Threading

ALİ KUTAY BİLGİLİOĞLU
KOCAELİ ÜNİVERSİTESİ
Bilgisayar Mühendisliği(i.ö.)
200202108@kocaeli.edu.tr

ÖZET

Çok sayıda olan büyük verileri daha hızlı işlemek ve arayüz kullanımı ile masaüstü uygulaması gerçekleştirmek amacıyla oluşturulan proje. Projede Visual Studio ortamı ile beraber C# Programlama dili kullanılmıştır. Arayüz için ise .Net Framework desteğine sahip olan ve multi-threadlerle uyumlu çalıştığı bilinen Windows Forms kullanılmıştır.

I. GİRİŞ

Tasarlanan masaüstü uygulamasında kullanıcı kaç tane veri işleneceği, kaç thread kullanılacağı, Edit Distance'ın hangi sütünda yapılacağını ve sonuç olarak hangi sütunları göstermesi gerektiğini programa rahatça belirtebilir. Bunun sonucunda program ekrana kullanıcının istediği şartlar doğrultusunda çalışarak arayüzdeki tabloya verileri bastırır. Aynı zamanda çalışma durunca program toplam runtime süresini ve threadlerin her birinin runtime'ını göstermektedir.

C# Threading sayesinde kullanıcının girdiği thread kadar thread aynı fonksiyonda çalışır, birbirleri ile bir Global değişken üzerinden haberleşir, böylece senkronizasyon için threadleri bekletmeye gerek kalmaz. Bu haberleşme sayesinde threadler aynı fonksiyon içinde çalışsa da farklı veriler üzerine çalışarak ulaşılması gerekilen sonuca daha hızlı ulaşılmasını sağlar.

II. Temel Bilgiler

Program C# dili ile yazılmış cpu-bound işlemler Program.cs dosyasında yapılırken I/O-bound işlemler Form.cs dosyasına yazılmış, gerekilen modeller rows.cs dosyasında belirtilmiştir.

III. Yöntem Ve Program Mimarisi

Bu kısımda programın farklı özelliklerini oluşturmak için kullandığımız araçlar ve yöntemler üzerinde durularak ayrıntılı olarak bilgi verilecektir. Program mimarisi daha detaylı bir şekilde açıklanacaktır.

A. Verilerin Filtrelenmesi:

Verilerin filtrelenmesinde standart yöntemlerin dışına çıkılarak Windows'un kendi içinde bulundurduğu özellik kullanıldı. Windows Powershell'de herhangi ekstra paket gerekmeden .csv (comma seperated values) dosyaları üzerinde oynamalar, ekrana yazdırma gibi işlemler yapılabiliyor. Bu işlemler özellikle de büyük boyutlu .csv dosyalarının içinde ne olduğuna bakmak için kullanılabilir, çünkü büyük boyutlu .csv dosyalarını bazı programlar açmayı reddetmektedir. Csv dosyasının 6 sütuna düşürülmesi de Powershell kullanılarak yapılmıştır.

```
Product     : Checking or savings account
Issue       : Managing an account
Company     : BANK OF AMERICA, NATIONAL ASSOCIATION
State       : WA
ZIP code    : 98312
Complaint ID : 3229361

Product     : Credit reporting, credit repair services, or other personal consumer reports
Issue       : Improper use of your report
Company     : Experian Information Solutions Inc.
State       : TX
ZIP code    :
Complaint ID : 3228597
PS C:\Users\akuta\yazlab1.2> Import-Csv rows.csv|select Product,Issue,Company,State,"ZIP code","Complaint ID"|Export-Csv yazlab.csv -NoTypeInformation
```

Daha sonrasında 6 sütuna düşürülmüş .csv verisi C#'da LinqToCsv eklentisi kullanılarak okunmuştur ve işlenmesi üzerine listeye atılmıştır. Listeye veriler atılırken null veriler kontrol edilmiştir ve null değer içeren satırlar listeye eklenmemiştir.

```
if (row.Product == null || row.Company == null || row.ID == null || row.Issue == null || row.State == null || row.Zip_Code == null)
{
    continue;
}
```

Aynı zamanda satırlarda bulunan tüm noktalama işaretleri kaldırılmıştır.

```
row.Product = row.Product.Replace(".", "");
row.Company = row.Company.Replace(".", "");
row.Issue = row.Issue.Replace(".", "");
row.State = row.State.Replace(".", "");
```

B. String Karşılaştırması

String karşılaştırması için, daha sonra thread içinde kullanılması için bir fonksiyon yazılmıştır. Bu fonksiyonda Stringler enumerable tiplere çevrilmiştir. İlk başta hangi Stringin uzun olduğuna bakılmıştır, aynı olan kelimeler bir set'de saklanmıştır, daha sonra bu set'in uzunluğunun, uzun olan Stringin uzunluğuna bölünüp, yüzdeye çevrilmiştir.

```
if (set2.Count() > set1.Count())
{
    diff = set2.Where(e => !set1.Any(f => f == e)).ToList();
    oran = 100 - (diff.Count() * 100) / set2.Count();

}
else
{
    diff = set1.Where(e => !set2.Any(f => f == e)).ToList();
    oran = 100 - (diff.Count() * 100) / set1.Count();
    //Console.WriteLine((diff.Count()*100)/set2.Count());
}
//Console.WriteLine(diff.Count());
return oran;
```

C. Thread Metodu

Her verinin diğer her bir veri ile karşılaştırılması gerekiyor. Bu yapı için iki adet iç içe döngü yapısı gerekiyor. Önce ilk elemanı diğer tüm elemanlarla karşılaştırdıktan sonra ikinci elemanın ilk ve/veya kendinden önceki veriler hariç yine tüm veriler ile karşılaştırılması gerekir.

Normal bir thread uygulamasında threadlerin birbiri ile çakışmaması için Thread.sleep kullanılarak bir senkron ayarlanır. Ancak bu senkronizasyon programda belli bir yavaşlığa sebep olmaktadır. Bundan dolayı Threadlerin birbiri ile aktif bir şekilde iletişim kurabilmesi için hangi indekslere bakıldığının bilgisini tutan bir Global değişken oluşturulmuştur.

```
public static List<int> indexes = new List<int>();
```

Herhangi bir indeksi, önünde bulunan tüm indekslerle karşılaştırmaya başladığında, thread bu listeye bu indekse baktığını veya şu anda bakmakta olduğunu belirtmek için bu indeksi global değişkene atar, diğer threadler ise yeni bir indekse başlamadan önce bu indeksin listede olup olmadığını kontrol eder böylece başlamayı düşündüğü indeksin hali hazırda işlenmiş ya da işleniyor olup olmadığı durumunu anlar, eğer işleniyorsa bir sonraki indekse geçip aynı aşamaları burada da uygular.

```
if (Globals.indexes.Contains(i))
{
    continue;
}
Globals.indexes.Add(i);
```

Daha sonra threadler karşılaştırmaya kullanıcının vermiş olduğu şartlar doğrusunda devam eder.

Kullanıcının girdiği thread sayısı döngüde kullanılır, bu döngü kullanıcının girdiği sayı kadar döner ve her dönüşünde yeni bir thread başlatır.

```
for (int a = 0; a < Globals.ui_info.Thread_; a++)
{
    Console.WriteLine("thread start");
    workerThreads[a] = new Thread(Program.thread_method);
    workerThreads[a].IsBackground = false;
    workerThreads[a].Start(a);
}
```

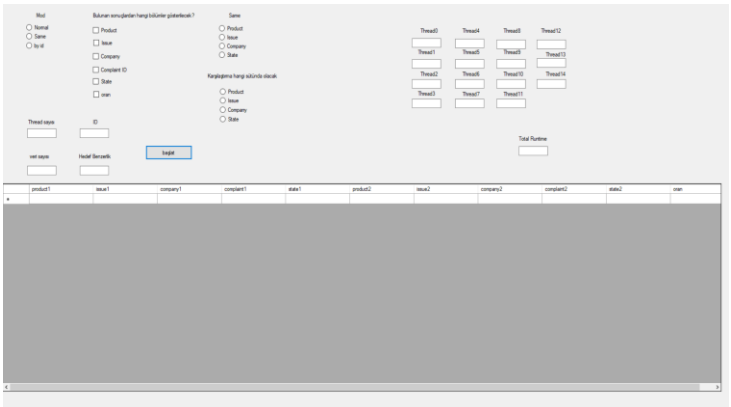
Threadler tarafından bulunan sonuçların indeksi ve benzerlik oranı daha sonra arayüze bastırmak için yine bir global değişkende toplanır, böylece Threadlerin bulduğu sonuçlar ortak bir listeye atılır.

```
if (Globals.ui_info.Comp == 1)
{
    for (int j = i + 1; j < Globals.ui_info.Count_; j++)
    {
        diff = stringDifference(Globals.yazlab_list.ElementAt(i).Product, Globals.yazlab_list.ElementAt(j).Product);
        if (diff > Globals.ui_info.Oran)
        {

            //f.insert_table(row, i, j, diff);
            //row++;
            Globals.result_i.Add(new found_index(i, j, diff));

        }
    }
}
```

Yukarda görüldüğü üzere ilk başta kullanıcının karşılaştırılmasını istediği satır kontrol ediliyor. Daha sonrasında ise yine kullanıcının girdiği veri sayısı kısıtlamasına göre döngüye giriliyor. Eğer benzerlik kullanıcının istediği orandan fazlaysa bulunan sonucun indeksleri bir listeye ekleniyor. Şimdi de kullanıcının girdiği verilerin nasıl sınıflandırıldığı ve kullanıcının arayüz tarafını göreceğiz.



D. Arayüz

Yukardaki örnekte görüldüğü gibi sadece 15 Thread çalışma sonucunun yazılabileceği yer var. Bunun nedeni ise programın yazıldığı bilgisayarın 15 adet işlemciye sahip olması, bundan dolayı maksimum performan 15 thread ile alınabiliyor, 15'in üstünde bir thread kullanımı programı daha da yavaşlatıyor. Runtime ölçümlerine bakalım

```
var timer = System.Diagnostics.Stopwatch.StartNew();
for (int a = 0; a < Globals.ui_info.Thread_; a++)
{
    Console.WriteLine("thread start");
    workerThreads[a] = new Thread(Program.thread_method);
    workerThreads[a].IsBackground = false;
    workerThreads[a].Start(a);
}

for (int a = 0; a < Globals.ui_info.Thread_; a++)
{
    workerThreads[a].Join();
}
timer.Stop();
var runtime = (float)(timer.ElapsedMilliseconds / 1000);
```

Threadler başlatılmadan önce sayaç başlıyor, threadler başladıktan sonra da tüm kodun bekletilmesi için (doğru ölçüm yapabilmek adına) Join terimi kullanılıyor her bir thread için. Threadler çalışmayı durdurunca da sayaç duruyor ve ölçülen zaman ekrana bastırılmak üzere kaydediliyor. Peki her bir threadin runtime nasıl hesaplanıyor?

```
for (int i = (int)threadID; i < Globals.ui_info.Count_; i++)
{
    //Console.WriteLine($"count: {Globals.indexes.Count}");
    if (Globals.indexes.Count() == (Globals.ui_info.Count_ - 1))
    {
        timer.Stop();
        Globals.thr_runtime[(int)threadID] = (float)(timer.ElapsedMilliseconds / 1000);
        Console.WriteLine($"thread_id:{threadID} runtime:{timer.ElapsedMilliseconds / 1000} seconds");
        return;
    }
```

Eğer bütün indeksler hali hazırda gezilmişse program kendisi için sona gelindiğini if bloğu ile anlıyor ve threadin ilk başında başlatılan sayaç durduruluyor. Her bir threadin kendi id'si vardır bu id sırası ile 0-14 arasındadır. Bir global float dizisinin [id] indeksine bu id'ye sahip olan threadin runtime'ı kaydediliyor. Böylece her threadin çalışma süresi hesaplanabilir oluyor.

Kullanıcı checboxlardan istediği hangi veri ya da verilerin gösterileceğini seçiyor, böylece sadece bu seçtiği veriler daha önce toplanan sonuçların indeksleri sayesinde yazdırılabiliyor, her veri yazdırıldığında tabloda yeni bir satır açılarak bir sonraki veri için hazır olunduğuna emin olunuyor.

```
if (show_product.Checked)
{
    dataGridView1.Rows[row].Cells[0].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).I).Product;
    dataGridView1.Rows[row].Cells[5].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).J).Product;
}
if (show_issue.Checked)
{
    dataGridView1.Rows[row].Cells[1].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).I).Issue;
    dataGridView1.Rows[row].Cells[6].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).J).Issue;
}
if (show_company.Checked)
{
    dataGridView1.Rows[row].Cells[2].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).I).Company;
    dataGridView1.Rows[row].Cells[7].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).J).Company;
}
if (show_complaint.Checked)
{
    dataGridView1.Rows[row].Cells[3].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).I).ID;
    dataGridView1.Rows[row].Cells[8].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).J).ID;
}
if (show_state.Checked)
{
    dataGridView1.Rows[row].Cells[4].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).I).State;
    dataGridView1.Rows[row].Cells[9].Value = Globals.yazlab_list.ElementAt(Globals.result_i.ElementAt(a).J).State;
}
if (show_oran.Checked)
{
    dataGridView1.Rows[row].Cells[10].Value = Globals.result_i.ElementAt(a).Diff;
}
```

Yukarda gözlemlenebileceği üzere if blokları sayesinde kullanıcının hangi kutuları işaretlediği anlaşılıyor ve böylece bu verileri tabloya yazdırıyor.

Başlat butonuna basıldığı zaman o anda kutulara yazılmış olan tüm değerler alınır ve thread metodunun içinde kullanılması için bir global değişkene atar, aynı zamanda threadlerin başlamasına da neden olur.

IV. Deneysel Sonuçlar

Proje üstünde çalışma sonucunda yüksek işlemci gücü gerektiren işlemlerin nasıl çözülebileceğini, paralel çalışan bir işlemci grubun senkronizasyon için nasıl yönlendirilebileceği öğrenildi.

Kullanıcı arayüzü tasarımı sayesinde masaüstü uygulamasının çalışma prensipleri öğrenilmiş olundu, C# programlama dili .NET framework ve araçlarının kullanımı öğrenilmiş olundu.

Büyük verilerin küçük zamanlarda işlenerek zaman tasarrufu üzerine genel iş verimliliğin artabileceği ve bunun da maddi ve manevi kazanç sağlayabileceği anlaşılmıştır.

Bilgisayar oyunları gibi ağır işlem yükü gerektiren işlemlerin CPU üzerinde kullanılarak nasıl yapılacağı anlaşıldı ve ilk defa bilgisayarın işlemcisi bir program üzerinde son performansta çalışmaya başladı.

10000 verinin birbiri ile karşılaştırılmasının 1 thread için 58 saniye sürdüğü öğrenilirken 15 thread için bu sürenin 18 saniyeye düştüğü gözlemlenmiştir. Bu da bize multi-thread kullanımının işlemleri neredeyse 3.5 katı hızına çıkardığını göstermektedir.

V. SONUÇ

C# da .csv dosya formatının işlenmesi öğrenildi.

PowerShell kullanarak .csv dosyalarının bir SQL programı gibi işlenebileceği

Thread kullanımı ve thread senkronizasyonu

VI. Yalancı Kod

- Başla
- Csv dosyasını oku
- Okunan verilerin null kontrol et
- Okunan verileri bir listeye at
- Arayüzü çalıştır
- Kullanıcıyı bekle
- Butona basıldığında
- Kullanıcının hangi sütunda arama yapılmasını istediğini bir sayı ile belirt
- Sayıyı bir global değişkene at
- Kullanıcının girdiği veri sınırı, thread sayısı ve hedef oranı integer'a parse edip aynı global değişkende topla
- Kullanıcının belirttiği thread sayısı büyüklüğünde bir thread listesi oluştur
- Runtime sayacını başlat
- Threadleri başlat
- Threadleri bekle
- Thread için:
- Thread sayacını başlat
- Kullanıcının girdiği veri sayısı kadar dönecek dngü başlat, başlangıcını thread id olarak ayarla
- Tüm threadler tarafından işlenen indeks sayısının kullanıcı ile eşleşip eşleşmediğini kontrol et, eşleşiyorsa return ile thredi bitir ve runtime çıkar
- Indeksin herhangi bir başka thread tarafından işlenip işlenmediğini kontrol et, işlendiyse bir sonraki indekse ilerle
- Kullanıcının verdiği veriye göre koşul bloğuna gir ve karşılaştırmak üzere döngü başlat
- Eğer bulunan oran kullanıcının istediği orandan yüksekse verinin indeksini kaydet
- Runtime sayacını durdur ve thread süreleri ile beraber arayüze bastır
- Bulunan verileri arayüzdeki tabloya bastır
- Bitiş

VII. Kaynakça

https://www.yazilimkodlama.com/programlama/c-radiobutton-kullanimi-ve-ornegi/

https://stackoverflow.com/questions/10696941/c-sharp-get-string-from-textbox

https://stackoverflow.com/questions/2397895/how-to-insert-value-into-datagridview-cell

https://www.w3schools.com/cs/cs_arrays.php

https://codepad.co/snippet/looping-through-textboxes-in-c

https://stackoverflow.com/questions/19737436/looping-through-each-row-in-a-datagridview

https://www.geeksforgeeks.org/joining-threads-in-c-sharp/

https://stackoverflow.com/questions/29413797/calling-a-thread-from-another-class-c

https://learn.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.wait?view=net-7.0

https://social.msdn.microsoft.com/Forums/vstudio/en-US/25a094d4-a784-4218-bd41-c2076c2e2293/load-form-in-background-thread?forum=csharpgeneral

https://www.codeproject.com/Articles/3440/NET-multi-threading-and-communication-between-thr