Bulut-tabanlı Ortamda Yürütülen Bir Yazılım Performans Testi Deneyimi

Ali Yamuç, Ethem Cem Özkan, Burak İbrahim Sevindi,
Hakan Şimşek, Turan Bahattin Özen

TÜBİTAK BİLGEM YTE, Çukurambar Mah. 1478. Cad. No: 22 Çankaya ANKARA
{ali.yamuc, cem.ozkan, burak.sevindi, hakan.simsek,
turan.ozen}@tubitak.gov.tr

Özet. Bu bildiride, TÜBİTAK BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü (YTE) tarafınca geliştirilmeye devam edilen yüksek performans, erişilebilirlik ve ölçeklenebilirlik gereksinimleri olan bir takip ve izleme projesi için bulut-tabanlı ortamda yürütülen yazılım performans test deneyimleri paylaşılmaktadır. Bu kapsamda böyle yüksek performans, erişilebilirlik ve ölçeklenebilirlik gereksinimleri olan yazılım sistemleri için performans testinin gerekliliği ve önemi üzerinde durulmaktadır. Daha sonra performans testi yürüttüğümüz projenin gereksinimlerine değinilerek; problemin zorluğu vurgulanmaktadır. Son olarak; performans test ortamının Amazon Web Services (AWS) üzerinde hazır edilmesi, performans testlerinin koşturulması, ölçevlerin (metrics) toplanması, analiz ve ayarlama (tuning) faaliyetlerinde bu projede edinilen deneyimler, sağlanan kazançlar paylaşılmakta ve ileriye yönelik hedeflere değinilmektedir.

Anahtar Kelimeler: Performans Testi, Yüksek Performanslı Yazılım Sistemleri, Bulut-tabanlı Ortamda Performans Testi, Yazılım Performans Mühendisliği

1 Giriş

Büyük ölçekli yazılım sistemlerinin artışı yazılım performans mühendisliği için yeni zorluklar ortaya çıkarmıştır [1]. Dağıtık bir şekilde pek çok makineye konuşlandırılan bu yazılım sistemleri mükemmele yakın erişebilirlik süresini ve binlerce eşzamanlı bağlantıyı ve işlemi sağlamayı gerektirmektedirler. Literatürdeki bazı çalışmalar göstermiştir ki [2, 3, 4], bu tarz yazılım projelerinin en sık başarısız olma nedeni işlevsellik sorunlarından ziyade performans sorunlarından kaynaklanmaktadır. Performans sorunlarının çok ciddi sonuçlarının görüldüğü popüler olaylardan bazıları şunlardır: ABD hükümetinin healthcare.gov hizmetini halka açması [5], Apple MobileMe'nin lansmanı [6], Firefox 3.0'ın piyasa sürülmesi [7] esnasında ölçeklenememesi; NASDAQ'ın Facebook hisselerinin ilk halka arzı esnasında yaşanılan oldukça maliyetli eş-zamanlılık hataları [8]; AWS'de bulunan oldukça maliyetli bellek sızıntıları [9]. Bu tarz yazılım performans arızalarının finansal ve saygınlık açısından ciddi sonuçlarının olduğu çeşitli çalışmalarda raporlanmıştır [10,11]. Yazılım performans testi, özellikle büyük ölçekli yazılım sistemlerinin performans gereksinimlerini karşıladığından emin olmamızı sağlayan temel yöntemlerden biridir [12]. Performans gereksinimleri; çıkan iş oranı (throughput), cevap süresi, kaynak kullanım verimliliği

gibi performans ölçevleriyle tanımlanabilir. Çıkan iş oranı birim zamanda tamamlanan işlemlerin sayısını, cevap süresi istemciden bir isteğin alınmasından istemciye cevap dönülmesine kadar geçen süreyi, kaynak kullanım verimliliği (CPU, bellek, disk I/O, ağ I/O vb.) ise kullanılan sistem kaynaklarının miktarını ve verimliliğini tanımlamaktadır.

Performans gereksinimlerinin doğrulaması ölçüme dayalı ve deneysel bir yaklaşım olan performans testi yürütülmesinin yanı sıra analitik performans modelleriyle veya bunların karma kullanımlarıyla da gerçekleştirilebilir. Analitik performans modelleriyle ile mimari tasarımı daha gerçeklemeden, performans hedeflerinin sağlanıp sağlanamayacağının değerlendirilmesi yapılabilir ve farklı ödünleşim (trade-off) analizlerinde bulunarak olası yanlış mimari tasarım kararlarından kaynaklanabilecek büyük maliyetlerden kurtulmak mümkün olabilir. Yazılım performans testinde yazılım sistemi belli bir zaman dilimi boyunca belli bir iş yüküne tabi tutularak çıkan iş oranı, cevap süresi, kaynak kullanım verimliliği gibi performans ölçevleri toplanır, sonuçlar analiz edilir. Daha maliyetli bir yöntem olmasına rağmen ürün-ortamına yakın bir performans test ortamının kullanılmasıyla gerçek performans ölçevlerinin sonuçları alınır ve ürün ortamında bir sürpriz ile karşılanması önlenmiş olur. Performans modellerini sağladığı esneklik ve maliyet faydasına rağmen içerebileceği yanlış varsayımlar gerçekten uzaklaştırabilme olasılığına sahiptir. Bundan dolayı tüm bu teknik ve yaklaşımları bir mühendislik disiplini içerisinde birbirlerini destekleyecek şekilde kullanmak son yıllarda öne çıkan önemli bir konudur ve yeni bir mühendislik disiplini olarak Smith [13] tarafından yazılım performans mühendisliği olarak ortaya atılmıştır. Yazılım performans mühendisliği; performans problemlerini gidermek, yazılım sistemlerinin performansını değerlendirmek ve iyileştirmek için yazılım geliştirme süreci boyunca kullanılan sistematik, niceliksel ve analitik yaklaşımlardan oluşan yazılım mühendisliği aktiviteleri içermektedir.

Yazılım performans mühendisi, performans kaygısı yüksek olan sistem operasyonlarını (kullanım durumları) tüm sistem operasyonlarından indirgeyebilir. Seçilen bu operasyonların göreceli olarak ağırlıklandırılması, operasyonel profili [14] oluşturur. Operasyonel profile dayanarak performans test senaryoları çeşitli iş yükü modelleriyle oluşturulur. Performans testi, iş yükü modelleri kullanılarak sistemin tamamı veya bir kısmı üzerinde gerçekleştirilir. Farklı iş yükü modellerine göre farklı performans test türleri oluşmaktadır. Bunların her biri farklı türdeki performans sorunlarını giderebilir. Ortaya çıkarabildiği performans sorunlarıyla birlikte bu performans türlerinin bazıları şunlardır:

- **Yük Testi** – Spesifik ve beklenen bir yük altında sistem davranışının belirlenmesini sağlar.
- **Stres Testi -** Uzun bir süre yükün dramatik bir şekilde artarak sistemin kırılma noktalarının belirlenmesini sağlar.
- **Ani Yük (Spike) Testi –** Yükteki dramatik ani değişimlere karşı sistemin nasıl davrandığının belirlenmesini sağlar.
- **Dayanıklılık (Endurance/Soak) Testi -** Beklenen bir yükün sürekli uygulanarak zaman içerisinde ortaya çıkan kaynak sızıntılarının belirlenmesini sağlar.

- **Ölçeklenebilirlik Testi -** Artan iş yüküne göre bağlı olarak sistemin daha fazla kaynağı doğrusal olarak artan bir şekilde kullanabildiğinin belirlenmesini sağlar.

Performans test türleri farklı ihtiyaçlara binaen ve keşfedilmesi istenen olası farklı performans sorunları için daha fazla çeşitlendirilebilir ve bundan ötürü literatürde daha pek çok performans test türü raporlanmıştır [15]. Performans testini daha olgun bir şekilde yürütebilmek için mümkün olduğunca farklı performans test türlerinden faydalanmak gerekmektedir. Çünkü her biri performans gereksiniminin doğrulanması, en zayıf eleman (darboğaz) tespiti, kaynak sızıntısı, ölçeklenebilirlik problemi gibi farklı sorunlara, farklı perspektiflerden dokunmaktadır.

Bildiri şu planlamadan oluşmaktadır. Bölüm 2'de geliştirilen takip ve izleme projesinin performans gereksinimleri anlatılacaktır. Bölüm 3'de uyguladığımız performans test süreci anlatılacaktır. Bölüm 4'de hazırladığımız bulut-tabanlı performans test ortamı tanıtılacaktır. Bölüm 5'de performans testlerini koşarken, ölçevleri toplarken ve analiz faaliyetlerinde edinilen deneyimler paylaşılacaktır.

2 Performans Gereksinimleri

TÜBİTAK BİLGEM YTE tarafınca geliştirilmesi devam eden, vatandaş odaklı hizmetler sunan ve yüzbinlerce kullanıcılı bir takip ve izleme sistemi projesinde performans testlerini gerçekleştirdik. Bu projede özellikle takip ve izleme ile alakalı bir modül, makine-makine etkileşiminin çok yüksek boyutlarda olmasından dolayı oldukça yüksek performans gereksinimlerine sahiptir. Sistem çeşitli firmaların üretim bantlarıyla entegre olacağından ve üretimin 7/24 devam edeceği varsayıldığından, sistemin yüksek erişilebilirlikte olması gerekmektedir. Geliştirilmesi devam eden bu yazılım sistemi, canlıya alındığında ilk yıllarda 19000 işlem/saniye iş çıkarma yeteneğine sahip olması gerekirken, ileriki yıllarda en az 30000 işlem/saniye oranlarına hızlıca çıkacağı öngörülmektedir. Bu yüzden yazılım sisteminin performans gereksinimlerini karşılamasının yanı sıra, kesintisiz olarak kullanımda kaldığı zaman boyunca ölçeklenebilmelidir. Ayrıca, bu modüldeki işlemlerin müşteri memnuniyeti açısından cevap süresi maksimum 1 saniye olmalıdır. Tüm bu gereksinimler daha dar kapsamlı olan, mevcut kullanımdaki takip ve izleme sistemlerinin değerleri göz önüne alınarak ve uzman görüşlerine dayanılarak hesaplanmıştır.

Bu performans, ölçeklenebilirlik ve erişilebilirlik gereksinimleri bir kurumsal bilgi sistemi için oldukça eşine az rastlanabilecek, oldukça zorlu bir problemi ortaya koymaktadır. Bu problemin üstesinden gelebilmek için yazılım geliştirme sürecinin erken safhalarında başlayarak performans testleri tanımlanmış süreçlere uygun bir şekilde yürütülmektedir.

3 Performans Test Süreci

Performans test sürecinde öncelikle performans gereksinimleri tanımlanır. Daha sonra, performans testlerini yürütmek için hedeflenen ürün ortamına yakın bir test ortamı tanımlanır ve hazırlanır. Uygulamanın operasyonel profiline göre gerçekçi ve

kilit senaryolar tanımlanır. Test verisi olarak fonksiyonel testlerde kullanılan veriler kullanılır fakat dinamik olarak çokça üretilmesi gereken test verileri için betiklerin hazırlanması gerekir. Dış servis bağımlılıklarını kırmak için sanal servisler oluşturulur. Performans test tasarımı seçilen bir performans test aracıyla gerçeklenir. Oluşturulan performans test betikleri yerel geliştirme ortamında düşük bir yükte çalıştırılarak doğrulandıktan sonra, gerçek test ortamına aktarılır. Testin koşturulması esnasında çevrimiçi olarak sistem kaynakları (CPU, Network I/O, Disk I/O), sunucu günlükleri (log), cevap süresi ve çıkan iş oranı izlenerek, olası performans sorunları anlık tespit edilmeye çalışılır. Test bitiminden sonra tüm ölçevler toplanarak kaydedilir ve sonuçlar konsolide edilerek analiz edilir. Analiz esnasında istenilen tüm ölçevlerin kabul edilen limitler içerisinde olduğuna ve hiçbir eşik değerinin aşılmadığına ve istenen tüm bilgilerin toplandığına emin olunduğunda, ilgili performans testi bu konfigürasyonda tamamlanmış olur. Eğer bir sorun gözlemlenirse darboğaz unsurlar tespit edilmeye çalışılır. Eldeki bulgulara göre gereken hata raporları açılır ve iyileştirici aksiyonlar planlanır. Yapılan ayarlama faaliyetlerinden sonra iyileşme olup olmadığını anlamak için yeniden test yapılır. Buna göre gerçekleşen ilerlemeleri kayıt altına alarak dayanaklar oluşturulur. Bu projede uygulanan bu test süreci, edinilen deneyimlere göre sürekli olarak iyileştirilmeye devam etmektedir.

4 AWS üzerindeki Performans Test Ortamı

Takip ve izleme projesi sahip olduğu yüksek performans gereksinimlerinden dolayı performans testlerini yürütmek için yüksek sunucu kapasitesine ihtiyaç duyulmuştur. Böyle yüksek kapasiteli ortamı geleneksel bir ortam yerine bulut ortamlarda hazırlamak; farklı teknolojilerin, konfigürasyonların ve yazılım tasarımlarının denenebilmesi esnekliğini maliyet etkin bir şekilde sağladığı görülmüştür. Bu konuda bazı çalışmalar [16, 17]; AWS'nin sağladığı hizmet kalitesinin ve maliyetinin alternatif bulut hizmetlerine ve kurum içi sahipliğe göre daha iyi olduğunu niceliksel olarak göstermiştir. Ayrıca, AWS üzerinde kurum içi bilgi birikimine sahip olunması ve ortamın betiklerle tanımlanarak kolay yönetilebiliyor olmasından dolayı AWS tercih edilmiştir.

Performans test ortamı 2 ayrı ortamdan oluşmaktadır. İlki takip ve izleme sistemini, diğeri ise yükleri oluşturan JMeter ortamını içerir. Performans test aracı olarak JMeter v2.13; açık-kaynak ve ücretsiz olması, kullanım kolaylığı, her ortamda çalışabilir olması, güçlü bir topluluğu ve dokümantasyonu olmasından dolayı tercih edilmiştir. Bulut-tabanlı ortam üzerinde performans testi yapılırken sunucuların oluşturulması, gerekli ağ ayarlarının yapılması, sunucular üzerine ilgili programları kurulması, testlerin çalıştırılması ve en son bu sunucuların silinmesi işlemlerinin yapılması gerekmektedir. Bu işlemlerin her performans testi öncesi el ile yapılması maliyetli ve hataya açık bir işlem olacağı için performans test ortamı AWS Cloudformation aracı kullanılarak, betikler ile otomatik olarak oluşturulmaktadır. JMeter ortamı, GitHub (https://github.com/oliverlloyd/jmeter-ec2) üzerinde bulunan JMeter-EC2 adlı açık kaynak kodlu proje kullanılarak AWS üzerinde oluşturulmaktadır.

5 Edinilen Deneyimler

Performans testlerini yürütmek için öncelikle geliştirilmekte olan sistemin mevcut dış sistem entegrasyonlarını sanallaştırmamız gerekti. Sanal servisler, geliştirilmekte olan sistemin performans testleri sırasında dış sistemlere olan bağımlılıklarını kırmak, dolayısıyla sistemin gerçek performans seviyelerini elde edebilmek için gerekmektedir. Buradan yola çıkarak, sanal servisleri oluşturmak için Kurumsal Hizmet Veri Yolundan istifade edilmiştir. Uygulamanın operasyonel profili, alan uzmanlarına danışılarak belirlenmiş ve senaryolar oluşturulmuştur.

Performans testleri AWS ortamında koşarken, sadece AWS'in sunduğu ölçevleri kullanabildik. Bu ölçevlerden de sadece CPU ölçümünün sağlıklı olarak alınabildiği ve diğer sistem kaynaklarıyla ilgili ölçevlerin toplanmasında AWS'den kaynaklı sorun görülmüştür. İlgili sunucuların günlük analizleri her bir sunucunun ilgili dosyaları elle kontrol edilerek yapılmış, otomatik ve konsolide bir günlük yönetimi ihtiyacı görülmüştür. Çıkan iş oranı ve cevap süreleri ancak test bitiminde JMeter'ın sonuç dosyasından alınabildiği için anlık olarak analizin çok sağlıklı bir şekilde yapılamadığı tespit edilmiştir.

Performans testlerinde ileride ortaya çıkması durumunda düzeltme maliyetinin oldukça yüksek olacağı, performansla ilgili hatalar raporlanarak çözümler aranmıştır. Raporladığımız hatalar kabaca şöyle sınıflandırılabilinir.

- Eş-zamanlı programlama hataları. Optimistik kilit arızaları (optimistic lock failure), yarışma durumuna (race condition) bağlı veri tutarlılığı problemleri.
- Veritabanı sorgularındaki optimizasyon problemleri (Lazy loading yerine eager loading yapılması gibi).
- Bazı iş kuralları uygulamanın istemci tarafında işletilirken, sunucu tarafında yapılmaması ile ilgili hatalar.
- HTTP/JSON servis arayüzlerinin eksik, fazla ve hatalı parametreleri ve dönen cevap mesajlarının anlaşılmayan hata mesajları içermesi gibi hatalar.

6 Sonuç

Bu bildiride, TÜBİTAK BİLGEM YTE tarafınca geliştirilmeye devam edilen takip ve izleme sistemi projesi için gerçekleştirilen performans testlerinin niçin yapılması gerektiğini, nasıl bir süreçle yürütüldüğünü, nasıl bir performans test ortamında gerçekleştirildiğini ve performans testleri esnasında edinilen deneyimleri paylaştık. İleride düzeltilmesi zor olan pek çok performansla ilgili hata daha geliştirme safhasında müdahale edilerek çözülmüş ve ciddi bir maliyetten böylelikle sakınılmıştır. Mevcut altyapıdaki izleme, ölçev toplama, günlük yönetimi, daha kaliteli çevrimiçi ve çevrimdışı analiz ihtiyacı gibi alanlarda iyileştirme yapabildiğimiz takdirde geliştirilen yazılım sisteminin performans, ölçeklenebilirlik ve erişilebilirlik hedeflerine daha kolay bir şekilde ulaşabileceği ve ürün müşteriye teslim edilmeden istenilen fonksiyonel olmayan kalite hedeflerine zamanında ve bütçesinde erişilebileceği görülmüştür. Bundan dolayı, daha nitelikli bir şekilde performans testlerini yürütmek için uygulama performans yönetimi araçlarının geliştirilmesi veya hazır araç kullanılması

zaruret olarak görülmüştür. İlerleyen günlerde daha nitelikli bir şekilde performans testleri, analizi ve sorun çözümlemesi yapabilecek noktada olmayı ve analitik performans modellerinden de istifade edildiğinde, daha verimli ve etkin bir şekilde yazılım performans yönetimi yapabilmeyi hedefliyoruz.

Kaynaklar

1. Software Engineering Institute, "Ultra-Large-Scale Systems: The Software Challenge of the Future", Carnegie Mellon University, Pittsburgh 2006
2. Bass L., Nord R., Wood W., Zubrow D., "Risk Themes Discovered Through Architecture Evaluations", IEEE/IFIP Conference on Software Architecture (WICSA'07), 6-9 Jan. 2007
3. Weyuker, E., Vokolos, F., "Experience with performance testing of software systems: issues, an approach, and case study" Trans. Softw. Eng. cilt 26 sayı 12 s.1147–1156, 2000
4. Syer M. D., Shang W., Jiang Z.M., Hassan A. E., "Continuous validation of performance test workloads", Automated Software Engineering, s. 1-43, 2016
5. HealthCare.gov, https://en.wikipedia.org/wiki/HealthCare.gov/ Erişim Tarihi: 16/05/2016
6. Cheng, J., "Steve jobs on MobileMe". http://arstechnica.com/apple/2008/08/steve-jobs-on-mobileme-the-full-e-mail/ Haber Tarihi: 2008, Erişim Tarihi: 16/05/2016
7. SiliconBeat: Firefox download stunt sets record for quickest meltdown. http://www.siliconbeat.com/2008/06/17/firefox-download-stunt-sets-record-for-quickest-meltdown/ Haber Tarihi: 2008, Erişim Tarihi: 16/05/2016
8. Benoit, D., "Nasdaqs blow-by-blow on what happened to Facebook" http://blogs.wsj.com/deals/2012/05/21/nasdaqs-blow-by-blow-on-what-happened-to-facebook/ Haber Tarihi: 2012 Erişim Tarihi: 16/05/2016
9. Williams, A., "Amazon web services outage caused by memory leak and failure in monitoring alarm" http://techcrunch.com/2012/10/27/amazon-web-services-outage-caused-by-memory-leak-and-failure-in-monitoring-alarm/ Haber Tarihi: 2014, Erişim Tarihi: 16/05/2016
10. Howell Jr., T., Dinan, S., "Price of fixing, upgrading obamacare website rises to $121 million", http://www.washingtontimes.com/news/2014/apr/29/obamacare-website-fix-will-cost-feds-121-million/ Haber Tarihi: 2014, Erişim Tarihi: 16/05/2016
11. Harris, C., "IT downtime costs $26.5 billion in lost revenue" http://www.informationweek.com/it-downtime-costs-$265-billion-in-lost-revenue/d/d-id/1097919 Haber Tarihi: 2011, Erişim Tarihi: 16/05/2016
12. Bondi A. B., "Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice", Addison-Wesley Professional, 2014
13. Smith C., "Software performance engineering", Performance Evaluation of Computer and Communication Systems, Lecture Notes in Computer Science cilt.729 s. 509-536, 2005
14. Musa J.D., "Operational Profiles in Software-Reliability Engineering", IEEE Software, cilt.10 sayı 2 s.14-32, 1993
15. Black R., Mitchell J., "Advanced Software Testing Vol. 3", Rocknook Computing, 2011
16. Marathe A., Harris R., Lowenthal D. K., Supinski B. R., Rountree B., Schulz M., Yuan X., "A Comparative Study of High-Performance Computing on the Cloud", Proceedings of the 22nd Int. Symp. on High-Performance Parallel and Distributed Comp., s.239-250, 2013
17. Zhai Y., Liu M., Zhai J., Ma X., Chen W., "Cloud Versus In-house Cluster: Evaluating Amazon Cluster Compute Instances for Running MPI Applications", Proceeding SC '11 State of the Practice Reports, no 11, 2011