Yerel ve Hızlı Bulut Servisi: Bulutçuklar

Emre Çalışır, Gülfem Işıklar Alptekin, Atay Özgövde
Galatasaray Üniversitesi, Bilgisayar Mühendisliği, İstanbul

emrecalisir@gmail.com
gisiklar@gsu.edu.tr
aozgovde@gsu.edu.tr

Özet

Günümüzde kullanıcı eğilimleri her an ve her yerden bilgiye ulaşmaya doğru hızla ilerlemektedir. Teknoloji açısından bakıldığında, geliştirilen ürün ve servislerin de kullanıcı eğilimlerine yönelik yapılanması kaçınılmazdır. Bu bağlamda mobil/giyilebilir cihazların giderek yaygınlaşmasını izlemekteyiz. Mobil cihazlar doğaları gereği ufak boyutludurlar ve kaynakları kısıtlıdır. Mobil uygulamaların gerçek güçlerine ulaşması, hesaplama gücü ve pil ömrü gibi kısıtların ne ölçüde aşılabileceğine bağlıdır. Örneğin yüz ve ses tanıma, doğal dil işleme gibi gerçek zamanlı yoğun hesaplama gerektiren uygulamaları doğrudan mobil cihazlarda koşmak mümkün olamamaktadır. Olası bir yaklaşım, gün geçtikçe gelişen 'Bulut Hesaplama' altyapısını kullanarak mobil cihazların kendi üzerilerinde koşması beklenen hesaplamanın bir kısmını bulut kaynaklarına devretmesidir. Ancak bu durumda da, yüksek ve değişken bir ağ gecikmesiyle karşılaşılmaktadır. Bu duruma çözüm olarak önerilen bulutçuklar (Cloudlet), akademik dizinde mobil cihazın coğrafi olarak yakınında konumlanan ve yerel ağ hızlarında bağlanılan yerel bulut servisi olarak tanımlanmıştır. Kavramsal olarak önemli bir çözüm teşkil etse de bulutçuk tipi hesaplamanın hangi yöntemler ile yerelde yap-boz şekilde çalıştırılabileceği önemli akademik ve mühendislik sorunları içermektedir. Bu çalışmada bulutçuk kavramı tanıtılmış, akademik yazında bulutçuk ile ilgili önerilen mimari gerçekleme yöntemleri tanıtılmış ve karşılaştırmalı olarak incelenmiştir.

Anahtar Kelimeler

Bulut bilişim, bulutçuk, hesaplama transferi, mobil hesaplama

1. Giriş

Son yıllarda dağıtık sistemler ve yaygın hesaplama alanlarında belirgin bir paradigma değişikliği göze çarpmaktadır: Bir yandan hesaplama aygıtları gittikçe küçülmekte, insan yaşamı ve bedeni ile bütünleşmektedir, diğer yandan ise, bulut hesaplama sistemleri sayesinde, hesaplamanın kendisi esnek olarak merkezden sunulan bir meta haline gelmektedir. Böyle bakıldığında, hesaplamanın bir taraftan güçsüz mobil donanımlara yayıldığını, diğer bir taraftan ise gittikçe merkezileştiğini görmekteyiz. Bir çelişki gibi gözüken bu iki eğilimi beraberce anlamlandırmak, bu çalışmada aktarılan *bulutçuk* kavramını bir perspektife oturtabilmek için gereklidir. Bahsi geçen iki eğilim aslında birbiriyle çelişen değil, birbirini tamamlayan iki eğilim olarak görülmelidir.

Cisco'nun bir araştırmasına göre, dünyada uygulama başına mobil veri trafiğinin 2017'de 2011 yılına kıyasla %66 artması beklenmektedir [1]. Acaba sayıları ve kullandığı veri trafiği devamlı artan mobil donanımların hesaplama kısıtları bulut hesaplama olanakları ile aşılabilir mi? Hem her yerde yaygın bir şekilde (*ubiquituous*) kullanılacak kadar ufak form faktörlü olup, hem de sanki merkezde bir sunucuda çalışıyormuşçasına yoğun hesaplama kabiliyetine sahip olabilirler mi? Şu anki bulut mimarisinde ağ gecikmeleri sebebiyle gerçekleşemeyen bu ideal durumu yakalama gayreti olarak, son zamanlarda ortaya sürülen kavramlardan en öne çıkanı *bulutçuk* (cloudlet) olmuştur.

Bulutçuk, kendisinden hesaplama hizmeti alacak düğümlere coğrafi olarak yakın konumda bulunan mini veya mikro veri merkezleri olarak tanımlanabilir (Şekil 1).

Bulutçuk yaklaşımında düğümlerin (mobil donanımın) kaynak sıkıntısı, uzaktaki buluta bağlanmak yerine kendine yakın ve kaynak bakımından zengin olan bulutçuğa bağlanmasının yardımıyla çözülebilir. Bir bulutçuk, mobil donanımların veya merkezi bilgisayarların oluşturduğu geçici (*ad-hoc)* ağlardan oluşturulabileceği gibi, sadece hesaplama transferi yapılması için ortama koyulmuş yüksek işlem güçlü bir bilgisayardan da oluşturulabilir.

Bulut bilişimden faydalanırken, geniş ağ bağlantısının limitlerini aşmanın yollarından biri bulutçuk kullanmak olabilir. Bu yöntemin güçlü yanları şöyle özetlenebilir:

- Bulutçuk servis verdiği mobil cihaza tek adım (*hop*) uzaklıktadır.
- Mobil donanım ile bulutçuk arasında yüksek bant genişliğine sahip güvenli kablosuz bir ağ bağlantısı kurulduğu için, hem oldukça düşük hem de kestirebilir gecikme dağılımları ile çalışılır.
- Mobil donanım ve bulutçuk arasındaki bağlantının internete bağlanma zorunluluğu yoktur.

Şekil 1- Bulutçuk kullanımı örnek senaryosu

Tablo 1, bulutçuk ile bulut arasındaki en temel farklılıkları özetlemektedir.

Tablo 1 - Bulutçuk ve bulut sistemleri arasındaki temel farklar

	Bulutçuk	**Bulut**
Durum	Yazılım	Donanım ve yazılım
Yönetim	Kendi kendine kontrol edilebilir. Çok az yönetim ihtiyacı.	Profesyonel olarak yönetiliyor. 7/24 çalışıyor.
Ortam	Kutu içinde bir veri merkezi	Soğutma ve havalandırmalı makine odaları
Sahibi	Dağıtılmaya ve yerel şirketler tarafından kullanılmaya uygun.	Amazon, Yahoo gibi şirketlerin merkezileştirdiği kendi ortamları.
Ağ	LAN seviyesinde gecikme	WAN seviyesinde gecikme
Paylaşım	Aynı anda az sayıda kişi	Aynı anda yüzlerce/ binlerce kişi

Akademik yazın incelendiğinde bulutçuk kullanım senaryolarının belirlenmesinin ve ihtiyaç analizinin yapılmakta olduğu görülmektedir. Ancak özellikle yazılım açısından nasıl bir yaklaşım izleneceği ve pratik dünyada bulutçuk çalışma modelinin ne olacağı henüz oldukça belirsizdir. Bu çalışmada hali hazırda öne çıkan temel yaklaşımları aktarmak ve karşılaştırmalı tartışmak amaçlanmıştır.

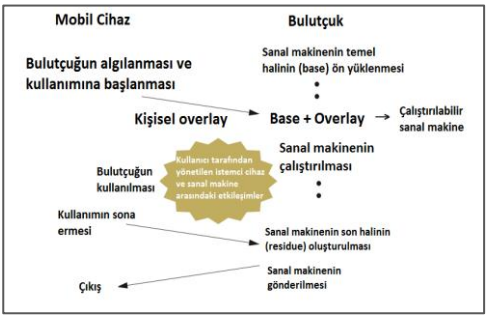
2. Bulutçuk Yaklaşımları

Bulutçuk ile ilgili farklı yaklaşımları içeren akademik çalışmalar bulunmaktadır. Sonraki bölümlerde anlatılan yaklaşımlar, hesaplama transferinin hangi kısımlarının nerede ve nasıl yapılacağı kararına ilişkin farklar içermektedir. Yaklaşımlardan birinde, mobil donanım basit istemci (*thin client*), bulutçuk ise tüm önemli hesaplamaların transfer edileceği yakındaki bir donanım olarak düşünülmüşken, bir diğer yaklaşımda mobil donanım uygulamanın temel bileşenlerini kendi üzerinde işlerken, sadece kendi işleyemeyeceği bileşenleri bulutçukta çalıştırmaktadır. Akademik dizinde önerilen çeşitli bulutçuk yaklaşımları ilerleyen bölümlerde incelenmiştir.

2.1. Sanal Makine Seviyesinde Hesaplama Transferi Yaklaşımı

2009 yılında Carnegie Mellon Üniversitesi'nde geliştirilen 'dinamik sanal makine sentezi' yaklaşımı, önerilen ilk bulutçuk yaklaşımı olarak kabul edilebilir [2]. Bu çalışmada bulutçuk, güvenilir, kaynak bakımından zengin, internet bağlantısı hızlı, mobil kullanıcıların yakınında olduğundan kullanıma hazır olan bir veya birden çok bilgisayardan oluşan ve kullanıcıya bir adım (*single hop*) uzakta olan bir sistem olarak tanımlanmıştır. Dinamik sanal makine sentezi, adından da anlaşılacağı gibi sanal makine teknolojisine dayanmaktadır. Bu teknolojide, mobil kullanıcı kendisiyle aynı ortamda bulunan yüksek işlem kapasiteli bulutçuğa, kablosuz ağ bağlantısı üzerinden bir sanal makine bindirimi (*overlay*) gönderir (Şekil-2). Bindirim, sanal makinenin taze kurulduğu andaki anlık durum görüntüsü (*snapshot*) ile bulutçuk üzerinden yapılmak istenen hesaplamanın çalıştırılabilir kod seviyesinde barındırıldığı sanal makinenin anlık durum görüntüsünün ikili tabanda alınmış farkını içeren dosyasıdır. Sanallaştırma teknolojileri bu bindirimleri ön tanımlı sanal makineler üzerine etki ettirmeye (bir nevi yamamaya) izin vermektedir. Bu durumda bindirim kısmını elinde tutan mobil aygıt bunu bulutçuğa göndererek istediği hesaplama işlevine sahip bir sanal makineyi gene bulutçuk üzerinde yaratabilmektedir. Bu yaklaşımın püf noktası bindirimin (overlay) boyutunun, sanal makine boyutuyla karşılaştırıldığında çok küçük olmasıdır. Buna göre bulutçuğa sanal makinenin tamamı değil, sadece bindirim (overlay) gönderilir ve bu

bindirimin boyutu bulutçukta işlenecek uygulamanın türüne göre değişmektedir. Bulutçuk, bu bindirimi kendisinde önceden bulunan ilgili ön tanımlı sanal makine üzerinde çalıştırdıktan sonra gerekli hesaplamaları yapar ve işlemi bittikten sonra sanal makinenin son halini temsil eden bindirimi (residue) kullanıcıya geri gönderir. Dinamik sanal makine sentezi, 60-90 saniye arasında sürmektedir [2]. Bu da basit ya da geçici işler için kabul edilebilir seviyenin üzerinde bir gecikmedir. Satyanarayanan'ın 2012 yılındaki çalışmasıyla birlikte bu sürenin çeşitli optimizasyonlar yardımıyla 20-30 saniye süresine indirilebildiği gösterilmiştir [3].



Şekil 2- Dinamik sanal makine sentezi yaklaşımı

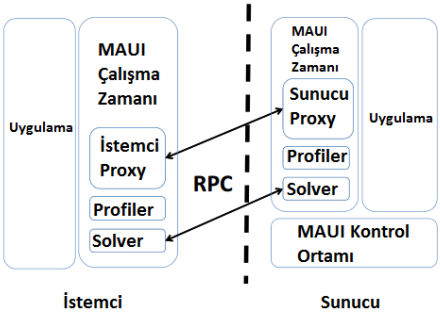
2.2. Metot Seviyesinde (RPC) Hesaplama Transferi Yaklaşımı

Hesaplama transferi yaklaşımlarının bir diğeri ise, MAUI mimarisidir [4]. MAUI yaklaşımından önce, başlıca iki temel yaklaşım bulunmaktaydı:

1. Hesaplamanın tamamını bulutçuğa transfer edip orada işlemek (dinamik sanal makine sentezi)
2. Uygulamayı bileşenlere ayırarak, yoğun kaynak kullanımı gerektiren bileşenleri bulutçukta işlemek.

MAUI ise, sahip olduğu ince ayarlı (fine-grained) kod transferi yeteneği sayesinde enerji tasarrufunuençoklarken, uygulamanın geliştirilme esnasında asgari miktarda kod değişikliği yapılmasını hedeflemiştir. MAUI mimarisinde uygulamanın iki farklı sürümü bulunur. Bunlardan biri istemci cihazda çalışırken, diğeri uzak MAUI sunucusunda çalışır (Şekil 3).

MAUI mimarisindeki en önemli fonksiyonlardan birini, optimizasyon aracı sağlamaktadır. Uygulamanın çalışma esnasında, bir uzak (*remote*) metot algılandığında optimizasyon aracı devreye girer. Bu araç, bağlantı kısıtlarını göz önünde bulundurarak uygulamanın en az enerji harcayarak çalışabilmesi doğrultusunda, uzak metodun nerede çalıştırılacağına karar verir.



Şekil 3- Metot seviyesinde (RPC) hesaplama transferi yaklaşımı - MAUI

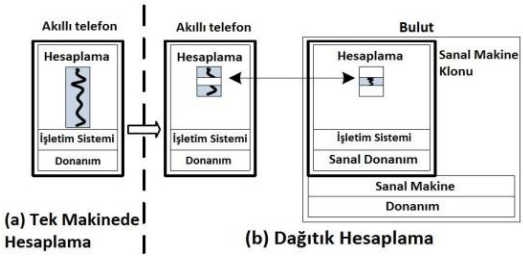
Bunun yanı sıra, uzak sunucuda işlenmiş ve istemciye dönmüş olan metodun transferine ait bilgiler de optimizasyon aracı tarafından toplanır. Bu bilgiler ışığında, bir sonraki uzak metodun transfer edilip edilmeyeceğine karar verilir. Eğer metot uzak sunucuda çalıştığı sırada bağlantı koparsa, bu metodun istemci cihazda çalışması sağlanır.

MAUI mimarisinde, Microsoft .NET Common Language Runtime kullanıldığı için, uygulama .NET uyumlu mimarilerde sorunsuz çalışabilmektedir. Bunun yanısıra MAUI, yazılımcının uzak (*remote*) olarak tanımladığı tüm metotları otomatik olarak

algılayarak, sadece bu metodların uzak sunucuda işlenmesini sağlamaktadır.
Ayrıca MAUI, kullandığı "serializable" fonksiyonu sayesinde uygulamanın çalışma esnasında transfer edilecek metodunun ağda taşınma maliyetini belirleyerek bunu bir doğrusal programlama formülünde kullanmaktadır.

2.3. Metot Seviyesinde (Parçacık) Hesaplama Transferi Yaklaşımı

Mobil cihazların kendi hesaplamalarını bulutçuğa delege ederek daha performanslı ve daha az enerji tüketerek çalışmasını sağlayan bir diğer mimari örneği de 'Clone Cloud' yaklaşımıdır [5].



Şekil 4- Metot seviyesinde (parçacık) hesaplama transferi yaklaşımı - Clone Cloud

Bu mimaride, istemci tarafında çalışan uygulamanın çalışma esnasında yaptığı analizler sonucunda, fazla kaynak kullanımı gerektiren parçacıkların olduğu yerler (transfer noktaları) belirlenir. Uygulamaya ait bu noktalardaki parçacıklar, bulutta oluşturulan cihazın sanal bir kopyasına işlenmek üzere transfer edilir (Şekil 4).

Clone Cloud altyapısı, Java uygulama katmanındaki Dalvik sanal makinesinin üzerinde çalışmaktadır. Orta katman sanal makinesi tercih edilmesinin sebebi; bu sanal makinenin hem mobil platformlarda yaygın olarak kullanılması, hem de uygulamanın gerekli kısımlarının platformdan bağımsız taşınabilmesidir. Clone Cloud mimarisinde uygulamanın çalışma zamanında parçacıkların taşınacağı noktalar belirlenir; bu noktalara göç noktaları denmektedir. Uygulama çalışırken bu noktalara gelindiğinde buradaki bağımsız parçacıklar (thread), o esnada sahip oldukları verilerle birlikte (sanal durum, program sayıcı, saklayıcılar ve yığın) uzak sunucuda işlenmek üzere transfer edilir. Bu transfer işlemi sonrasında, istemcideki çalışma bloke olmadan devam ederken, uzak bilgisayarda bulunan klonlanmış sanal makinede ise transfer edilmiş parçacıklar işlenmektedir. Transfer edilen parçacık, sunucunun işlemci ve bağlantı yeteneklerini kullanarak işlendikten sonra, istemci cihaza geri transfer edilerek parçacığın transferden önceki durumuyla işlendikten sonraki durumu birleştirilir.

Clone Cloud mimarisinin içerdiği statik analiz ve dinamik ayrımlama (profiling) eklentileri, uygulamanın hangi kısımlarda hesaplamayı transfer etmesi gerektiğini belirleyebilmektedir. Dolayısıyla, uygulama geliştiriciler transferin nerede yapılacağını belirlemek ve ona göre geliştirme yapmak zorunda kalmamaktadır.

3. Sonuç ve Tartışma

Bu çalışmada, mobil donanımların yeteneklerini ve performanslarını artırmaya yönelik önerilen *bulutçuk* kavramı irdelenmiş ve bu alandaki çalışmaların önde gelenlerinden olan dinamik sanal makine sentezi, MAUI ve Clone Cloud yaklaşımları gösterilmiştir. Dinamik sanal makine sentezi, bulutçuk kavramı üzerine yapılan ilk çalışmalarından olduğu için önemlidir. Bu yöntemi kullanmanın olumlu yönleri olduğu gibi, olumsuz yönleri de vardır.

Dinamik sanal makine sentezindeki kısıtlardan biri, bulutçuğu sağlayan servis sağlayıcının sunduğu donanım ve yazılımlara bağımlı kalınmasıdır. Bir diğer kısıt ise, bulutçukta yoğun kaynak kullanımı gerektiren bir işlem sürerken, istemci bilgisayarda çalışılamamasıdır. Bulutçuğa aynı anda birçok kullanıcının bağlandığını düşünülürse, her kullanıcının bulutçukla sanal makine transferi yapması da, bulutçukta ağır yük meydana getirmektedir. MAUI ve Clone Cloud mimarilerinde ise, sanal makine sentezine kıyasla mobil donanım bulutçuktaki işlemin bitmesini beklemeden çalışmasına devam edebilmektedir. Bu yaklaşımlarda metot seviyesinde veri transferi olduğu için sanal makine bindirimine göre daha etkin sonuçlar alınabilmektedir. Tablo 2, üç mimarinin belirli kriterler ışığında karşılaştırmasını içermektedir.

Tablo 2 - Hesaplama transferi yaklaşımlarının karşılaştırılması

İncelenen Alan	Dinamik Sanal Makine Sentezi	MAUI	Clone Cloud
Transfer Yöntemi	Sanal makine	Metot (Uzak prosedür çağrısı)	Metot (Parçacık durdurma ve devam etme)
Geliştirme Ortamı	Linux	Microsoft .NET Framework	Android (Dalvik VM)
Taşınma Noktalarına Karar Verilmesi	-	Uygulama geliştirilirken	Hiçbir zaman
Kullanıcı Ortamı	Linux	Her ortam	Her ortam
İstemcinin Bloke Olması	Evet	Hayır	Hayır

4. Referanslar

[1] Cisco VNI, 2013, A.T. Kearney Analysis.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing, Pervasive Computing", IEEE Pervasive Computing 8 (4), 2009, 14-23.

[3] Sarah Clinchy, Jan Harkesz, Adrian Fridayy, Nigel Daviesy, Mahadev Satyanarayanan, "How Close is Close Enough? Understanding the Role of Cloudlets in Supporting Display Appropriation by Mobile Users", In Proc: IEEE International Conference on Pervasive Computing and Communications (PerCom), 2012, 122-127.

[4] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, Paramvir Bahl. "MAUI: making smartphones last longer with code offload", In Proc: MobiSys, 2010, 49-62

[5] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, Ashwin Patti, "Clonecloud: Elastic execution between mobile device and cloud", In Proc: 6th Conference on Computer Systems (EuroSys '11), 2011, 301-314.