Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-15
Sayı-1
2022

Bulut Uygulamalarında Evrensel Duyarlılık Analizi

Global Sensitivity Analysis in Cloud Applications

Hüseyin Kaya
Payten Teknoloji A.Ş.
İstanbul, Türkiye
huseyin.kaya@payten.com
ORCID: 0000-0002-6481-5431

Öz

Beş milyarı aşkın internet kullanıcısının bulunduğu, her iki alışverişten birinin internet üzerinden gerçekleştiği günümüz dünyasında, küresel ticaretin ana yükünü bulut uygulamaları çekmektedir. Bu nedenle, son kullanıcılara güvenilir, hızlı ve kesintisiz hizmet sağlanması ve artan rekabet ortamında maliyetlerin düşürülebilmesi için bulut uygulamalarının kopya sayısının, kaynaklarının ve uygulama içi parametrelerinin en uygun şekilde seçilmesi oldukça önemlidir. Parametre seçimini belirleyen önemli unsurlardan birisi ise parametrelerin bulut uygulamasının performans ve maliyet üzerindeki etkisinin nesnel olarak ölçülebilmesidir. Bu çalışmada, bahsedilen ölçümlerin Yüksek Boyutlu Model Gösterilimi adlı evrensel duyarlılık analizi yöntemi ile nasıl yapılabileceği tarif edilmiş ve önerilen yaklaşımın uygulanabilirliği TeaStore adlı örnek bir bulut uygulaması üzerinde deneysel olarak gösterilmiştir. Bulut uygulamaları alanında ilk kez yapılan bu çalışma sayesinde, öncelik verilecek veya ihmal edilecek kadar önemsiz parametrelerin belirlenmesi sağlanmaktadır. Bunun sonucu olarak bulut uygulamalarının yönetiminde kullanılan araçların iyileştirilmesi mümkün olabilecektir.

Anahtar sözcükler: Bulut Uygulamaları, Evrensel Duyarlılık Analizi, Yüksek Boyutlu Model Gösterilimi

Abstract

In today's world, where there are more than five billion internet users and one of every two shopping is done over the internet, cloud applications are the main burden of global trade. For this reason, it is very important to choose the optimal number of replicas, resources and in-application parameters of cloud applications in order to provide reliable, fast and uninterrupted service to end users and to reduce costs in an increasingly competitive environment. One of the important factors in parameter selection is the objective measurement of the effect of the parameters on the performance and cost of the cloud applications. In this study, it is described how the aforementioned measurements can be made with the global sensitivity analysis method called High-Dimensional Model Representation, and the applicability of

the proposed approach is experimentally demonstrated on a benchmark cloud application called TeaStore. Thanks to this study, which is the first in the field of cloud applications, it is possible to determine the high-priority parameters or insignificant ones to be neglected. As a result, it will be possible to improve the tools used in the management of cloud applications.

Keywords: Cloud Applications, Global Sensitivity Analysis, High Dimensional Model Representation

Gönderme, düzeltme ve kabul tarihi: 22.03.2022 -31.05.2022 - 05.06.2022
Makale türü: Araştırma

1. Giriş

İçinde bulunduğumuz İnternet çağının en çok kullanılan kavramlarından birisi olan bulut, dünyaya yayılmış, birbiri ile bağlantılı yüzbinlerce veri merkezi ve bunların içindeki milyonlarca fiziksel ve sanal sunucudan oluşan devasa bir depolama ve hesaplama havuzudur [1]. Son kullanıcılar açısından yekpare ve yalın bir sistem gibi görünse de bulut servisleri, kesintisiz ve hızlı hizmet verebilmek adına birden fazla kopya (İng. replica) üzerinde çalıştırılır. Ayrıca uygulamaya ayrılan kaynak miktarı ve uygulama ince ayarları gibi diğer birçok parametre, yedekleme, performans, maliyet iyileştirmesi ve benzeri sebeplerle durmaksızın değiştirilir. Ölçekleme terimi altında toplanabilecek bu faaliyetleri otomatikleştirmeyi hedefleyen birçok otomatik ölçekleme yaklaşımı bulunmaktadır [2,3].

Bunlardan en bilineni bulut uygulamalarının kopya sayısının değiştirilmesi anlamına gelen yatay ölçekleme'dir (İng. horizontal scaling) [4]. Buna göre uygulamaya gelen istek sayısına bağlı olarak kopya sayısı arttırılır veya azaltılır. Yatay ölçeklemede bütün kopyaların aynı kaynak sınırlamasına tabi

olması sağlanır. Böylece bir kopyanın diğerinden daha yavaş/hızlı olması durumunda kopyalar arasındaki yükün dengesiz dağılmasının önüne geçilir. Eğer özellikle belirtilmemişse, ölçekleme denilince kastedilen genellikle yatay ölçeklemedir. Bu eğilimin doğal sonucu olarak, otomatik ölçekleme yaklaşımlarının büyük çoğunluğu da aslında yatay ölçekleme odaklıdır [5].

Maliyet veya performans eniyilemesi amacıyla uygulama kaynaklarının değiştirilmesi olarak tanımlanabilecek dikey ölçekleme (İng. vertical scaling) ise, uygulamaya ayrılan işlemci, bellek ve ağ kapasitesi benzeri kaynakların sınırlarının arttırılıp/azaltılması ile sağlanır [6]. Her ne kadar amacı aynı olsa da dikey ölçekleme yataydan birkaç noktada belirgin bir şekilde ayrışmaktadır. İlk olarak dikey ölçekleme tanımı gereği, uygulama kaynaklarına müdahale ettiği için, her ölçekleme sonrasında bütün kopyaların kapatılıp, belirlenen sınırlarla yeniden açılması gereklidir. Bu ise ölçekleme kaynaklı servis kesintilerine neden olabilmektedir. Bu sorunu, ölçekleme hızını bir miktar yavaşlatma pahasına dahi olsa, kopyaları teker teker güncelleyerek aşmak mümkündür. Öte yandan dikey ölçekleme, kaynak kullanım sınırlarını devreye sokarak, yatay ölçekleme ile elde edilemeyecek ölçüde bir hareket serbestliği getirmekte, bu sayede performans ve maliyet iyileştirmesinin daha hassas yapılabilmesini sağlamaktadır. Bu nedenle, bulut bilişim sektöründe dikey ölçeklemeye olan ilgi giderek artmaktadır [7].

Bulut uygulamalarının yönetiminde en az ölçekleme kadar kritik öneme sahip bir diğer nokta ise uygulama içi parametrelerin en uygun bir şekilde belirlenmesidir [8]. Uygulamanın kullandığı parametre sayısına ve alabileceği değerlere bağlı olarak, en uygun parametreler basit bir tarama yöntemi veya kısıtlamasız eniyileme yöntemi ile belirlenebilir. Bu işlem yaygın olarak performans eniyilemesi (İng. performance optimization) adıyla bilinmekle birlikte amaç olarak performans yerine maliyet de kullanılabilir. Bu nedenle amacın genel niteliğini yansıtabilmek için bu çalışmada bulut uygulamalarının parametre eniyilemesi ince ayar (İng. auto-tuning, fine-tuning) olarak adlandırılmıştır [9].

Birkaç kuraldan oluşan en sade ölçekleme yazılımından, derin öğrenme ile desteklenmiş pekiştirmeli öğrenme yaklaşımına kadar geniş bir yelpazede yer alan bütün ölçekleme ve ince ayar yöntemleri, parametre ve maliyet/performans arasında var olduğu düşünülen ilişki ile ilgili bazı kabuller yapar ve/veya bu ilişkiyi modellemeye çalışır. Sözgelimi yatay ölçekleme çözümleri, kopya sayısı ile performans/maliyet arasında doğrusal bir ilişki olduğu savına göre çalışır ki genellikle durum zaten böyledir [10]. Ancak uygulamaya gelen yükün durumuna veya kopyalar arasındaki haberleşme topolojisine bağlı olarak kopya sayısını arttırmak performansı her zaman beklendiği kadar düzeltmeyebilir ve hatta nadir olarak düşürebilir. Bu nedenlerle ölçekleme ve ince ayara konu olan parametrelerin, uygulama performansını/maliyetini nasıl etkilediğini incelemek son derece önemlidir. Parametrelerin performans/maliyeti, bireysel (İng. individual) veya topluca (İng. cooperative) nasıl etkilediğinin nicel olarak belirlenmesi sayesinde mevcut ölçekleme yöntemleri iyileştirilebilir veya duyarlılık analizine dayanarak yeni ölçekleme yöntemleri türetilebilir. Bu doğrultuda, literatürde ilk kez olarak bu çalışmada, evrensel duyarlılık analizinin (İng. global sensitivity analysis), yatay/dikey ölçekleme ve ince ayar ile ilgili bütün parametreleri kapsayacak şekilde kullanılabileceği ortaya atılmış ve örnek bir bulut uygulamasında deneysel olarak gösterilmiştir.

Bu makalenin ikinci bölümünde, evrensel duyarlılık analizi ve bu amaçla kullanılacak Yüksek Boyutlu Model Gösterilimi yöntemi kısaca tanıtılacaktır. Üçüncü bölümde ise bulut uygulamalarında nasıl kullanılacağı anlatılacak ayrıca analizlerin yapılacağı örnek bulut uygulamasının kurulumu, sentetik yük üretimi, veri toplama safhaları ve duyarlılık analizi ile ilgili hesaplamaların detaylarına yer verilecektir. Sonuçların gösterildiği ve yorumlandığı son iki bölüm ile makale sonlandırılacaktır.

2. Evrensel Duyarlılık Analizi

Duyarlılık analizi yöntemleri, yerel (İng. *local*) ve evrensel (İng. *global*) olmak üzere iki temel sınıfa ayrılmaktadırlar [21]. Yerel yöntemler, parametre uzayının belli bölgeleri üzerine odaklanırken, evrensel yöntemler parametre uzayının tamamı üzerinde çalışmaktadır. Ölçekleme ve ince ayar bağlamında amacımız parametrelerin genel davranış kalıplarını çıkarmak olduğu için, bu çalışmada evrensel duyarlılık analizi özellikle tercih edilmiştir.

2.1 Yüksek Boyutlu Model Gösterilimi

Bu çalışmada, ölçekleme parametrelerinin evrensel duyarlılık analizini gerçekleştirmek için Yüksek Boyutlu Model Gösterilimi (YBMG) isimli yöntem kullanılmıştır [11]. Gösterilim, kısaca, n değişkenli herhangi bir fonksiyonun, değişken sayısı birden n'ye kadar değişen fonksiyonların toplamı olarak ifade edilmesi prensibine dayanır [12].

$$f(x_1,\dots,x_n) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i<j}^{n} f_{ij}(x_i,x_j) + \cdots + f_{12\cdots n}(x_1,\dots,x_n) \quad (1)$$

(1) eşitliğinin solunda yer alan f fonksiyonu, analitik yapısı bilinen açık bir model olabileceği gibi, iç yapısı tam olarak bilinmeyen kapalı kutu karmaşık bir model de olabilir. YBMG'nin en temel avantajı tüm değişken etkileşimlerini içeren orijinal modeli, parçalara ayırarak değişken etkileşimlerinin kademeli olarak arttığı çok sayıda fonksiyonun toplamı olarak ifade etmesidir. Sözgelimi, $f_1(x_1)$ terimi x_1 değişkeninin model çıktısı üzerindeki bireysel etkisini göstermekteyken, $f_{12}(x_1,x_2)$ terimi x_1 ve x_2 değişkenlerinin bir arada modele yaptığı katkıyı göstermektedir. Terimler arasındaki tek sabit olan f_0 ise modelin değişken bağımsız davranışını ifade etmektedir. Özetle, YBMG terimleri evrensel duyarlılık analizi için gereken bütün bilgiyi içermektedir.

YBMG açılımını yorumlamanın bir diğer yolu ise değişkenleri girdi (İng. *input*), f fonksiyonunu ise çıktı (İng. *output*) olarak ifade etmektedir. Böylece girdi/çıktı ilişkisinin kurulabildiği birçok model için YBMG'nin nasıl konumlandırılacağı daha açık olarak anlaşılmaktadır. Değişken etkileşimlerini birbirinden ayrıştırabilmek için YBMG terimlerinin dik olması

gereklidir. Bunu sağlamanın yolu ise aşağıdaki sıfırlanma varsayımıdır [13]:

$$j \in \{i_1, \dots, i_k\} \Rightarrow \int f_{i_1 i_2 \cdots i_k} dx_j = 0. \quad (2)$$

Diklik sayesinde terimleri, sabitten başlayarak kademeli olarak hesaplamak mümkün hale gelir. İlk olarak sabit terimi bulmak için (1) denkleminin her iki tarafının bütün değişkenlere göre tümlevi alınır. Sıfırlanma koşulu nedeniyle sabit terim dışındaki bütün terimler yok olur ve sabit terim (3) eşitliğindeki gibi elde edilir.

$$f_0 = \int \cdots \int f(x_1, \dots, x_n) \prod_k^n dx_k \quad (3)$$

Değişkenlerin bireysel etkilerini ifade eden tek değişkenli YBMG terimlerini bulmak için ise, (1) denkleminin bu kez ilgili değişken (etkisi belirlenmek istenen) dışında kalan tüm değişkenlere göre tümlevi alınır. Bu sayede bulmak istediğimiz terim dışındaki bütün YBMG terimleri yok olur.

$$f_i(x_i) = \int \cdots \int f(x_1, \dots, x_n) \prod_{k \neq i}^n dx_k - f_0 \quad (4)$$

İki ve üzeri dereceden YBMG terimlerini yukarıdaki stratejiyi kademeli olarak uygulayarak bulmak mümkündür. Bununla birlikte atmosferik fizikten, kimyaya kadar birçok alanda üçüncü ve üzeri derece etkileşimlerin modele katkısının, diğerlerine göre ihmal edilebilecek kadar az olduğu görülmektedir [14]. Dolayısı ile YBMG uygulamalarının birçoğunda bir veya ikinci mertebeden YBMG terimlerinin kullanılması genellikle yeterli olmaktadır. Bu çalışmada da sabit terim ve tek değişkenli terimlerin hesaplanması ile yetinilecektir.

Modelin analitik yapısı bilinmiyor veya tümlevler analitik olarak hesaplanamıyorsa YBMG terimlerinin herhangi bir noktadaki değerini bulmak için genellikle Monte Carlo sayısal tümlev hesaplama yöntemine başvurulur [15]. Noktanın konumu değiştirilerek YBMG terimleri çizelge (İng. *lookup tables*) şeklinde elde edilebilir. Değişken sayısının az olduğu durumlarda Monte Carlo yerine ızgara (İng. *grid*) yöntemi kullanılabilir. Buna göre her değişken ekseni belli sayıda dilimlere ayrılır ve sadece ızgaranın düğüm noktaları üzerinde ilgili model çalıştırılır [16]. Bu çalışmada da değişken sayısı dikkate alınarak ızgara yöntemi tercih edilmiştir.

2.2 YBMG ile Evrensel Duyarlılık Analizi

YBMG ile duyarlılık analizinin ilk adımı olarak (1) numaralı eşitlikteki sabit terim sol tarafa alınarak modelden çıkarılır:

$$f - f_0 = \sum_{i=1}^n f_i + \sum_{i<j}^n f_{ij} + \cdots + f_{12\dots n} \quad (5)$$

Ortalamanın (f_0) modelden çıkarılması modelin sıfır normalizasyonu anlamına gelmektedir. Modelin ortalamadan ne kadar saptığını ifade eden evrensel varyansı hesaplamak için ise normalize edilen modelin karesinin tümlevi alınır. YBMG terimlerinin dikliği sayesinde,

$$D \equiv \int f^2 - f_0^2 \quad \text{ve} \quad D_{i_1 \cdots i_k} \equiv \int f_{i_1 \cdots i_k}^2 \quad (6)$$

olmak üzere,

$$D = \sum D_i + \sum D_{ij} + \cdots + D_{12\dots n} \quad (7)$$

eşitliği elde edilir [12]. Yukarıdaki eşitlikte D, evrensel varyansı ifade ederken, eşitliğin sağındakiler YBMG terimlerinin varyanslarını göstermektedir. Sıfırlanma koşulunun bir gereği olarak bütün YBMG terimlerinin ortalaması sıfırdır, dolayısı ile karelerinin tümlevi terimlerin sıfır civarındaki salınımını ölçmektedir. Denklemin her iki tarafı evrensel varyansa bölünerek normalize edilebilir. Böylece Sobol katsayıları olarak da bilinen aşağıdaki duyarlılık katsayıları bulunur:

$$S_{i_1 \cdots i_k} \equiv \frac{D_{i_1 \cdots i_k}}{D} \quad \text{(duyarlılık katsayıları)} \quad (8)$$

Kolayca görülebileceği gibi bütün duyarlılık katsayılarının toplamı bire eşittir.

$$1 = \sum S_i + \sum S_{ij} + \cdots + S_{12\dots n} \quad (9)$$

Sıfır ile bir arasında değişen Sobol katsayıları, değişkenlerin model üzerindeki bireysel veya birlikte yaptıkları katkıların nicel ölçüsüdür. Bir değişken veya değişken grubuna karşılık gelen duyarlılık katsayısı ne kadar büyük ise model üzerindeki etkisinin o kadar fazla olduğu anlaşılır. Öte yandan sıfıra yakın katsayılar, ilgili değişkenlerin model üzerindeki katkısının ihmal edilebilecek kadar önemsiz olduğunu göstermektedir. Değişken etkileşimlerinin nispeten az olduğu sistemlerde, katsayıların toplamı hızlı bir şekilde bire yakınsar. Ayrıca katsayıların birbirine göre büyüklükleri karşılaştırılarak hangi değişken veya değişken grubunun model üzerinde daha etkili olduğu bulunabilir.

3. Önerilen Yaklaşım

Bu çalışma bulut uygulamalarının performans ve maliyetini etkileyen çeşitli parametrelerin evrensel duyarlılık analizi ile irdelenmesini konu almaktadır. Bu bağlamda bulut uygulamalarının yönetiminde rol oynayan parametreler girdi, uygulamaların performans veya maliyeti ise çıktı olarak değerlendirilmiştir. Parametreler ve performans/maliyet arasında var olduğu düşünülen girdi/çıktı ilişkisi ise YBMG ile modellenmiştir.

Çalışmanın etki alanını genişletmek adına seçilen parametrelerin çeşitliliğine özellikle dikkat edilmiştir. Bu doğrultuda yatay/dikey ölçekleme ve ince ayar alanlarından birer parametre seçilmiştir. Parametrelerin hangi YBMG değişkenlerine karşılık geldiği Çizelge-1'de gösterilmiştir. Aralarındaki birebir ilişki nedeniyle *parametre* ve *değişken* ifadeleri kimi zaman birbiri yerine kullanılsa da YBMG ve bulut uygulamaları bağlamında yapılan analizlerde sırasıyla *değişken* ve *parametre* ifadeleri özellikle tercih edilecektir.

Çizelge-1: YBMG değişkenleri ile uygulama parametreleri arasındaki eşleşme.

Değişken	Alan	Parametre	Birim
x_1	Yatay Ölçekleme	Kopya sayısı	adet
x_2	Dikey Ölçekleme	İşlemci sınırı	milicpu

x_3	İnce Ayar	Yığın sınırı	megabyte

Çizelge-1'de belirtilen parametrelerin her biri üç boyutlu Kartezyen koordinat sisteminde (bundan böyle kısaca *değişken* veya *parametre uzayı* olarak adlandırılacaktır), bir eksene karşılık gelmektedir. Dolayısı ile uygulamanın herhangi bir anda geçerli olan parametreleri parametre uzayında bir noktaya karşılık gelmektedir. Modelin bu nokta üzerinde aldığı değer ise, uygulamanın ilgili parametreler altında gösterdiği performansı veya maliyetini ifade etmektedir. Dolayısı ile model girdisi parametreleri, çıktısı ise uygulama performansı veya maliyetidir. Bulut uygulamalarının ve kurulu olduğu veri merkezlerinde bileşenlerin karmaşık yapısı nedeniyle modelin analitik yapısının bilinmemektedir.

Duyarlılık analizini yapabilmek için YBMG terimlerinin sayısal tümlev ile yaklaşık olarak hesaplanması gerekmektedir. Ancak bunun öncesinde her değişkenin alabileceği değerlerin, diğer bir ifade ile tanım kümesinin belirlenmesi gerekmektedir. Aralıklar belirlenirken ilgili parametrelerin görevleri dikkate alınmıştır. Sözgelimi, uygulamanın tamamen durmasına neden olacak sıfır kopya veya sıfır işlemci sınırı gibi değerler tanım kümesinin dışında bırakılmıştır. Son durumda Çizelge-2'de gösterilen tanım kümeleri belirlenmiştir.

Çizelge-2: Değişkenlerin alabileceği değerler.

Değişken	Tanım Kümesi	Eleman Sayısı (N_i)
x_1	$\{1,2,\dots,9\}$	9
x_2	$\{100,200,\dots,900\}$	9
x_3	$\{100,200,\dots,900\}$	9

3.1 Örnek Bulut Uygulaması

Bulut uygulamalarının yönetimi ve ölçeklenmesi ile ilgili araştırma geliştirme çalışmalarında ortak bir standart yakalamak, karşılaştırılabilirliği sağlamak ve bilgi paylaşımını kolaylaştırmak amacıyla hazırlanmış birtakım örnek (İng. *benchmark*) bulut uygulamaları bulunmaktadır. Bunlardan en yaygın ve güncel olanlarından birisi mikro servis mimarisi ile hazırlanmış *TeaStore* adlı uygulamadır [17]. Kap (İng. *container*) teknolojisini ve mikro servis mimarisini kullanan *TeaStore* uygulaması, yaygın bir kap yönetim sistemi olan *Kubernetes* ortamına kolaylıkla kurulabilmektedir [18]. Bu çalışmadaki analizler için *TeaStore* uygulamasının arayüzünü oluşturan *teastore-webui* mikro servisi kullanılmıştır. Bu noktadan sonra uygulama ifadesi ile TeaStore uygulamasının *teastore-webui* mikro servisi kastedilecektir.

3.2 Parametreleri Değiştirme Mekanizması

Bu çalışmada uygulamanın bütün parametreleri, *Kubernetes* üzerinde robotik otomasyon araçları geliştirmek için kullanılan *Operator-SDK* [19] aracılığı ile değiştirilmiş, performans ve maliyet ölçümleri de yine aynı platformun sunduğu imkânlar ile elde edilmiştir. Bu bağlamda *Golang* [20] tabanlı bir *Kubernetes* operatörü hazırlanmıştır. Uygulama parametreleri her *Kubernetes* operatöründe ön tanımlı olarak bulunan ve belirli aralıklarla *Kubernetes* tarafından çağrılan *Reconcile* fonksiyonu içinde değiştirilmiştir. Hazırlanan operatörün çalışma prensibi Şekil-1'deki sözde kod ile gösterilmiştir.

```
# Parametre değerlerini seç
replica = 2     # x_1
cpulimit = 300  # x_2
heaplimit = 400 # x_3
# Reconcile metodu, Kubernetes
# tarafından belirlik aralıklarla çağrılmaktadır.
function Reconcile(context, request)
   deployment = getDeployment("teastore")
   deployment.replica = replica
   deployment.cpulimit = cpulimit
   deployment.heaplimit = heaplimit
   deployment.update()
```

Şekil-2: *TeaStore* parametrelerin *Golang* tabanlı bir *Kubernetes* operatörü aracılığı ile nasıl değiştirilebildiğini gösteren sözde kod.

3.3 Değişken Uzayında Gezinme

Çizelde-2'den görülebileceği gibi bu çalışmada kullanılan değişken sayısı üçtür ($n = 3$) ve her değişken için dokuz farklı değer yer almaktadır ($N_1 = N_2 = N_3 = 9$). Bu nedenle, sayısal türevleri alabilmek adına parametre uzayı bir ızgara ile bölünmüştür. Bütün YBMG terimlerini hesaplayabilmek için modelin ızgaranın bütün noktalarında çalıştırılıp çıktıların elde edilmesi gereklidir. Bunu sağlamak için örnek bulut uygulamasının parametrelerini dilediği gibi değiştirebilen ve uygulama performansını ve maliyeti ölçebilen bir *Golang* programı yazılmıştır. Program ızgaranın herhangi bir düğümünden başlamakta ve rastgele seçtiği bir eksen doğrultusunda ilerlemektedir. Sınır dışına çıkmaya neden olacak bir seçim olursa, legal bir adım bulana kadar seçim tekrarlanmaktadır. Programın ayrıntıları altıncı bölümde belirtilen adreste verilmiştir.

3.4 Yük Üretme ve Performans Ölçümü

Değişken uzayındaki hareket şekli belirlendikten sonra her düğüm noktasında model çıktısının hesaplanması yani uygulama performansının/maliyetinin ölçülmesi gereklidir. Ölçümlerin tutarlı olması için ölçümler esnasında bulut uygulamasının iş yükünün sabit olması gereklidir. Bunu sağlamak için *TeaStore* uygulamasına *HTTP-Load-Generator* [21] adlı sentetik yük üreteci vasıtasıyla 100 istek/saniye hızıyla yük gönderilmiştir. Böylece uygulamanın arayüzünde aynı anda gezinti yapan çok sayıda müşterinin neden olacağı internet trafiği taklit edilmiştir. Performans ölçümleri sırasında yükün aniden kesilmemesi için, yük profilinin zaman aralığı bir ay gibi uzun bir süreye ayarlanmıştır. Yük gönderimi için kullanılan yük ve kullanıcı davranış profilleri hakkında ek bilgiler altıncı bölümde belirtilen adreste sunulmuştur.

Evrensel duyarlılık analizi öncesindeki son adım model çıktısını temsil eden uygulama performansının ölçümüdür. Bu çalışma için seçilen bulut uygulaması *Kubernetes* üzerine kurulduğu için, bu platformun metrik ölçme kabiliyetlerinden faydalanılmıştır. Özet olarak *Kubernetes*, üzerine kurulu bütün kapların anlık işlemci, bellek ve ağ kullanımını Prometheus [22] adlı bir bileşen aracılığı ile ölçmekte ve bir metrik havuzunda tutmaktadır. Depolanan metriklere PromQL [23] adlı özel bir filtreleme dili aracılığı ile

erişilmektedir. Bu çalışmada uygulama performansını temsilen, uygulamanın isteklere yanıt verme kapasitesi temel alınmıştır. Bunu belirlemek için birim sürede üretilen paket sayısı Şekil-3'te örneği verilen sözde program.

```
# uygulama performansı şimdilik sıfır
app_performance = 0
# PromQL dilinde performans metriği
m = container_network_transmit_packets_total
# uygulama içindeki bütün kapları tara
for each container in application
   # son 1 dakikalık veriyi kullan (PromQL)
   query = rate(q{pod=container}[1m])
   # Prometheus API'sine sorgu at
   perf = promv1api.Query(context, query)
   # uygulama performansını güncelle
   app_performance += perf
```

Şekil-3: *Kubernetes* ortamına kurulu olan *TeaStore* uygulamasının Promethus aracılığıyla performans ölçümünü yapan sözde-kod.

3.5 Girdi/Çıktı Verisinin Üretilmesi

Yük üretme, uygulama değişkenlerine müdahale etme, değişken uzayında rastgele gezinme ve performans ölçümü aşamaları bir araya getirildiğinde YBMG terimlerini elde etmek ve buna bağlı olarak evrensel duyarlılık analizlerini yapabilmek için gerekli veri hazırlanmış olmaktadır. Veri üretimi için gerekli bütün işlemler Şekil-4'teki sözde kod ile özetlenmiştir.

```
Uygulamayı başlat
Sabit yük göndermeye başla
Sayacı başlat: k = 1
   120 saniye bekle
   Mevcut parametreleri kaydet: x1(k), x2(k), x3(k)
   Performansı ölç ve kaydet: y(k)
   Bir parametreyi rastgele değiştir
   Sayacı arttır: k = k + 1
   Dördüncü adıma geri dön
```

Şekil-4: Veri toplama için uygulanan adımların özeti.

Veri üretimi, bir mikro servis uygulaması olan *TeaStore*'un bütün servisleri ile ayağa kaldırılması ile başlamaktadır. Ardından uygulamaya sabit yük gönderilmeye başlanır ve veri toplama süresince hiç kesilmez. Üçüncü adımda kayıtları numaralandırmak için kullanılan sayaç değişkeni tanımlanmaktadır. Dördüncü ve sonrasındaki adımlar *Kubernetes* operatörü içinde çalışmaktadır. Dördüncü aşamadaki 120 saniyelik bekleme süresi *Kubernetes* operatörlerinin içindeki *Reconcile* fonksiyonunun hangi aralıklarla devreye girdiği dikkate alınarak seçilmiştir. Uygulamanın yeni parametrelerle devreye girmesi ve performans ölçümünün güvenilir bir şekilde yapılabilmesi için bu sürenin yeterli olduğu görülmüştür. Beşinci ve altıncı adımlara gelindiğinde uygulamanın mevcut parametrelerle yeterince çalıştığı dikkate alınarak, parametre ve performans ölçümleri kaydedilir. Yedinci adımda parametrelerden en fazla birinde, ileri veya geri rastgele değişiklik yapılarak değişken uzayında gezinti yapılır. Yeterince örnek birikene kadar dört ve dokuz arası adımlar tekrar edilir. Bu çalışmada program yaklaşık olarak yirmi gün çalıştırılmış ve toplam 21.168 örnek alınmıştır. Çizelge-3'te bu şekilde toplanan verinin bir kısmı listelenmiştir.

Çizelge-3: Evrensel duyarlılık analizi amacıyla toplanan girdi/çıktı verisinin bir bölümü.

	Girdi			Çıktı
k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$y^{(k)}$
1	3	500	500	2640
2	4	500	500	2380
3	4	500	500	3029
4	5	500	500	2856
⋮				
21167	1	600	400	1268
21168	1	500	400	581

Dikkat edilmesi gereken önemli bir nokta aynı parametreler (örnek olarak $k = 2$ ve $k = 3$) altında çalışan programın performansının farklı ölçülebilmesidir. Bunun birkaç nedeni bulunmaktadır. Öncelikle her ne kadar uygulamaya sabit yük gönderilse de istekleri üreten ve gönderen sunucu üzerindeki yoğunluğa bağlı olarak gerçekte gönderilen istek sayısında salınımlar olmaktadır. Benzer durum istekleri karşılayan uygulama için de geçerlidir. Dahası çalışmanın yapıldığı veri merkezinde, uygulama ile ilgili bütün sunucu ve ağ bileşenleri performans ölçümlerinde az da olsa salınımlara neden olmaktadır. Salınımların yıkıcı etkisini azaltmak için veri toplama safhası özellikle uzun tutulmuş ve ızgaranın her düğüm noktası için birden fazla ölçüm yapılması sağlanmış ve ölçümlerin ortalaması alınmıştır. Bir başka ifade ile Çizelge-3'te sunulan veri girdilere göre gruplanmış ve her grup için çıktının ortalaması alınmıştır. Böylece bir sonraki bölümde bahsedilecek sayısal tümlev aşamalarında her düğüm noktasına karşılık sadece bir değerin bulunması garanti edilmiştir.

4. Analiz Sonuçları

4.1 YBMG Terimlerinin Hesaplanması

Öncelikle (3) numaralı denklemin sayısal yaklaştırımı ile sabit terim bulunur:

$$f_0 \approx \frac{1}{N}\sum_{s=1}^{N} f\left(x_1^{(s)}, x_2^{(s)}, x_3^{(s)}\right) = 2910{,}41 \text{ paket/sn}$$

Burada, N sayısal tümlev için kullanılan ızgaradaki toplam düğüm sayısını, s ise düğümlerin tekil sıra sayısını göstermektedir. Burada her düğümün tekil bir parametre kombinasyonuna karşılık geldiği tekrar hatırlanmalıdır. Düğüm noktasına karşılık gelen model çıktısı ise yukarıda anlatıldığı gibi Çizelge-3'teki ilgili kayıtların ortalaması hesaplanarak elde edilmiştir. Diğer bir deyişle,

$$B^{(s)} = \{y^{(s)} \mid (x_1^{(k)}, \ldots, x_3^{(k)}) = (x_1^{(s)}, \ldots, x_3^{(s)})\}$$

kümesi, s sıra sayılı düğüm noktasında yapılan bütün performans ölçümlerini ifade etmek üzere, model çıktısı bu kümenin ortalaması olarak tanımlanmıştır.

$$f(x_1^{(s)}, \ldots, x_3^{(s)}) \equiv \frac{1}{|B^{(s)}|}\sum_{l=1}^{|B^{(s)}|} B_l^{(s)}$$

Tek değişkenli (*birli terim* olarak da adlandırılmaktadır) YBMG terimleri ise, (4) numaralı eşitlikteki tümlevlerin sayısal yaklaştırımı ile bulunur.

$$f_1(x_1) \approx \frac{1}{N_2 \times N_3} \sum_{s=1}^{N_2 \times N_3} f(x_1, x_2^{(s)}, x_3^{(s)}) - f_0$$

$$f_2(x_2) \approx \frac{1}{N_1 \times N_3} \sum_{s=1}^{N_1 \times N_3} f(x_1^{(s)}, x_2, x_3^{(s)}) - f_0$$

$$f_3(x_3) \approx \frac{1}{N_1 \times N_2} \sum_{s=1}^{N_1 \times N_2} f(x_1^{(s)}, x_2^{(s)}, x_3) - f_0$$

Birli terimlerin herhangi bir noktadaki değerini bulmak için o noktanın modele girdi olarak verilmesi gerekmektedir. Verinin oluşturulması sırasında, model ızgara üzerinde çalıştırıldığı için bu çalışmada birli YBMG terimlerinin ızgara üzerindeki değerleri kolaylıkla bulunabilmektedir. Böylece birli terimleri Çizelge-4'teki gibi çizelge (İng. *look-up table*) şeklinde ifade etmek mümkündür.

Birli terimlerin grafik olarak gösterilimi ise Şekil-5'te yapılmıştır. Çizelge ve grafikte göze çarpan ilk özellik performans metriğinin her zaman pozitif ölçülmesine karşın YBMG terimlerinin negatif değer üretebilmesidir. Bunun nedeni bu terimlerin değişkenlerin performans üzerindeki bireysel katkıları ifade etmesidir. Örneğin $f_1(2) = -578$ değeri, kopya sayısının iki seçilmesi durumunda diğer parametrelere bakılmaksızın ortalama performansın 578 birim kadar düşeceği şeklinde yorumlanmalıdır. Benzer şekilde tek başına işlemci sınırının 700 seçilmesi ortalama performansı 1899 birim arttıracaktır. Şekil-5'e bu çerçevede bakıldığında kopya sayısının az olduğu bölgelerde beklendiği gibi negatif değerler, fazla olduğu yerlerde ise pozitif değerler görülmektedir.

Çizelge-4: Birli YBMG terimlerinin çizelge ile gösterilmesi.

x_1	$f_1(x_1)$	x_2	$f_2(x_2)$	x_3	$f_3(x_3)$
1	-1650	100	-2849	100	-302
2	-578	200	-2434	200	-36
3	-28	300	-1428	300	94
4	322	400	-528	400	10
5	441	500	685	500	84
6	282	600	1817	600	17
7	418	700	1899	700	-16
8	529	800	1559	800	19
9	254	900	1279	900	130

Performansa katkının büyüklüğü bağlamında bakıldığında iki numaralı değişkenin yani işlemci sınırının performans üzerinde çok etkili olduğu görülmektedir. 600 milicpu'ya kadar limit artışlarının performansı her seferinde arttırdığı görülmektedir. Ancak 700 milicpu ve sonrasında ek katkı gelmemektedir. Benzer davranış bir numaralı değişken olan kopya sayısı için de geçerlidir. Dörtten fazla kopya kullanmanın ek performans getirmediği görülmektedir. Üçüncü ve son değişken olan yığın sınırına karşılık gelen YBMG teriminin sıfır civarında küçük salınımlar yaptığı dolayısı ile diğer iki değişkene göre neredeyse ihmal edilebilir olduğu görülmektedir.

4.2 Duyarlılık Katsayılarının Hesaplanması

Birli YBMG terimleri, değişkenlerin performans üzerindeki bireysel katkılarını görsel olarak göstermekle birlikte, katkıları nicel olarak tek bir sayıya indirgemek için evrensel duyarlılık katsayılarının hesaplanması gereklidir. Bunun için Bölüm 2.2'de gösterilen varyanslar aşağıdaki gibi sayısal tümlev ile yaklaşık olarak bulunmuş ve elde edilen duyarlılık katsayıları Çizelge-5'de sunulmuştur.

$$D \approx \frac{1}{N} \sum_{s=1}^{N} f^2\left(x_1^{(s)}, x_2^{(s)}, x_3^{(s)}\right) - f_0^2$$

$$\mathrm{D_i} \approx \frac{1}{N} \sum_{s=1}^{N} f_i^2\left(x_i^{(s)}\right), \quad S_i = \frac{D_i}{D}, \qquad i = 1,2,3$$

Çizelge-5: Performans dikkate alınarak hesaplanan duyarlılık katsayıları.

Duyarlılık Katsayısı	Değer
S_1	0,1041
S_2	0,7303
S_3	0,0033

Çizelge-5'e göre ikinci değişkenin yani işlemci sınırının en etkili parametre olduğu anlaşılmaktadır. Diğer bir ifade ile model üzerinde etkili olan bütün faktörlerin toplam 100 birimlik etkisinin yaklaşık 73'ü tek başına ikinci değişken tarafından sağlanmaktadır. İkinci sırada ise yaklaşık %10 ile kopya sayısı gelmektedir. Yığın sınırı ise binde üç gibi küçük bir paya sahiptir. Elde edilen duyarlılık katsayıları birli YBMG terimlerinin grafiklerine bakılarak yapılan yorumlarla paralellik göstermektedir. Ayrıca kopya sayısı ve işlemci sınırının katkılarının toplamının %83'e ulaşması, modelin %17'lik bir hata payı ile bu değişkenlere karşılık gelen birli YBMG terimlerinin toplamı şeklinde ifade edilebileceğini göstermektedir.

4.3 Analizlerin Maliyet için Tekrarlanması

Bu bölüme kadar yapılan bütün ölçme ve hesaplamalarda model çıktısı olarak *TeaStore* uygulamasının performans ölçümü kullanılmıştır. Ancak giriş bölümünde bahsedildiği üzere kullanılan kaynakların maliyeti de bulut uygulamalarının yönetimi açısından en az performans kadar önemlidir. Bu doğrultuda bir önceki bölümde yapılan hesaplamalar performans yerine maliyet dikkate alınarak tekrar edilmiştir. Bulut uygulamasının maliyeti ise işlemci ve bellek kullanımının bulut piyasasında geçerli ortalama saatlik ücretler[1] ile çarpılmasıyla belirlenmiştir.

Çıktı olarak maliyeti temel alarak hesaplanan evrensel duyarlılık katsayıları Çizelge-6'da, birli terimlerin grafikleri ise

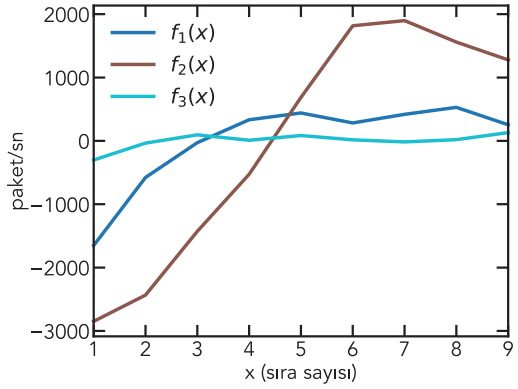
[1] https://github.com/kubecost/cost-model

Şekil-6'da gösterilmiştir. Performansa göre yapılan analize benzer şekilde kopya sayısı ve işlemci sınırının maliyet üzerinde belirgin bir etkisi gözlenirken, yığın sınırının hiçbir etkisinin olmadığı anlaşılmaktadır. Bir önceki durumdan farklı olarak kopya sayısı ile model çıktısı arasında neredeyse doğrusal bir ilişki gözlenmektedir. İşlemci sınırının etkisi ise belli bir noktadan sonra sabit kalmaktadır.

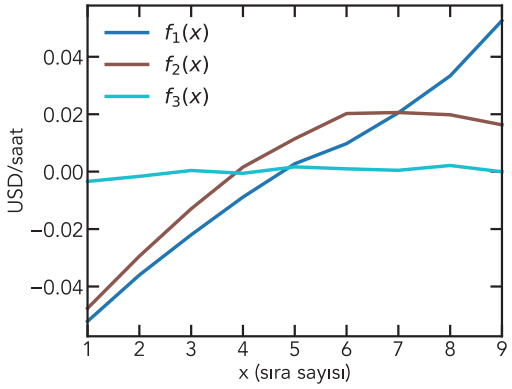
Kopya sayısının ve işlemci sınırının maliyet üzerindeki etkisi ise sırasıyla %60 ve %33 olarak hesaplanmıştır. Yığın sınırının maliyet üzerindeki binde ikilik katkısı ihmal edilebilecek kadar küçüktür. İlk iki değişkenin toplam etkisinin %93 olması, maliyetin sadece iki değişkenli toplamsal bir model ile ifade edilebileceği anlamına gelmektedir.

Çizelge-6: Maliyet dikkate alınarak hesaplanan duyarlılık katsayıları.

Duyarlılık Katsayısı	Değer
S_1	0,5977
S_2	0,3279
S_3	0,0016



Şekil-5: Performansa göre hesaplanan birli YBMG terimleri



Şekil-6: Maliyet dikkate alınarak hesaplanan birli YBMG terimleri

5. Sonuç ve Tartışma

Bu çalışmada bulut bilişim uygulamalarının yönetiminde kullanılan ölçekleme ve ince ayar kavramları genel hatları ile tanıtılmış ve bunlarla ilinti olan parametrelerin performans ve maliyet üzerindeki etkileri bulut bilişim alanında ilk defa kullanılan Yüksek Boyutlu Model Gösterilimi (YBMG) adlı evrensel duyarlılık analizi yöntemi ile analiz edilmiştir

Bütün analizler *Kubernetes* platformu ve onun üzerine kurulan *TeaStore* adlı örnek mikro servis uygulaması aracılığıyla yapılmıştır. Çalışmanın parametre yönetimi ve veri toplama safhaları için ise *Operator-SDK* platformuna dayanan *Golang* tabanlı özel bir *Kubernetes* operatörü hazırlanmıştır

Duyarlılık analizi sonuçlarına göre dikey ölçekleme kategorisinde değerlendirilen işlemci sınırı parametresinin performansın neredeyse dörtte üçünden, maliyetin ise üçte birinden tek başına sorumlu olduğu görülmektedir. Yatay ölçeklemeyi temsil eden kopya sayısı parametresi ise performansın yüzde on birinden, maliyetin ise yüzde altmışından sorumludur. İnce ayar ile ilgili olan yığın sınırı parametresinin ise performans ve maliyete anlamlı bir katkısı bulunamamış, dolayısı ile ölçekleme çalışmalarında, en azından TeaStore uygulaması bağlamında, göz ardı edilebileceği anlaşılmıştır. Ayrıca *TeaStore* uygulaması için, performans ve maliyetin küçük bir hata payı ile kopya sayısı ve işlemci sınırı parametrelerine göre tanımlanmış tek değişkenli fonksiyonların doğrusal toplamı olarak ifade edilebileceği anlaşılmıştır.

Bu çalışmada önerilen duyarlılık analiz yaklaşımının diğer bulut uygulamalarına küçük değişikliklerle uyarlanabileceği açıktır. Dahası bulut uygulamalarının yönetimi için kullanılan genel maksatlı ölçekleme araçlarının parametre seçim mekanizmalarına entegre edilerek, ölçekleme süreçlerinin iyileştirilmesi sağlanabilir. Analiz sonuçlarının diğer bir kullanım alanı ise uygulama dinamiklerinin daha iyi anlaşılması ve bu sayede iş geliştirme süreçlerinin verimliliğinin arttırılmasıdır.

Çalışmanın bir sonraki safhasında, değişken sayısının ve olası değerlerin arttırılarak uygulama ölçeğinin büyütülmesi planlanmaktadır. Ayrıca iki değişkenli YBMG terimlerinin ve ikili duyarlılık katsayılarının da dikkate alınarak parametre etkileşimlerinin daha hassas modellenmesi planlanmaktadır.

6. Kod ve Veri Paylaşımı

Çalışma sonuçlarının üretilmesi için kullanılan yazılım ve elde edilen veri https://github.com/app2scale/gsa adresinde açık erişime sunulmuştur.

7. Teşekkür

Bu çalışma İTÜ ARI Teknokent bünyesinde yürütülen 62719 numaralı Ar-Ge projesi kapsamında yapılmıştır.

Kaynakça

[1] Waldrop, M. M. *Data center in a box*, Scientific American, 2007, 297(2), pp. 90-93.

[2] Dutreilh, X., Moreau, A., Malenfant, J., Rivierre, N. ve Truck, I. J. *From data center resource allocation to control theory and back*, IEEE 3rd international conference on cloud computing, 2010, pp. 410-417.

[3] Lorido-Botran, T., Miguel-Alonso, J. ve Lozano, J.A. *A review of auto-scaling techniques for elastic applications in cloud environments*, Journal of grid computing, 2014, 12(4), pp. 559-592.

[4] Baresi, L., Hu, D.Y.X., Quattrocchi, G. and Terracciano, L., *KOSMOS: Vertical and Horizontal Resource Autoscaling for*

Kubernetes, International Conference on Service-Oriented Computing, 2021, pp. 821-829s

[5] Qu, C., Calheiros, R.N. ve Buyya, R. *Auto-scaling web applications in clouds: A taxonomy and survey*, ACM Computing Surveys, 2018, 51(4), pp.1-33.

[6] Incerto, E., Tribastone, M. ve Trubiani, C. *Combined vertical and horizontal autoscaling through model predictive control,* European Conference on Parallel Processing, 2018, pp. 147-159.

[7] Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L. ve Meng, X. *A reinforcement learning based auto-scaling approach for SaaS providers in dynamic cloud environment*, Mathematical Problems in Engineering, 2019.

[8] Whaley, R.C., Petitet, A. ve Dongarra, J.J. *Automated empirical optimizations of software and the ATLAS project*, Parallel computing, 2001, 27(1-2), pp.3-35.

[9] Di Sanzo, P., Rughetti, D., Ciciani, B. ve Quaglia, F. *Auto-tuning of cloud-based in-memory transactional data grids via machine learning*, Second Symposium on Network Cloud Computing and Applications, 2012, pp. 9-16

[10] Nguyen, T.T., Yeom, Y.J., Kim, T., Park, D.H. ve Kim, S. *Horizontal pod autoscaling in kubernetes for elastic container orchestration*, Sensors, 2020, *20*(16), p. 4621.

[11] Rabitz, H. ve Alış, Ö.F. *General foundations of high-dimensional model representations*, Journal of Mathematical Chemistry, 1999, 25(2), pp.197-233.

[12] Sobol', I. M. *On sensitivity estimation for nonlinear mathematical models.* Matematicheskoe modelirovanie, 1990, 2(1), pp. 112-118.

[13] Sobol', I.M. *Theorems and examples on high dimensional model representation*, Reliability Engineering and System Safety, 2003, 79(2), pp.187-193.

[14] Li, G., Rosenthal, C. ve Rabitz, H. *High dimensional model representations*, The Journal of Physical Chemistry A, 2001, 105(33), pp.7765-7777.

[15] Li, G., Wang, S.W. ve Rabitz, H. *Practical approaches to construct RS-HDMR component functions,* The Journal of Physical Chemistry A, 2002, 106(37), pp.8721-8733.

[16] Li, G., Wang, S.W., Rosenthal, C. and Rabitz, H. *High dimensional model representations generated from low dimensional data samples. I. mp-Cut-HDMR.* Journal of Mathematical Chemistry, 2001,30(1), pp.1-30.

[17] von Kistowski, J., Eismann, S., Schmitt, N., Bauer, A., Grohmann, J. ve Kounev, S. *Teastore: A micro-service reference application for benchmarking, modeling and resource management research*, IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2018, pp. 223-236.

[18] Burns, B., Grant, B., Oppenheimer, D., Brewer, E. ve Wilkes, J. *Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade*, Queue, 2016, *14*(1), pp. 70-93.

[19] Dobies, J. ve Wood, J. *Kubernetes Operators: Automating the Container Orchestration Platform*, O'Reilly Media, 2020.

[20] Pike, R. *The go programming language*. Talk given at Google's Tech Talks, 14, 2009.

[21] von Kistowski, J., Deffner, M. ve Kounev, S. *Run-time prediction of power consumption for component deployments*, IEEE International Conference on Autonomic Computing, 2018, pp. 151-156.

[22] Turnbull, J. *Monitoring with Prometheus*, Turnbull Press, 2018.

[23] Sabharwal, N. ve Pandey, P. *Working with Prometheus Query Language (PromQL). In Monitoring Microservices and Containerized Applications*, Apress, Berkeley, 2020, pp. 141-167.