Performans Testi ve CPTS Öğrenme Notları (Learning Performance Testing & CPTS Notes)

1. Giriş

Bugünkü konularım **performans testleri türleri (Performance Testing Types)** ve **CPTS (Cloud Performance Test Service)** oldu.

Performans testleri, bir yazılımın yalnızca fonksiyonel olarak doğru çalışmasını değil, aynı zamanda **yük (load)** ve **trafik artışı (traffic spikes)** altında nasıl davrandığını görmek için kritik öneme sahiptir. Gerçek dünyada kullanıcı sayısı bir anda arttığında sistemin çökmesi (crash), yanıtların yavaşlaması (latency) ya da hata üretmesi (error rate) hem müşteri kaybına hem de finansal zarara yol açabilir.

CPTS gibi araçlar bu testleri **bulut (cloud)** ortamında yapmayı sağlar. Lokal bilgisayarın sınırlarını aşarak binlerce kullanıcıyı aynı anda simüle edip, sistemin **nerede güçlü, nerede zayıf** olduğunu görebilirim.

Projelerdeki kullanım alanları:

- API performansını ölçmek
- Sistemin dayanıklılığını test etmek
- Kullanıcı yüküne hazırlık yapmak (kampanyalar, canlı yayınlar, finansal sistemlerde işlem yoğunluğu)
- Canlıya çıkmadan önce riskleri minimize etmek
- Olası bottleneck noktalarını önceden keşfetmek

---

2. Performance Testing (Performans Testleri)

2.1 Load Testing (Yük Testi)

Tanım: Sistemin belirli sayıda eşzamanlı kullanıcı (concurrent users) veya işlem altındaki davranışını ölçmek.
Hedef: Sistemin **normal şartlarda kaldırabileceği maksimum yükü** öğrenmek.

Örnek Senaryolar:

- Bir e-ticaret sitesinde kampanya zamanı aynı anda **5000 kullanıcının sepete ürün eklemesi**.
- Üniversite kayıt sisteminde yeni dönem başladığında **binlerce öğrencinin aynı anda ders seçmesi**.

Load testleri ile sistemin yanıt süresi (response time) 1 saniye içinde mi kalıyor, yoksa yük arttıkça 5–10 saniyeye mi çıkıyor sorusuna cevap buluruz.

2.2 Stress Testing (Stres Testi)

Tanım: Sisteme kapasitesinin çok üzerinde yük bindirilerek **çökme noktalarının** belirlenmesi.
Hedef: Sistemin hangi noktada hata verdiğini ve hata öncesinde nasıl davrandığını anlamak.

Örnek Senaryolar:

- Bankacılık uygulamasında normalde 10.000 işlem varken, testte **100.000 işlem gönderildiğinde** sistem çöker mi?
- Oyun sunucusunda normalde 20.000 oyuncu varken, bir anda 200.000 oyuncu bağlandığında sunucu kapanıyor mu yoksa hata logları mı üretmeye başlıyor?

Stress testi, yazılımın **dayanıklılık limitlerini** görmemizi sağlar.

2.3 Endurance Testing (Dayanıklılık / Uzun Süre Testi)

Tanım: Sistemin uzun süre boyunca sabit yük (steady load) altında nasıl performans gösterdiğini ölçmek.
Hedef: Bellek sızıntısı (memory leak), performans düşüşü (performance degradation) veya kaynak tüketiminde artış olup olmadığını gözlemlemek.

Örnek Senaryolar:

- Netflix benzeri bir platformun 24 saat boyunca sürekli 10.000 kullanıcıya hizmet vermesi sırasında **RAM tüketimi artıyor mu?**
- Bir ERP (Kurumsal Kaynak Planlama) sisteminin günlerce aynı yük altında çalışması sonucu **CPU ve bellek kullanımı dengede kalıyor mu?**

Endurance testleri, sistemin **sürdürülebilirliğini** garanti altına almak için önemlidir.

2.4 Peak Testing (Pik Yük Testi)

Tanım: Sisteme kısa sürede **ani yük artışı (sudden traffic spike)** uygulanması.
Hedef: Trafik dalgalanmalarında (traffic fluctuations) sistemin tepkisini ölçmek.

Örnek Senaryolar:

- Bir bilet satış sitesinde ünlü bir konserin satışa açıldığı ilk dakikalarda **on binlerce kişinin aynı anda sisteme girmesi**.
- E-ticaret sitesinde Black Friday başladığında 5 dakikada gelen trafik dalgası.

Peak testleri ile sistemin **ani dalgalanmalara karşı refleksi** ölçülür.

2.5 Scalability Testing (Ölçeklenebilirlik Testi)

Tanım: Sistemin yeni kaynak (additional resources) eklendiğinde performansını artırabilme yeteneğinin ölçülmesi.
Hedef: Yatay ölçeklendirme (horizontal scaling → yeni sunucu eklemek) veya dikey ölçeklendirme (vertical scaling → CPU, RAM artırmak) ile sistem performansının iyileşip iyileşmediğini görmek.

Örnek Senaryolar:

- Kullanıcı sayısı iki katına çıktığında yeni sunucular ekleniyor. Bu durumda sistem **aynı hızda çalışabiliyor mu?**
- Bir uygulamanın veri tabanı RAM artırıldığında **sorgu süreleri düşüyor mu?**

Scalability testleri, sistemin **geleceğe hazır olup olmadığını** gösterir.

3. CPTS (Cloud Performance Test Service)

3.1 CPTS nedir? (What is CPTS?)

CPTS, Huawei Cloud'un sunduğu **bulut tabanlı performans test aracı**dır.

Özellikler:

- Yük testi (Load Testing)
- API testi (API Testing)
- Performans raporları (Response Time, TPS, Error Rate)
- Farklı bölgelerde test yapabilme (multi-region testing)

Avantajları:

- Lokal donanım kısıtlaması yok → binlerce kullanıcıyı cloud üzerinden simüle edebilirim.
- Test senaryoları kolay tanımlanır, tekrar kullanılabilir.
- Sonuçlar görsel raporlar halinde gelir, detaylı analiz yapılabilir.
- CI/CD pipeline ile entegre edilebilir → otomatik test akışı.

3.2 CPTS'de performans durumları oluşturmamız ne zaman gerekir? (When do we need to create performance cases on CPTS?)

Performans senaryoları CPTS üzerinde şu durumlarda oluşturulur:

- Yüksek trafik beklenen projeler (kampanya dönemleri, canlı yayınlar).
- Canlıya çıkmadan önce uygulamanın **dayanıklılığını** görmek.
- Uzun süreli yük altında sistemin kaynak kullanımını analiz etmek.
- Ani yük dalgalanmalarında sistemin cevap verme kabiliyetini test etmek.
- Yeni sunucular eklendiğinde sistemin **ölçeklenebilirliğini** doğrulamak.

3.3 CPTS'de test senaryoları nasıl oluşturulur? (How to create test cases on CPTS?)

Teorik olarak öğrendiğim adımlar:

1. CPTS arayüzüne giriş yapılır.
2. "Create Test Case" seçeneği seçilir.
3. Test türü belirlenir (Load, Stress, Endurance, Peak, Scalability).
4. Parametreler tanımlanır:
    - Kullanıcı sayısı (Number of Users)
    - Süre (Duration)
    - İstek frekansı (Request Frequency)
    - Bölge (Region)
5. Test çalıştırılır.
6. Sonuçlar raporlanır:
    - Response Time (yanıt süresi)
    - TPS (Transaction Per Second)

- Error Rate (hata oranı)
- Kaynak kullanımı (CPU, RAM, Disk IO)

---

4. Kapanış

Bugün **performans testlerinin türlerini** (Load, Stress, Endurance, Peak, Scalability) ve bunların hangi senaryolarda gerekli olduğunu öğrendim.

Ayrıca **CPTS'in bu testleri bulut tabanlı** şekilde yapmayı sağladığını kavradım. Böylece lokal kaynaklara bağımlı kalmadan, çok daha büyük ölçekli kullanıcı senaryolarını test edebilmenin mümkün olduğunu fark ettim.

Bana katkısı şu oldu:

- Her performans testinin farklı bir amaca hizmet ettiğini öğrendim.
- Bir projeyi canlıya almadan önce bu testlerin yapılmasının, kullanıcı deneyimini korumak için ne kadar kritik olduğunu anladım.
- Artık sadece fonksiyonelliğe değil, performansa da odaklanmam gerektiğini biliyorum.
- Bir sistemin gerçekten **yüksek ölçekli trafiğe hazır olup olmadığını** anlamanın yolunun bu testlerden geçtiğini kavradım.
- İlerde projelerde API'lerin veya web servislerin dayanıklılığını test ederken bu bilgileri doğrudan uygulayabileceğim.
