Dağıtık Sistemlere Giriş

Ödev – 2

**Servis Odaklı Mimari ile Paralel Dağıtık Matrix Çarpımı Uygulaması**

**GRUP 7**

**Bilgisayar Mühendisliği Bölümü**

**Kocaeli Üniversitesi**

Özet

*Bu projede matris çarpma işlemini dağıtık sistemlerde gerçekleştirme amacına yönelik iki uygulama ve dağıtık sistemler olmadan matris çarpma işlemini yapan bir uygulama yapılmıştır. Dağıtık mimariye uygun olarak oluşturulan uygulamalardan biri client biri de server’dır. Server uygulamasını iki bilgisayar çalıştırmakta client uygulamasını ise bir bilgisayar çalıştırmaktadır. Böylelikle server olan iki bilgisayar paralel olarak eş zamanlı işlem gerçekleştirerek verimliliği sağladığı görülmeye çalışılmıştır. Bu grafiksel olarak vs. yollarla rapor edilmiştir.*

# Giriş

Çalışmanın konusu farklı bilgisayarlarda eş zamanlı olarak bir işlemin gerçekleştirilmesine yönelik olarak matris çarpım işleminin paralel olarak gerçekleştirilmesidir. Dağıtık sistemler mantığı bu proje ile anlaşılmaya çalışılmıştır.

Yapmış olduğumuz projede öncelikle tek bilgisayarda matris çarpım işlemini gerçekleştirdik. Daha sonra farklı iki bilgisayarda paralel olarak eş zamanlı matris çarpım işlemini gerçekleştirdik. Çıkan sonuçları grafiksel olarak da matlab’da raporlayarak, dağıtık sistem mimarisi uygulandığı zaman daha verimli bir sonuç alındığını gözlemlemiş olduk. Dağıtık sistem mimarisini uygulamak için ise SOA yani servis yönelimli mimariyi kullanarak web servis uygulamalarını gerçekleştirdik.

İş hayatında ve günümüzde profesyonel olarak kullanılan yazılımlarda da dağıtık sistemli mimariye uygun olarak projeler geliştirilmekte ve çok büyük kitlelerin kullanımına imkan sağlayan uygulamalar kullanılmaktadır.

# Temel Bilgiler

Projeyi gerçeklerken NetBeans IDE’sini kullandık. Dil olarak Java’yı kullandık. Web servis uygulamasını gerçekleştirmek için SOA mimarisini kullandık. Bunu Java ile tümleştirilmiş haline ise JAX-WS denmektedir araştırmalarımız sonucunda elde ettiğimiz bilgilere göre.

# Uygulamalar Hakkında

1. **Normal Çarpım Uygulaması**

Class’ ımızın ismine Normal\_Carpim\_Jframe adını verdik. Bu class’ da tek makinede boyutları dışarıdan girilen iki kare matrisin çarpımını gerçekleştireceğiz. Değişken olarak matris1, matris2 ve sonuç\_matris adında iki boyutlu diziler tanımladık. Float tipinde toplam ve ort değişkenleri de matris çarpma işlemlerinde kullandığımız değişkenlerdir.

Ara yüzümüzde jTextField1 jLabel1 jButton1 jLabel2 jScrollPane1 jTextArea1 nesnelerini kullandık. TextField nesnesi dışarıdan matris boyutu girmek için, Label1 ve Label2’yi kullanıcıyı bilgilendirmek için kullandık. Buton1 nesnesi çarpma işlemini başlatan komutu vermeye, textarea1 de çarpma işleminin sonucunu göstermeye yarar.

Buton’a bastığımız anda nanoTime\_bas değişkeni ile çarpma işlemini başlattığımız zamanı tutarız. Satir\_sayisi değişkeni ile ise dışarıdan girdiğimiz kare matris boyutunu tutarız. Yukarıda tanımladığımız matris dizilerine de bu boyutu veririz ve matrislerimizi boyutlandırmış oluruz.

Random nesnesi ile iç içe iki for kullanarak matris1 ve matris2 dizilerimizin elemanlarını rastgele olarak belirliyoruz.

Matrislerin çarpma işlemini, birinci matrisin satırı ile ikinci matrisin sütununu tek tek çarparak elde ettik. Döngü içinde bu sonuçları topla değişkenine ardından iki boyutlu sonuç\_matris adlı matrisimize attık.

File, FileWriter, BufferedWriter nesneleriyle dosya1, yazici1, yaz1 değişkenlerini oluşturduk. Bu işlemleri üç kere yapıyoruz. Çünkü üç tane matrisimiz var ve bunları metin belgelerine yazdırmamız gerek. Dosya1, dosya2 ve dosya3 değişkenlerine oluşturacağımız metin belgelerinin yollarını tanımlıyoruz. Null olarak tanımladığımız yazici1 değişkenini, dosya1 ve false parametresini alan FileWriter nesnesi ile oluşturuyoruz. BufferedWriter nesnesi de yazici1 parametresini alır ve yaz1 değişkeni oluşturulur.

Son olarak yaz1 değişkenine iç içe iki for ile matrisin elemanlarını tek tek yazdırıyoruz ve satır bitiminde alt satıra geçiyoruz ve devam ediyoruz. En sonda ise yaz1’ i kapatıyoruz. Bu işlemleri diğer iki matris dizisi için de yapıyoruz.

nanoTime\_bit değişkeni ile çarpma işleminin bitiş zamanını alıyoruz. Fark değişkenine başlangıç ve bitiş zamanlarının farkını atıyoruz ve süreyi TextArea1 nesnesine yazdırıyoruz.

Normal\_Carpim\_Jframe class’ ını run fonksiyonunun içinde çağırarak çalıştırdık.

1. **Dağıtık Sistem Uygulamaları**
2. **Client Uygulaması**

**(DenemeSon\_Ikı)**

**1- myJFrame (main class)**

Client tarafında ilk çalışan(main) class’tır. İlk olarak “islem\_kuyruk” ve ” matris\_2\_kuyruk” adında iki adet kuyruk tanımlıyoruz. Bu kuyruklar dışarıdan girilen matris boyutuna göre işlem sırasını tutar. Tek boyutludurlar. Ardından “sonuc\_listesi” adında dizi listesi tanımladık. Bu dizi listesi web servislerden dönen satırları tutar. İki boyutludur. İlk boyut dönen satırın numarasını tutar. Çünkü satırlar web servislerden karışık şekilde gelebilir. İkinci boyutu ise satırı tutar. Matrisin boyutunu kaca\_kac değişkeni, matrisleri ise matris1 ve matris2 değişkenleri tutar.

File, FileWriter, BufferedWriter nesneleriyle dosya1, yazici1, yaz1 değişkenlerini oluşturduk. Bu işlemleri iki kere yapıyoruz. Çünkü iki tane matrisimiz var ve bunları metin belgelerine yazdırmamız gerekir. Dosya1 ve dosya2 değişkenlerine oluşturacağımız metin belgelerinin yollarını tanımlıyoruz. Null olarak tanımladığımız yazici1 değişkenini, dosya1 ve false parametresini alan FileWriter nesnesi ile oluşturuyoruz. BufferedWriter nesnesi de yazici1 parametresini alır ve yaz1 değişkeni oluşturulur.

Random nesnesi ile iç içe iki for kullanarak matris1 ve matris2 dizilerimizin elemanlarını rastgele olarak belirliyoruz.

Son olarak yaz1 değişkenine iç içe iki for ile matrisin elemanlarını tek tek yazdırıyoruz ve satır bitiminde alt satıra geçiyoruz ve devam ediyoruz. En sonda ise yaz1’ i kapatıyoruz. Bu işlemleri diğer matris dizisi için de yapıyoruz.

Ara yüzümüze bakacak olursak dışardan kare matris boyutunu girmek için bir metin kutusu ve çarpma işlemini başlatacak buton vardır. En altta ise çarpma işleminin ne kadar sürede yapıldığını gösteren bir metin alanı vardır.

Buton’ a bastığımızda matris boyutunu kaça\_kac değişkinimize atıyoruz. islem\_kuyruk ve matris\_2\_kuyruk değişkenleri matris boyutuna göre doldurulur. islem\_baslangic değişkeni ile başlangıç zamanı tutulur. rasgele\_sayi\_dosya fonksiyonuna matris boyutu gönderilerek matrisler oluşturulur ve metin belgelerine yazdırılır.

Islem\_1\_Yolla class’ ımızda kullanmak üzere thread tanımlarız. Islem\_2\_Yolla class’ ımızda kullanmak üzere de thread tanımlarız. Ayrıca ikitane daha thread islem1 ve işlem2 class’ları için iki adet daha thread tanımlanır.

Islem\_1\_Yolla matris boyutunu alır ve matris2 yi satır satır thread aracılığıla web servise gönderir.

Islem\_2\_Yolla da aynı işlemi yapar fakat; farklı bilgisayardaki web servise yollar.

Islem\_1 ve Islem\_2 ise web serviste satır satır çarpma işleminin yapılmasını sağlar.

Sonuc\_Yazdir class’ı thread kullanarak web servisten gelen matrislerin satırlarını alır ve sıralı bir şekilde tutar.

Class’larda thread kullanmak için Runnable diye tanımlamamız gerekir. Class’ın içinde run metodunu çağırarak thread’i kullanmamız gerekir.

1. **İslem\_yolla1.java ve islem\_yolla2.java**

public Islem\_1\_Yolla(int satir\_sayisi, LinkedBlockingQueue<Integer> matris\_2\_kuyruk, List<Integer> matris2)

Islem\_1\_Yolla sınıfımız 3 adet parametre almaktadır. “satir\_sayisi” değişkeni 2. Matrisimizin(yani çarpılacak matrisin) boyutunu tutar. Çarpılacak matrisimiz kare matris olduğundan satır ve sütun boyutları aynıdır. Yani bu değişkenin tuttuğu değer aynı zamanda 2. Matrisimizin boyutunun da sayısal değeridir.

“matris\_2\_kuyruk” parametresi ise bağlı blok kuyruğu(LinkedBlockingQueue) tipindedir. Bu kuyruğu kullanmaktaki amacımız ise şöyledir: İstemcinin kod kısmında bulunan 2. Matrisi, olduğu gibi web servislere gönderemiyoruz. Bütün olarak gönderemediğimiz için 2. matrisi satır satır web servislere göndereceğiz. Bu işlemi de kuyruklar sayesinde gerçekleştireceğiz. 2. matrisin satır sayısı kadar olan sayıları sırasıyla bu kuyruğa ekledik. Yani matrisimiz 20x20 boyutunda olursa bu kuyruğa 1’den 20’ye kadar sayıları vermiş oluruz. Böylece 2. Matrisimizin satırlarını oluştururken, kuyruktan sırayla sayıları alıp, ilgili satır numarasına ait 2.matrisimizin satırlarını oluşturuyoruz.

Son parametremiz ise” matris2” dir. İstemcimizin kod kısmında matris2’yi “Islem\_1\_Yolla” sınıfına gönderiyoruz. Buradan da “matris\_2\_kuyruk\_1”kuyruğunu kullanarak ilgili satır numarasına ait 2.matrisimizin satırlarını oluşturmak üzere web servislere gönderiyoruz. Web servis’lerde 2.matrisin satırlarını tek tek alıp tekrar düzenleyip matrisin tamamını oluşturan metod “ikinciMatrisAl” metodudur. Bu yüzden “Islem\_1\_Yolla” sınıfında while döngüsünü kullanarak 2 matrisin satır sayısı kadar döndürüyoruz ve “ikinciMatrisAl” metodunu çağırarak web servislerin 2. Matrisi satırlarıyla beraber oluşturmasını sağlıyoruz.

public Islem\_2\_Yolla(int satir\_sayisi, LinkedBlockingQueue<Integer> matris\_2\_kuyruk, List<Integer> matris2)

Islem\_1\_Yolla sınıfımız 3 adet parametre almaktadır. “satir\_sayisi” değişkeni 2. Matrisimizin(yani çarpılacak matrisin) boyutunu tutar. Çarpılacak matrisimiz kare matris olduğundan satır ve sütun boyutları aynıdır. Yani bu değişkenin tuttuğu değer aynı zamanda 2. Matrisimizin boyutunun da sayısal değeridir.

“matris\_2\_kuyruk” parametresi ise bağlı blok kuyruğu(LinkedBlockingQueue) tipindedir. Bu kuyruğu kullanmaktaki amacımız ise şöyledir: İstemcinin kod kısmında bulunan 2. Matrisi, olduğu gibi web servislere gönderemiyoruz. Bütün olarak gönderemediğimiz için 2. matrisi satır satır web servislere göndereceğiz. Bu işlemi de kuyruklar sayesinde gerçekleştireceğiz. 2. matrisin satır sayısı kadar olan sayıları sırasıyla bu kuyruğa ekledik. Yani matrisimiz 20x20 boyutunda olursa bu kuyruğa 1’den 20’ye kadar sayıları vermiş oluruz. Böylece 2. Matrisimizin satırlarını oluştururken, kuyruktan sırayla sayıları alıp, ilgili satır numarasına ait 2.matrisimizin satırlarını oluşturuyoruz.

Son parametremiz ise” matris2” dir. İstemcimizin kod kısmında matris2’yi “Islem\_2\_Yolla” sınıfına gönderiyoruz. Buradan da “matris\_2\_kuyruk\_2”kuyruğunu kullanarak ilgili satır numarasına ait 2.matrisimizin satırlarını oluşturmak üzere web servislere gönderiyoruz. Web servis’lerde 2.matrisin satırlarını tek tek alıp tekrar düzenleyip matrisin tamamını oluşturan metod “ikinciMatrisAl” metodudur. Bu yüzden “Islem\_2\_Yolla” sınıfında while döngüsünü kullanarak 2 matrisin satır sayısı kadar döndürüyoruz ve “ikinciMatrisAl” metodunu çağırarak web servislerin 2. Matrisi satırlarıyla beraber oluşturmasını sağlıyoruz.

1. **Islem\_1 ve Islem\_2**

**İslem1**

Bu class’ımızda ilk olarak Islem\_1 fonksiyonu çalışır. Çümkü yapıcı fonksiyondur. Gerekli parametreleri alır.

Class’mız thread’le çalışmaktadır. Bu yüzden run metotu çalışır. Bu metot için matris2 nin tüm satırlarının web servise gitmiş olması gerekir. Öncelikle bunu kontrol ediyoruz. Eğer koşul gerçekleşti ise islem1\_kuyruk değişkenindeki sırada olan elemanı alıyoruz ve bu bizim satır numaramız olur. matris1\_satir list değişkeninin içini temizleriz. Sonra matris1\_satir’a döngü ile matris1 deki satırı satır numarasına göre tek tek ekleriz.

Web servise carpim web metodu sayesinde satır numarasını ve çarpılacak matris satırını yollarız. sonuc\_listesi list değişkenine dönen çarpım sonuçlarını satırlar halinde ekliyoruz.Carpim metodu web servisten implement edilmiştir. Web serviste çalışır.

Son olarak sonuç listesinin boyutu satır sayısı ile aynı olduğu zaman çarpıım sonuçlanmış demektir. MyJFrame class’ındaki islem\_bitis değişkenine erişilerek çarpım süreci elde edilir.

**İslem2**

İslem1 ile aynıdır. Yalnızca islem1 class’ı birinci bilgisayardaki web serviste çalışır. İslem2 class’ı ise ikinci bilgisayardaki web serviste çalışır.

1. **Sonuc\_yazdir**

Class ında web servislerden gelen sonuçları txt dosyasına yazdırma işlemini yapıyoruz. Bunu thread da çalıştırıyoruz. İlk olarak D://txt\_dosyaları için de matris\_sonuc\_satır-say isminde txt dosyası yoksa yeni oluşturuyoruz. Varsa üzerine yazıyoruz. Listede tuttuğumuz sonuçları ilk elemanı olan satır no değerleri ile sıralayıp txt dosyasına sıralı bir şekilde yazdırıyoruz. Bu işlem tüm çarpım işlemleri bittiğinde başlamaktadır.

1. **Web service (server) Uygulaması**

**(MatrisWebservice1)**

Web servis uygulamasında 3 adet metot kullanılmıştır. Bunlar : Carpim(), ikinciMatrisAl(), matrisAyarla() metotlarıdır.

* **Carpim**

Carpim metodu, birinci matrisin o an gelen satırının ikinci matrisin bütün sütunlarıyla çarpımı sonucu sonuç matrisinin ilgili satırının oluşturulduğu metottur. Burada oluşturulan ilgili satır client tarafına satır numarasıyla birlikte döndürülür.

* **ikinciMatrisAl()**

Bu metotta istemci tarafında threadler ile gönderilen ikinci matrisin satırları alınır sırayla ikinci matris satır satır oluşturulur.

* **matrisAyarla()**

Bu metot web servis tarafında, istemciden gönderilen matris uzunluğuna göre matrisleri oluşturur.

# Sonuçlar

Bu çalışmanın sonucunda dağıtık sistemler mimarisinin yararlarını belirgin bir şekilde öğrenmiş olduk. Eğer bir işlem paralel olarak birkaç bilgisayara yaptırılıyorsa bu durumda daha verimli bir sonucun elde edildiğini görmüş olduk.

Projede thread kullanarak uygulamalara eş zamanlı işlem yaptırma olayını gerçekleştirmiş olduk. Bu bize thread ile ilgili bilgiler kattı.

SOA mimarisini öğrenmiş olduk.

Java uygulaması geliştirme konusunda bir nebze de olsa pratiklik kazanmış olduk.

Java’da web servis kullanma olayını öğrenmiş olduk.

# Kaynakça

[1] **Serkan Yazıcıoğlı,** SOA Nedir? SOA’nın S’si, SOA’nın Getirileri, SOA Nerelerde Kullanılmalı

<http://blog.serkanyazicioglu.com/2011/12/soa-nedir/>

[2] **Tolga Ayaş**, SOA’nın Hedefleri ve Standartları

<http://www.yazilimdevi.com/Makaleler-1049-soa-nedir.aspx>

[3] **Yunus(Author of the quoted web site),** Net Beans ile Java Web Service (JAX-WS) Uygulaması

<http://www.yunusmete.com/2014/06/13/java-web-service-jax-ws-uygulamasi/>

[4] **Ahmet Sayar,** Yüz yüze görüşme sonucu alınan öneriler doğrultusunda Matris çarpımının paralel gerçekleştirilmesi için gereken algoritmanın oluşturulması

[5] **Emrah Mete,** Java’da Runnable yoluyla thread kullanımı.

<http://emrahmete.wordpress.com/2011/10/06/javada-thread-yapisi-ve-kullanimi-hakkinda-ipuclari/>

[6] **Yunus Mete,** Java ile Restful Web Servisler

<http://www.yunusmete.com/2014/10/12/java-restful-web-servisler/>