**SUNUCU İSTEK YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ**

YAZILIM LABORATUVARI-II PROJESİ

Gözde ÖRGÜ – Büşra ERKAN

170201005 - 170201018

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[gozde.orgu@gmail.com](mailto:gozde.orgu@gmail.com) [busraerkan39@gmail.com](mailto:busraerkan39@gmail.com)

**ÖZET**

*Sunucu İstek Yoğunluğunun Multithread İle Kontrolü Projesinde bir sunucuya gelen isteklerdeki aşırı yoğunluğu, multithread kullanarak alt sunucularla birlikte azaltmak amaçlanmıştır.*

*Proje Thread kullanımını göstermekte ve İşletim Sistemleri dersi konularını pekiştirmektedir.*

**1.Giriş**

Projede bizden multithread ile sunucu istek yoğunluğunu kontrol etmemiz beklenmektedir.

Threadler varlıkları bir işleme bağlı olan ve yine aynı anda birden fazla işi yapmaya yarayan yapılardır. Ancak thread’in var olması için önce bir işlemin var olması gerekir.

Multi-Threaded programlama bir programda aynı anda birden fazla işin yapılabilmesidir. Yani bir kod parçası işlemi gerçekleştirirken aynı anda ona paralel olarak bir başka kod parçasının çalışması demektir.

Projede istenenler şunlardır :

Uygulamada bir tane ana sunucu ve iki tane de alt sunucu bulunmaktadır. Ana sunucuya belirli aralıklarla istekler gelmektedir. Alt sunucular ana sunucudan istekleri alıp kendileri geri dönüş yapmaktadır. Alt sunucu oluşturucu tarafından kontrol edilen alt sunucular eğer %70 ve üzeri bir kapasiteye ulaşırsa yeni bir alt sunucu oluşturarak kapasitesini paylaşacaktır. %0 a ulaşırsa alt sunucu

ortadan kalkmalıdır. Ek olarak kapasite bilgileri ekrana yazdırılmalıdır.

**2.Temel Bilgiler**

Program Java dilinde yazılmış olup proje gelişiminde;

Tümleşik Geliştirme Ortamı olarak “NetBeans IDE” kullanılmıştır.

**3.Tasarım**

Projenin programlanma aşamaları aşağıda belirtilen başlıklar altında açıklanmıştır.

**3.1 Algoritma**

Proje üç sınıftan oluşmaktadır. MultiThreading.java sınıfı main metodunun bulunduğu sınıftır. MainThread.java ve SubThread.java sınıflarında thread işlemleri yapılmıştır.

Projede başlangıçta beş thread istenmiştir. Bunlar ana sunucu, alt sunucu 1, alt sunucu 2, alt sunucu oluşturucu ve sunucu takiptir. Program çalıştırıldığında MultiThreading.java sınıfında ana sunucu, iki tane alt sunucu, alt sunucu oluşturucu ve sunucu takip threadi oluşturulmuştur. Ana sunucu 10000 istek kapasitesine ve 0 doluluk oranına sahip olarak tanımlanmıştır. Alt sunucu 1 ve alt sunucu 2 5000 istek kapasitesine ve 0 doluluk oranına sahip olarak tanımlanmıştır. Alt sunucuların kapasite ve doluluk bilgileri subdoluluk ve subkapasite ArrayListleri’nde tutulmuştur. Alt sunucular, alt sunucu oluşturucu ve sunucu takip subthreads ArrayList’inde tutulmuştur. Ardından MainThread.java sınıfı çalıştırılmıştır. MainThread.java sınıfı MultiThreading.java sınıfından extends , Runnable sınıfından imlements edilmiştir. Thread start edildiğinde MainThread.java sınıfındaki run() fonksiyonu çalışır. Bu fonksiyonda mainThreadIslemi() fonksiyonu çağırılmıştır. mainThreadIslemi() fonksiyonunda 1-100 arasında rastgele sayı üretilerek istek oluşturulur ve ana sunucunun kapasitesi doldurulur. Eş zamanlı olarak kapasitenin izin verdiği ölçüde istek dönüş yapar. İstek dönüş 1-50 arasında rastgele üretilen sayılardan yapılır. Threadlerin birbiriyle senkronize olarak çalışması için bu işlemler senkronizasyon bloğu içinde yapılmıştır. İşlemler yapılırken lock kullanılmıştır.

SubThread.java sınıfı MultiThreading.java sınıfından extends, Runnable sınıfından implements edilmiştir. Subthreadler start edildiğinde SubThread.java sınıfındaki run() fonksiyonu çalışır. Bu fonksiyonda subThreadIslemi() fonksiyonu çağırılmıştır. Gelen threadin ID’sine göre işlem yapılır. subThreadIslemi() fonksiyonunda 1-100 arasında rastgele sayı üretilerek istek oluşturulur ve alt sunucuların kapasitesi doldurulur. Eş zamanlı olarak kapasitenin izin verdiği ölçüde istek dönüş yapar. İstek dönüş 1-50 arasında rastgele üretilen sayılardan yapılır. Alt sunucu oluşturucu sub thread ile mevcut alt sunucuların kapasitesi kontrol edilmiştir. Herhangi bir alt sunucunun kapasitesi %70 ve üzerinde ise yeni bir alt sunucu oluşturulmuştur ve kapasitesinin yarısı alt sunucuya aktarılmıştır. Eğer alt sunuculardan birinin kapasitesi %0’a ulaşmış ise mevcut olan alt sunucu sistemden kaldırılmıştır. Ancak en az iki alt sunucu sürekli çalışır durumdadır. Threadlerin birbiriyle senkronize olarak çalışması için bu işlemler senkronizasyon bloğu içinde yapılmıştır. İşlemler yapılırken lock kullanılmıştır.

Sunucu takip sub threadi ile sistemde mevcut olan tüm alt sunucuların bilgileri ekrana yazdırılmıştır.

**3.2 Sınıfların Özellikleri**

*MultiThreading.java*

Main metodunun bulunduğu sınıftır. Ana sunucu, alt sunucu 1, alt sunucu 2, alt sunucu oluşturucu ve sunucu takibi threadleri bu sınıfta oluşturulmuştur. Threadlere ait ilk kapasite ve doluluk oranları burada tanımlanmıştır.

*MainThreading.java*

MultiThreading.java sınıfından extends, Runnable sınıfından implements edilmiştir. Runnable sınıfı thread işlemleri için kullanılan bir sınıftır. Ana sunucu işlemleri bu sınıfta gerçekleştirilir. Kapasite ve doluluk miktarları burada güncellenmiştir.

*SubThread.java*

SubThread.java sınıfı MultiThreading.java sınıfından extends, Runnable sınıfından implements edilmiştir. Runnable sınıfı thread işlemleri için kullanılan bir sınıftır. Alt sunucu işlemleri bu sınıfta yapılmıştır. Kapasite ve doluluk miktarları burada güncellenmiştir. Alt sunucuların bölünmesi ve sistemden silinmesi bu sınıfta gerçekleştirilmiştir.

**3.3 Kullanılan Fonksiyonlar**

*public void run()*

Hem MainThread,java hem de SubThread.java sınıfında kullanılmıştır. Runnable sınıfından dolayı override edilen fonksiyondur. Thread start edildiğinde run() fonksiyonundaki işlemler gerçekleştirilir.

*public synchronized void mainThreadIslemi()*

MainThread.java sınıfında kullanılan bir fonksiyondur. Main thread start edildiğinde çalıştırılan run() fonksiyonunda çağırılır ve threade ait işlemler gerçekleştirilir. Kapasite ve doluluk miktarlarının güncellenmesi, istek kabulü ve istek dönüşü işlemleri bu fonksiyonda yapılır.

*public synchronized void subTreadIslemler2()*

SubThread.java sınıfında kullanılan bir fonksiyondur. Sub threadler start edildiğinde çalıştırılan run() fonksiyonunda çağırılır ve threadlere ait işlemler gerçekleştirilir. Kapasite ve doluluk miktarlarının güncellenmesi, istek kabulü ve istek dönüşü, alt sunucu oluşturma ve sistemden thread silme işlemleri bu fonksiyonda yapılır.

**4. Karşılaşılan Sorunlar**

* Güncellenen doluluk bilgisini sınıflar arasında kullanamadık. Değişkeni çağırdığımız sınıfta tuttuğu değer güncellenmiş bir şekilde alınamadı. Bu sorunu atomic integer tipinde bir değişken kullanarak çözdük.
* Bütün threadler eş zamanlı çalıştığından dolayı istek kabulü ve istek dönüşü sürelerini kesin olarak ayarlayamadık.

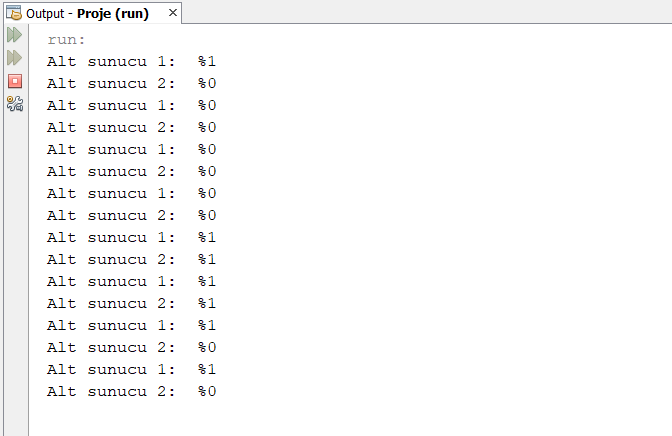
**5.Akış Şeması**

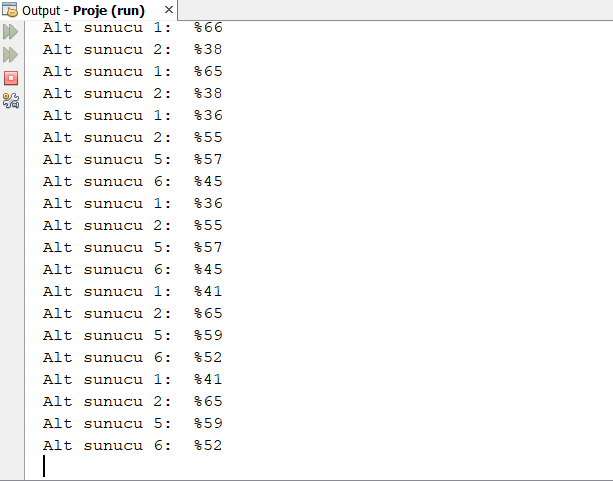


**6.UML Diyagram**



**7.Deneysel Sonuçlar**

****

****

**8.Kaynakça**

* Thread için ve Multithreading için :

<http://bilgisayarkavramlari.sadievrenseker.com/2010/03/22/thread-iplik/>

<https://isletimsistemleriveuygulamalari.wordpress.com/category/threads-multithreading-is-parcaciklari/>

<http://umiitkose.com/2015/04/java-thread-islemleri/>

<https://ufukuzun.wordpress.com/2015/02/14/javada-multithreading-bolum-1-merhaba-thread/>

<https://medium.com/gokhanyavas/javada-multithreading-bbc6a9181772>

* 74-75-76-77.videolar

<https://www.youtube.com/watch?v=QmwC_p0OGmI&list=PLHfYetw_BGF-Gm_MsqKApw5nHPuHsytr3>

* Atomic integer için :

<https://howtodoinjava.com/java/multi-threading/atomicinteger-example/>

* Lock için:

<https://ufukuzun.wordpress.com/2015/05/05/javada-multithreading-bolum-9-yeniden-girilir-kilitler-re-entrant-locks/>

* Senkronizasyon için :

<https://www.tutorialspoint.com/java/java_thread_synchronization.htm>

<https://docs.oracle.com/javase/tutorial/essential/concurrency/locksync.html>