamaktadır. Şekil 2'de gösterilen iletişim modeli kullanılarak uzun gecikme sürelerinin önüne geçmek için, uygulamada uzun bekleme sürelerine neden olan threadler, ağ üzerinden MEC sistemine iletilir. İletilen bu threadler, mobil cihaz yerine güçlü kaynakları sayesinde MEC sistemi tarafından değerlendirilir. Bu modelde mobil cihaz, sadece MEC sistemine uygulama verilerini iletmekten ve sonuçları MEC'ten alma görevini yerine getirmektedir.



Şekil 2. Sistem Modeli

2.2. Uygulama ve MEC Arasındaki İletişim Modeli

Uçlar arası iletişim için gRPC kullanılmıştır [28]. gRPC google tarafından geliştirilen ve platform bağımsız, açık kaynaklı ve yüksek performanslı bir RPC framework'üdür. Diğer web servislere nazaran daha hızlı olan, implementasyonu daha kolay olan ve Http/2 stream desteği sunduğu için uçları arası iletişimde gRPC kullanılmıştır. Uçlar arası serileştirme için gRPC ile tam uyumlu olarak çalışan Protobuf kullanılmıştır [29]. Protobuf (Protocol Buffers) Google tarafından geliştirilmiş bir binary serileştirme protokolüdür. Hem MEC tarafında hem de Mobil uygulama tarafında aynı dilin konuşulması için uygun bir proto file oluşturulmuştur. Mobil cihaz göndermek istediği verileri serileştirir (serialization) ve gRPC kullanarak ağ aracılığıyla bu verileri MEC server'a iletir. MEC server alınan verileri tam tersi yönde serileştirerek (deserialization) elde eder. MEC server aldığı bu veriler dahilinde mobil cihazın yerine kendi kaynaklarını kullanarak var olan işi yerine getirir. MEC server'ın ürettiği sonuçlar mobil cihaza iletilirken de aynı yapıyı kullanır. Bu sayede platform ve kullanılan teknolojiler bağımsız olarak aynı dili konuşan 2 adet uç nokta birbirleriyle sorunsuz şekilde iletişim kurabilmektedir.

2.3. Mobil Uygulama

Android ortamında geliştirilen multi-thread yapılı mobil uygulama, yoğun olarak CPU kullanımı ve buna bağlı olarak pil tüketimi ile cihaz kaynaklarını büyük oranda meşgul edecek şekilde tasarlanmıştır. Geliştirilen uygulama, gereksinim duyduğu işleri yerine getirmek için 4 adet eşzamanlı thread kullanmaktadır. Threadler işlemciyi yoğun şekilde kullanacak hesaplama işlemleri (Büyük sayıların fibonacci serileri ve faktöriyel hesaplamaları, çok sayıda rastgele şifre oluşturucu, Oluşturan şifreleri MD5 ile şifreleme vb.) ile hem cihaz kaynaklarını yoğun şekilde meşgul etmekte hem de büyük miktarlarda pil tüketimine sebep olmaktadır. Mobil uygulama threadlerinin sonuçlanması oldukça uzun bekleme sürelerine sebep olmaktadır.

2.4. MEC Sistemi ve Yapısı

Mobil cihazdan gelecek isteklere cevap verebilecek uygun bir MEC server programlanmıştır. MEC Server mobil cihazdan gelen talep doğrultusunda, kendi kaynaklarını kullanarak değerlendireceği threadler için gerekli verileri mobil cihazdan talep eder. Mobil cihaz, MEC'te değerlendirilmesini istediği thread'lere ait bytecode'ları, global değişkenleri ve kullanılan metotların aldığı parametre değerlerini MEC server'a offload veri ilerim yönetimi ile iletmektedir. MEC server cihazdan gelen bytecode'ları saklayarak bir sonraki istekte tekrar cihazdan yüklemek yerine sakladığı bu bytecode'lardan yararlanır. MEC server aldığı bytecodelardan yararlanarak thread'leri bir arayüz aracılığıyla ayağa kaldırır. Global değişkenleri ve parametreleri atanan threadler eşzamanlı olarak MEC server kaynakları kullanılarak çalıştırılırlar. MEC server her thread için üretilen sonuçları mobil cihaza iletir. Güçlü kaynaklara sahip MEC server mobil cihaza kıyasla bu işleri daha kısa sürelerde gerçekleştirmektedir.

2.5. Performans Kriteri

Planlanan modele göre, mobil cihaz öncelikle MEC server'ın ihtiyacı olan tüm verileri iletecek ve sonuçları bekleyecektir. Bu modelde çift taraflı bir veri iletimi söz konusudur. Cihaz ve MEC arasındaki ağ gecikmesi büyük önem arz etmektedir.

Bu durumda MEC server'ın tüm işleri tamamlamak için harcayacağı toplam süre;

$$L_T = L_O + E_M + L_R \quad (1)$$

Olarak hesaplanmaktadır.

Çalışmada uçlar arası gecikmenin toplam maliyeti 1'de verilen

denklemle, 3 farklı gecikme süresi göz önüne alınarak hesaplanmıştır. L_T bir thread'in tamamlanması için geçen toplam yürütme süresini temsil etmektedir. L_O mobil cihazdan verilerin MEC server'a iletilmesi için geçen toplam süreyi ifade etmektedir (Offloading Latency). E_M MEC serverda thread başına harcanan toplam süreyi ifade etmektedir (MEC execution time). L_R ise sonuçların mobil cihaza geri iletimi için geçen toplam süreyi ifade etmektedir. Mobil cihazın bir thread'i kendi bünyesinde çalıştırdığı senaryoda ortaya çıkan toplam gecikme ve MEC server'ın aynı işi yapmak için harcadığı toplam süre temel performans kriteri olarak belirlenmiştir. Her thread için lokal cihazda harcanan süre ile MEC server'da aynı thread için harcanan toplam süre karşılaştırılarak MEC server'ın çalışma süresinin ne kadar minimize ettiği saptanmıştır. Tüm threadler hem lokalde hem de MEC serverda çalıştırıldığı gibi, bazı threadler lokalde çalışırken aynı anda diğer kalan threadler eşzamanlı olarak MEC serverda çalıştırılarak farklı senaryolar ile, performans etkisinin kapsamı irdelenmiştir. (Örneğin Thread-0 lokalde, Thread-1 lokalde, Thread-2 MEC'te, Thread-2 MEC'te veya bir thread lokalde, diğer kalan 3 thread MEC'te vb.)

3. Araştırma Sonuçları ve Tartışma

3.1. Simülasyon Çalışması

Simülasyon çalışması için mobil tarafında, cores 1.2 GHz ARM Cortex-A7 işlemci, 1GB RAM özelliklerine sahip GM discovery 4 mobil cihaz kullanılmıştır. MEC server ve mobil cihaz arasında iletişim için ortalama 25Mbps indirme hızı ve 5Mbps yükleme hızlarına sahip bir ağ kullanılmıştır. Bu hızlar MEC sisteminin 5G hızlarına yakın olmadığı için performans çıktılarının tam anlaşılması için 5G ağ teknolojisine göre oranlama yapılması daha doğru sonuçlar verecektir. Bu senaryoda elde edilecek kazanç, 5G gibi daha büyük hızları garanti eden ağın kullanımında daha büyük kazançların elde edileceği anlamına gelmektedir.

5G ve MEC sistemi bulunduğumuz lokasyonda henüz mevcut olmadığı için MEC server için Intel© Core™ i5-3230M CPU 2.60GHz ×2, 6GB RAM özelliklerine sahip Asus X550V kişisel bilgisayar kullanılmıştır. Gerçekleştirilen denemelerde thread başına ortalama MEC ve lokal cevap süreleri programatik olarak hesaplanmıştır. Thread'ler eş zamanlı olarak hem mobil cihazda hem de MEC serverda yürütülerek, bir thread'in işini bitirme süresi mili saniye cinsinden hesaplanmıştır. Elde edilen sonuçlar detaylı olarak tablolar halinde verilmiştir.

3.2. Bulgular ve Tartışma

Elde edilen sonuçlar tablolar halinde verilmiş olup performanslar değerlendirilmiştir. Lokal ve MEC yürütme süreleri göz önüne alınarak MEC sisteminin lokaldeki yürütmeye göre ne kadar daha kısa sürede veya ne kadar daha uzun sürede mobil cihaza yanıt verdiği ortalama verim olarak hesaplanmış ve tabloda gösterilmiştir.

Tablo 1. Thread-0 için elde edilen sonuçlar

Lokal yürütme Süresi (ms)	Gönderilen Veri Boyutu (Bytes)	Çıktı boyutu (Bytes)	MEC yürütme Süresi (ms)	Ortalama Verim
26	1610	985	120	-78%
109	1612	1048	49	+55%
121	1617	2093	53	+56%
130	1621	5228	61	+53%
257	1634	16723	97	+62%

Tablo 1'de elde edilen sonuçlara göre thread-0 için MEC server, mobil cihazın kendi kaynaklarını kullanarak thread-0'ın yürütmesi için harcadığı toplam süreye göre %41 daha iyi performans göstermiştir. Thread-0 için toplam yürütme süresini mobile cihaza göre 263ms daha kısa sürede gerçekleştirmiştir.

Tablo 2. Thread-1 için elde edilen sonuçlar

Lokal yürütme Süresi (ms)	Gönderilen Veri Boyutu (Bytes)	Çıktı boyutu (Bytes)	MEC yürütme Süresi (ms)	Ortalama Verim
143	1544	16329	84	+41%
203	1556	35664	106	+48%
1058	1559	2093	517	+51%
885	1569	99098	412	+53%
867	1607	92277	435	+50%

Tablo 2'de elde edilen sonuçlara göre thread-1 için MEC server, mobil cihazın kendi kaynaklarını kullanarak thread-1'nin yürütmesi için harcadığı toplam süreye göre %50 daha iyi performans göstermiştir. Thread-1 için toplam yürütme süresini mobile cihaza göre 1602ms daha kısa sürede gerçekleştirmiştir.

Tablo 3. Thread-2 için elde edilen sonuçlar

Lokal yürütme Süresi (ms)	Gönderilen Veri Boyutu (Bytes)	Çıktı boyutu (Bytes)	MEC yürütme Süresi (ms)	Ortalama Verim
574	2392	916808	687	-20%
458	2397	921248	808	-76%
454	2390	925688	727	-60%
727	2402	983488	588	+19%
921	2452	1321432	604	+34%

Tablo 3'de elde edilen sonuçlara göre thread-2 için MEC server, mobil cihazın kendi kaynaklarını kullanarak thread-2'nin yürütmesi için harcadığı toplam süreye göre %8 daha kötü

performans göstermiştir. Thread-2 için toplam yürütme süresini mobile cihaza göre 280ms daha uzun sürede gerçekleştirmiştir.

Tablo 4. Thread-3 için elde edilen sonuçlar

Lokal yürütme Süresi (ms)	Gönderilen Veri Boyutu (Bytes)	Çıktı boyutu (Bytes)	MEC yürütme Süresi (ms)	Ortalama Verim
58	2908	72372	142	-59%
134	2934	84872	124	+7%
450	2954	169872	81	+82%
47	2901	68724	132	-64%
520	2978	403482	226	+57%

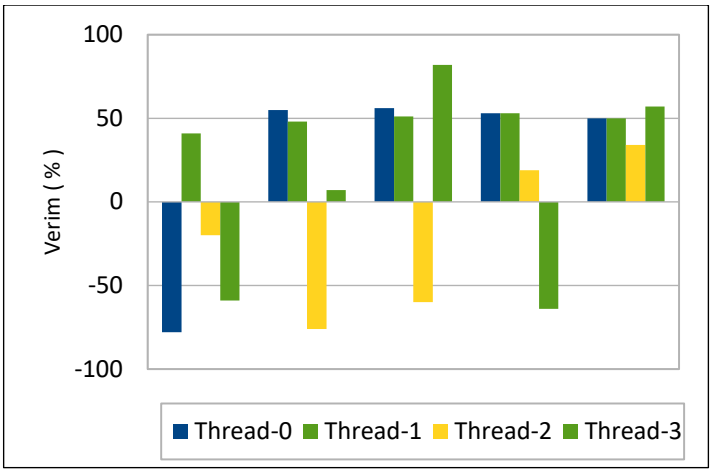
Tablo 4'de elde edilen sonuçlara göre thread-3 için MEC server, mobil cihazın kendi kaynaklarını kullanarak thread-3'nin yürütmesi için harcadığı toplam süreye göre %42 daha iyi performans göstermiştir. Thread-3 için toplam yürütme süresini mobile cihaza göre 504ms daha kısa sürede gerçekleştirmiştir.

Simülasyonda kullanılan MEC server thread-0-1-3 için oldukça iyi performans gösterse de thread-2 için tam tersi sonuç göstermiştir. Bu sonucun ortaya çıkmasında temel neden, sonuçların mobil cihaza gönderilmesinde geçen sürenin (L_R) uzun olmasında kaynaklanmaktadır. Thread-2'nin ilk çalıştırması daha detaylı olarak irdelendiğinde denklem 2'deki hesaplanan süreler denklem 3'te yerine koyulunca sonuçlar şöyle hesaplanmıştır;

$$L_O = 17ms, E_M = 308ms, L_R = 362ms \quad (2)$$

$$L_T = 17 + 308 + 362 = 687ms \quad (3)$$

MEC sistemine verilerin (bytecode'lar, global değişkenler ve metotların parametreleri vb.) gönderimi için geçen offloading süresi 17ms (L_O) olarak hesaplanmıştır. MEC'te thread-2'nin işini bitirmesi için geçen toplam süre 308ms (E_M) olarak hesaplanmış, sonuçların mobil cihaza iletilmesi için geçen süre ise 362ms (L_R) olarak hesaplanmıştır. MEC server thread-2'nin ilk yürütmesi için harcadığı toplam süre (E_M=308ms) ideal olmasına rağmen, sonuçların mobil cihaza gönderilmesi için geçen toplam sürenin (L_R=362ms) oldukça fazla olmasından dolayı bu tarz sonuçların ortaya çıktığı görülmektedir. Thread-2 için MEC server tarafından 5 kez çalıştırma sonucu -8% kötü performans gösterse de cihazın bu iş için harcayacağı toplam enerji miktarından tasarruf edileceği açıktır. Pil tasarrufunun daha önemli olduğu ve yürütme süresinin ihmal edilebilir olduğu durumlarda %8 gibi rakamlar göz ardı edilebilir.



Şekil 3. Thread başına ortalama verim

Şekil 3'te de gösterildiği gibi toplamda 4 thread için 20 kez MEC server tarafından elde edilen sonuçlardan 14 sonuç pozitif, kalan 6 sonuç ise negatif yönde daha kötü performans göstermiştir. Yukarıda da bahsedildiği gibi kötü sonuçlar için, bazı durumlarda pil tasarrufu bir kazanım koşulu olarak tolere edilebilir (Mobil cihaz sadece verileri MEC sistemin yollarken düşük miktarda bir enerji harcayacaktır.). Ortalama olarak MEC sistemi mobil cihaza göre thread başına; Thread-0 için +41%, Thread-1 için +51%, Thread-2 için -8%, Thread-3 için +42% performans göstermiştir. Bu sonuçlar bize ortalama olarak bir thread'in mobil cihaza göre MEC sisteminde ne kadar daha kısa sürede veya uzun sürede işlerini tamamladıklarını göstermektedir. Sonuçlara göre mobil cihaz bu 4 threadli uygulamanın tamamını MEC sisteminde yürütseydi kendi kaynaklarındaki yürütmeye göre uygulamayı %26 daha kısa sürede tamamlanmış bir şekilde kullanıcıya cevap verecekti. Thread-2 için kötü sonuçların elde edilmesi bu oranı aşağı çekmektedir. Bu tarz durumların meydana gelmesi halinde literatürde , daha kötü performans vermesi beklenen veya belirli yöntemlerle önceden daha kötü performans vereceği tahmin edilen işler , MEC sistemi yerine cihazda yürütülerek mevcut koşullarda en iyi performansın elde edilmesi sağlanmaktadır(Yukarda da bahsedildiği gibi tam-offloading veya yarı-offloading şeklinde işler belirli oranda MEC ve mobil cihaz arasında paylaştırılarak).Bizim tasarımımızda eğer thread-0-1-3 MEC sisteminde , thread-2 ise mobil cihazın kaynaklarında yürütülecek şekilde bir tasarım yapılırsa , uygulamanın toplam yürütme gecikmesi %47 daha kısa sürede tamamlanmış olacaktı. Bu uygulama için mobil cihaz sadece kendi kaynaklarını kullanarak tüm işleri tamamlamak için toplamda 100ms harcadığı varsayılırsa, bu senaryoda MEC sistemin getirdiği avantajla tüm işler 53ms de bitmiş olacaktır. Hassas cevap süresi gerektiren artırılmış gerçeklik (Agumented Reality) gibi mobil uygulamalar için kullanımı elverişli ve oldukça iyi bir kullanıcı deneyimi sağlayan MEC sistemi, yapılan bu simülasyon çalışmasında çok fazla iş yüküne sahip uygulamaların daha taşınabilir hale gelmesinde umut verici sonuçlar göstermektedir.

4. Sonuç

MEC sistemi mobil cihazlara daha yakın noktalarda konumlandırıldıklarından, MCC kaynaklarına göre gecikme süreleri bakımından daha avantajlıdır. Bu sayede yüksek işlem yüküne sahip uygulamaların optimum gecikmelerde, mobil cihaz yerine MEC sistemi ile değerlendirilerek yeni teknolojilerle birlikte gelen daha büyük ve son teknoloji uygulamaların mobil

tarafta da rahatça kullanılabilirliği yaygınlaşacaktır. MEC sistemi ayrıca cihazların enerji tüketimini azaltarak, pil ömürlerinin daha uzun süreli olmasına imkân sağlar. Yapılan bu çalışma ışığında bu sistemin taşınabilir cihazlara büyük avantajlar getireceği öngörülmektedir. 5G ağ teknolojisinin sunduğu yüksek bant genişliği ve kapsama alanı ile birlikte uygun bir altyapıya sahip olan bu sistemin gelişimi, yeni ağ teknolojileriyle giderek daha büyük bir kullanıcı havuzuna hitap etmesi beklenmektedir. Daha kısa gecikme süreleri veya daha az enerji tüketimi gibi mobil cihazların karşı karşıya kaldığı zorluklara çözüm getiren yeni çalışmalar ile birlikte, yıllar içinde MEC sisteminin daha olgun ve gelişmiş şekilde, mobil sistemlerde büyük bir araştırma konusu olacağı tahmin edilmektedir. Araştırma konusu olarak offloading kararları, kaynak tahsisi ve hareketlilik yönetimi gibi temel MEC sisteminin işleyişini doğrudan ilgilendiren konuların gelişimi gelecek çalışmalarda MEC için kritik önem taşımaktadır.

Kaynakça

[1] Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. Future Generation Computer Systems, 29(1), 84-106. doi:10.1016/j.future.2012.05.023

[2] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 393–413, 1st Quart., 2014.M. Armbrust et al., "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 52–58, Apr. 2010.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A servey on mobile edge computing: The communication perspective," to appear IEEE Commun. Survey Tuts. 2017. [Online]. Available: https://arxiv.org/ abs/1701.01090

[4] M. Agiwal, A. Roy and N. Saxena, "Next Generation 5G Wireless Networks: A Comprehensive Survey," IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 1617-1655, Third Quarter 2016.

[5] J. G. Andrews et al., "What will 5G be?" IEEE J. Sel. Areas Commun., vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[6] "Understanding 5G: Perspectives on future technological advancements in mobile," GSMA Intell.,London,U.K.,Dec.2014.[Online].Available:https://www.gsmaintelligence.com/research/?file=141208-5g.pdf&download

[7] "Mobile-edge computing—Introductory technical white paper," White Paper, ETSI, Sophia Antipolis, France,Sep.2014.[Online].Available:https://portal.etsi.org/portals/0/tbpages/mec/docs/mobileedge_computing_introductory_technical_white_paper_v1%2018-09-14.pdf

[8] S. Wang et al., "A survey on mobile edge networks: Convergence of computing, caching and communications," IEEE Access, vol.5, pp. 6757–6779, 2017.

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Commun. Surveys Tuts., vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[10] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," White Paper, ETSI, Sophia Antipolis, France, 2015.

[11] "Mobile edge computing use cases & deployment options," White Paper, Juniper, Sunnyvale,CA,USA,Jul.2016.[Online]Available:https://www.juniper.net/assets/us/en/local/pdf/whitepapers/000642-en.pdf

[12] C.-Y. Chang, K. Alexandris, N. Nikaein, K. Katsalis, and T. Spyropoulos, "MEC architectural implications for LTE/LTE-A networks," in Proc. ACM Workshop Mobility Evol. Internet Archit. (MobiArch), New York, NY, USA, Oct. 2016, pp. 13–18.

[13] C.-Y. Chang, K. Alexandris, N. Nikaein, K. Katsalis, and T. Spyropoulos, "MEC architectural implications for LTE/LTE-A networks," in Proc. ACM Workshop Mobility Evol. Internet Archit. (MobiArch), New York, NY, USA, Oct. 2016, pp. 13–18.

[14] "Mobile edge computing use cases & deployment options," White Paper, Juniper, Sunnyvale,CA,USA,Jul.2016.[Online]Available:https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf

[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE/ACM Trans. Netw., vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[16] B. Shi, J. Yang, Z. Huang, and P. Hui, "Offloading guidelines for augmented reality applications on wearable devices," in Proc. ACM Int. Symp. Multimedia, Brisbane, QLD, Australia, Oct. 2015, pp. 1271–1274.

[17] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in Proc. IEEE Annu. Consum. Commun. Netw. Conf. (CNCC), Las Vegas, NV, USA, Jan. 2017, pp. 160–164.

[18] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," IEEE Trans. Cloud Comput., to be published.

[19] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," IEEE Wireless Commun. Lett., vol. 6, no. 3, pp. 398–401, Jun. 2017

[20] H. Liu et al., "Mobile edge cloud system: Architectures, challenges,and approaches," IEEE Syst. J., to be published.

[21] S. Vakilinia, M. M. Ali, and D. Qiu, "Modeling of the resource allocation in cloud computing centers," Comput. Netw., vol. 91, pp. 453–470, Nov. 2015.

[22] S. Tekinay and B. Jabbary, "Handover and channel assignment in mobile cellular networks," IEEE Commun. Mag., vol. 29, pp. 42–46, Nov. 1991.

[23] G. P. Pollini, "Trends in handover design," IEEE Commun.Mag., vol. 34, pp. 82–90, Mar. 1996.

[24] V.Ateş and M. A. Akcayol, "Kablosuz Ağlarda Tahmine Dayalı Hücreler Arası GeçiĢ Algoritmaları", International journal of informatics technologies, 3.3, 2010.

[25] S. Wang et al., "Mobility-induced service migration in mobile microclouds," in Proc. IEEE Mil. Commun. Conf. (MILCOM), Baltimore, MD, USA, Oct. 2014, pp. 835–840

[26] R. Urgaonkar et al., "Dynamic service migration and workload scheduling in edge-clouds," Perform. Eval., vol. 91, pp. 205–228, Sep. 2015.

[27] N. Vastardis and K. Yang, "An enhanced community-based mobility model for distributed mobile social networks," J. Ambient Intell.Humanized Comput., vol. 5, no. 1, pp. 65–75, Feb. 2014.

[28] Google (2020). gRPC. Retrieved 23 October 2020, from https://grpc.io/

[29] Google (2015, March 1). gRPC.GRPC. https://developers.google.com/protocol-buffer