A red text on a white background

Description automatically generated

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

İŞLETİM SİSTEMLERİ ÖDEV-1

TESLİM TARİHİ

24.03.2024

HAZIRLAYAN

MERT TOSUN

210601027

BİLGİSAYAR MÜHENDİSLİĞİ

DERSİN EĞİTMENİ

Assist. Prof. Dr. Kemal Çağrı Serdaroğlu

Ödevin Amacı

Bu ödevin amacı, thread kullanarak eş zamanlı olarak çalışan işlemlerin senkronizasyon olmadan paylaşımlı bir kaynak alanına erişimini öğrenmektir. Ayrıca bu ödevi yaparak thread’leri nasıl oluşturacağımızı ve bu thread’lerin belirli görevleri gerçekleştirebileceğini görmüş oluruz.

THREAD KAVRAMI NEDİR

Bir process’in birden fazla işi aynı anda yapmasını sağlayan yapılara thread denir. Bir process bünyesinde bir ya da birden fazla thread barındırabilir. Thread’ler çok çekirdekli işlemcilerde farklı çekirdeklerde eş zamanlı olarak çalıştırılabilirler.

A KISMI

Ödevin ilk kısmında thread yani iş parçacıklarını kullanarak, oluşturduğumuz array ile bazı işlemler yaparak bu işlemlerin gerçekleşme sürelerini incelemeye çalıştım. İlk olarak arraye rastgele sayılar atayarak işe başladık ve bu rastgele sayıları alt kümelere bölerek bazı işlemler yaptık. Bu işlemleri yaparken işleme başladığımız kısım ile bitirdiğimiz kısmı nanoTime fonksiyonu ile belirterek süre hesaplaması yaptık. Arrayi bölerek oluşturduğumuz alt kümelere toplama , ortalama , en büyük ve en küçük bulma gibi işlemleri yaparak bitirdik.

B KISMI

Bu kısımda ise yine A kısmındaki gibi thread yani iş parçacıklarını kullanarak matrisli bir yapı oluşturduk. Bu yapıda bir n değişkeni tanımlayarak NXN boyutunda bir matris oluşturup bu matrisin içini rastgele sayılarla doldurduk. Ardından her bir satır için thread oluşturup işlemlerimizi yaptık. İşlemleri yaparken matrisleri baştan sona tarayarak en büyük bulma gibi işlemleri yapıyoruz.

C KISMI

Bu kısımda ise thread kullanmadan array oluşturarak toplama , en büyük bulma , en küçük bulma, ve ortalama işlemlerini yapıyoruz. Yine aynı şekilde arraye rastgele değerler atadık. İşlem sonuçlarını tutmak için arrayler oluşturduk. Daha sonra bir döngü oluşturarak işlemleri gerçekleştirdik.

C KISMI 2. TÜR

Thread kullanmadan matrisli işlem yapmak istersek, yine NXN boyutunda bir matris oluşturup içine rastgele değerler atayarak işleme başlıyoruz. Başlangıç zamanı için nanoTime fonksiyonunu koyuyoruz. Her satırda işlemi yapmak için bir döngü oluşturuyoruz. Sonra sonuçları ekrana yazdırıp süre fonksiyonunu sonlandırıyoruz.

KULLANILAN METOTLAR

* findMax : Alt kümelerdeki en büyük elemanı bulmak için bu metodu oluşturduk ve arrayde gezerek bulduğumuz her en büyük i. adımı max değişkenine atıyoruz.
* findMin : Alt kümelerdeki en küçük elemanı bulmak için bu metodu oluşturduk ve arrayde gezerek bulduğumuz her en küçük i. sonucu min değişkenine atıyoruz. Ve sonucu döndürüyoruz
* findSum : Başlangıçta toplam değerini 0’a eşitliyoruz ardından arraydeki değerleri gezerek(alt kümedeki veya matris için o satırdaki) toplama işlemi yapıp return ifadesiyle toplamı döndürüyoruz.

* findAverage: Toplam değerini çekiyoruz ardından array uzunluğuna bu değeri bölüyoruz.

TRY-CATCH YAPISI İLE INTERRUPTED EXCEPTİON ÖNLEME

Kodu yazarken threadlerle alakalı kaynak incelemesi yaptığımda bir try catch yapısıyla karşılaştım bu yapı try-catch bloğu kullanarak InterruptedException'ı yakalamak, programın hatasız ve beklenmedik durumlarda beklenen şekilde işlemesini sağlamada kullanıldığını öğrendim. Eğer bir InterruptedException oluşursa ve bu durum işlenmezse, program çökebilir. Bu hatayı yakalayarak programın çökmesini önleriz.

System.nanoTime() Fonksiyonu :

Java'da sistem saati çözünürlüğüne bağlı olarak nanosaniye cinsinden bir zaman döndüren bir metottur. Ölçüm yapmak istediğimiz kısımda başlangıç ve bitiş olarak tanımlayarak işlemin süresini ölçeriz.

KARŞILAŞTIRMA VE TEST

**Thread kullanarak Ve Thread kullanmadan array yapısı karşılaştırma;**

Değişken sayımız N:10

Alt küme sayımız M:5

Thread kullanarak; Thread kullanmadan;

ARRAY: 1 6 4 9 7 6 10 4 4 3 ARRAY: 3 3 4 6 3 8 9 5 6 6

Toplam süre: 1698 mikro saniye Toplam işlem süresi: 16414 mikro saniye

ARRAY: 3 7 1 4 9 6 5 2 1 4 ARRAY: 1 6 4 5 2 4 1 9 5 4

Toplam süre: 2261 mikro saniye Toplam işlem süresi: 19177 mikro saniye

Değişken sayımız N:200

Alt küme sayımız M:10

Thread kullanarak; Thread kullanmadan;

ARRAY : 200 rastgele değer ARRAY: 200 rastgele değer

Toplam süre: 1902 mikro saniye Toplam işlem süresi: 17463 mikro saniye

Thread kullanarak ve thread kullanmadan array yapısını karşılaştırmamız gerekirse thread kullandığımızda aldığımız süre değerleri daha küçük değerler oldu bunu farklı N ve M değerlerinde denediğimizde yine aynı değerleri aldık fakat bu her zaman böyle de çıkmayabilir bazen gereksiz sistemi yorabilir. Oluşturduğumuz sisteme göre thread kullanmak yukardaki değerlerde gözüktüğü gibi büyük performans farklarına fayda sağlayabilir.

**Thread kullanarak Ve Thread kullanmadan matrix yapısı karşılaştırma;**

Değişken sayımız N: 10 X 10

Thread kullanarak; Thread kullanmadan;

**ekran görüntüsü, yazı tipi, metin, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, yazı tipi, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Toplam işlem süresi: 1877 mikro saniye Toplam işlem süresi: 15178 mikrosaniye

Değişken sayımız N: 100 X 100

Toplam işlem süresi: 65909 mikro saniye Toplam işlem süresi: 56710 mikro saniye

Toplam işlem süresi: 56341 mikro saniye Toplam işlem süresi: 72778 mikro saniye

Toplam işlem süresi: 35237 mikro saniye Toplam işlem süresi: 42440 mikro saniye

Matrix kullanarak testleri gerçekleştirdiğimizde ise genel olarak thread kullanımının daha avantajlı olduğunu görüyoruz ama bazen threadsiz olan modelinde daha az sürdüğü oluyor kesin sonuç olmamakla beraber genelde thread bu modellerde daha avantajlı bir süre yönetimi sağlıyor.

**Sonuç**

Yaptığımız programlama ve aldığımız sonuçlar doğrultusunda bir sonuç çıkarmamız gerekirse thread yani iş parçacıkları kullanmak eş zamanlı bir şekilde işlem yürütmek bize bazı avantajlar sağlıyor bu avantajlar aynı anda birden fazla işlem yürütme ve arayüzü olan projelerde kullanıcı arayüzü hemen cevap verirken thread'lerin arka planda çalışarak işlemi bitirmesi ve sonucu getirmesi gibi şeyler olabilir. İşlemciyi daha verimli kullanmamıza olanak sağlar. Yazılımı daha modüler hale getirir. İşlemleri ve görevleri parçalara bölerek, bu parçaları farklı thread'lerde yürüterek daha kolay bakım yapılmasını ve kodun daha anlaşılır olmasını sağlar.

KAYNAKÇA

<http://www.ugurkizmaz.com/blog/>

<https://academy.patika.dev/courses/java-102/thread>

<https://www.yusufsezer.com.tr>