23.04.2020

Paralel Hesaplama

Seçim Sistemi Örneği

Prof.Dr. Mahir DURSUN

Ahmet ALTAY / 208010801

GAZİ ÜNİVERSİTESİ BİLİŞİM ENSTİTÜSÜ
BİLGİSAYAR BİLİMLERİ
YÜKSEK LİSAN ÖĞRENCİSİ

İÇİNDEKİLER

GİRİŞ

TANIM

Seçimlerde sandıkların içlerindeki oyların eş zamanlı olarak sayılması ve genel sayım sonucunun daha kısa sürelerde bulunması gerçek hayatta uygulanan bir paralelleştirme yaklaşımıdır. Bize bu örneği vererek paralel algoritmaları açıklayan Prof.Dr. Mahir DURSUN hocamıza teşekkürü borç bilirim. [1]

AMAÇ

Seçimlerde ki oy sayım işlemlerine benzeyen paralel ve sıralı algoritmalar oluşturularak bu yöntemler arasında ki başarımın ve performansın kıyaslanması.

KISITLAR

Bu dokümanı okuyanların temel düzeyde C# programlama dilini bildiği varsayılır. Doküman içerisinde .net core 3.1 SDK'sı kullanılarak bir komut satırı (Consol) uygulaması yapılacaktır. [2]

SENARYONUN İŞLETİLMESİ

ALGORİTMA

- Oy adında aşağıdaki elemanlara sahip bir nesne olsun
  - Id
  - İl
  - İlçe
  - Mahalle
  - Okul
  - Sandık
  - Parti (“A partisi”, “B partisi”)
- İller, İlçeler, Mahalleler, Okullar, Sandıklar şeklinde 5 adet içerisinde rastgele verilerin olduğu diziler tanımlayalım.
- Bu dizileri kullanılarak dizi sayısı + her sandıkta olacak oy sayısı kadar oylar adında bir dizi oluşturup içerisine rastgele bu dizilerdeki verileri dolduracağız.
- Yukarıdaki işlemi 3 ayrı algoritma kullanarak kodlayacağız.
  - Sıralı birbirini bekleyen işler şeklinde
  - İlleri görevlere (task) bölüp her görev altında sıralı olarak döngüler kurup oy dizisini il sayısı kadar görevde eş zamanlı doldurarak.
  - Tüm döngüleri Parallel kütüphanesinin For metodu ile döngüleri ile kurup işlemlerin tamamının paralel yapılmasını sağlayarak
- Rast gele veriler 3 ayrı yöntem ile oluşturulup her bir yöntem için geçen süre hesaplanıp ekrana geçen süre ile birlikte oluşturulan kayıt sayısı yazılarak bu 3 algoritma arasındaki en verimli olanın gözlenmesi.

ALGORİTMANIN ÖZET KODU

YÖNTEM 1

Bu yöntem ile kodlar sıralı algoritmada yazılmıştır. Herhangi bir task ya da paralel çağırımı yapılmamıştır. [3]

```
private static List<Oy> GenerateRandomData()
{
    var arr = new List<Oy>();
    for (int i = 0; i < CIller.Length; i++)
    {
        for (int j = 0; j < CIlceler.Length; j++)
        {
            for (int k = 0; k < CMahalleler.Length; k++)
            {
                for (int l = 0; l < COkullar.Length; l++)
                {
                    for (int m = 0; m < CSandiklar.Length; m++)
                    {
                        for (int n = 0; n < _count; n++)
                        {
                            var oy = new Oy()
                            {
                                Id = Guid.NewGuid(),
                                Il = CIller[i],
                                Ilce = CIlceler[j],
                                Mahalle = CMahalleler[k],
                                Okul = COkullar[l],
                                SandikNo = CSandiklar[m],
                                Parti = GenerateRandomPartyName()
                            };
                            Thread.Sleep(_delayMs);
                            arr.Add(oy);
                        }
                    }
                }
            }
        }
    }

    return arr;
}
```

YÖNTEM 2

Bu yöntemde iller task kullanılarak paralel olarak sayılmıştır. Yani her il aynı anda kendi altındaki sandıkları sıralı algoritma ile tek tek saymaktadır. Bu yöntemde sadece il seviyesinde paralelleştirme uygulanmıştır. [3, 4, 5]

```
private static ConcurrentBag<Oy> GenerateRandomDataUsingTasks()
{
    var arr = new ConcurrentBag<Oy>();
    var tasks = new List<Task>();
    for (int i = 0; i < CIller.Length; i++)
    {
        var ilAdi = CIller[i];
        var task = Task.Run(() =>
        {
            for (int j = 0; j < CIlceler.Length; j++)
            {
                for (int k = 0; k < CMahalleler.Length; k++)
                {
                    for (int l = 0; l < COkullar.Length; l++)
                    {
                        for (int m = 0; m < CSandiklar.Length; m++)
                        {
                            for (int n = 0; n < _count; n++)
                            {
                                var oy = new Oy()
                                {
                                    Id = Guid.NewGuid(),
                                    Il = ilAdi,
                                    Ilce = CIlceler[j],
                                    Mahalle = CMahalleler[k],
                                    Okul = COkullar[l],
                                    SandikNo = CSandiklar[m],
                                    Parti = GenerateRandomPartyName()
                                };
                                Thread.Sleep(_delayMs);
                                arr.Add(oy);
                            }
                        }
                    }
                }
            }
        });
        tasks.Add(task);
    }
    Task.WaitAll(tasks.ToArray());
    return arr;
}
```

YÖNTEM 3

Bu yöntemde tüm sandıklar paralel olarak sayılmıştır. Aynı anda her sandık bir thread ile okunarak tüm sandıkların eş zamanlı sayımı yaptırılmıştır. Bu yöntemde çok fazla kaynak kullanımı olacağı için eğer yeteri kadar işlemci veya işlemci çekirdeği yok ise yani kaynak yetersiz ise sıralı algoritmadan çok fazla farkı olmayacaktır. Tüm kaynakları kullanan en çok kaynak tüketen yöntem budur. Ayrıca bu yöntemde Paralel kütüphanesi kullanılarak Task ile yapılabilen işlemin daha kısa bir yol ile nasıl kodlanacağı da gösterilmiştir. [3, 5, 6]

```
private static ConcurrentBag<Oy> GenerateRandomDataUsingParallelLoop()
{
    var arr = new ConcurrentBag<Oy>();
    Parallel.For(0, CIller.Length, i =>
    {
        Parallel.For(0, CIlceler.Length, j =>
        {
            Parallel.For(0, CMahalleler.Length, k =>
            {
                Parallel.For(0, COkullar.Length, l =>
                {
                    Parallel.For(0, CSandiklar.Length, m =>
                    {
                        Parallel.For(0, _count, n =>
                        {
                            var oy = new Oy()
                            {
                                Id = Guid.NewGuid(),
                                Il = CIller[i],
                                Ilce = CIlceler[j],
                                Mahalle = CMahalleler[k],
                                Okul = COkullar[l],
                                SandikNo = CSandiklar[m],
                                Parti = GenerateRandomPartyName()
                            };
                            Thread.Sleep(_delayMs);
                            arr.Add(oy);
                        });
                    });
                });
            });
        });
    });
    return arr;
}
```

KAYNAK KOD

Bu deneyin kaynak kodları kişisel GitHub hesabımda Public ve MIT lisansı ile saklanmaktadır. Repoya aşağıdaki bağlantı adresinden ulaşabilirsiniz. [3]

Paralel Algoritmalar kök deposu

https://github.com/ahmetaltay33/parallel-algorithms

Bu deneyin deposu

https://github.com/ahmetaltay33/parallel-algorithms/tree/master/dotnetcore/election-sample

KAYNAK KODUN DERLENMESİ

Bilgisayarınızda .net core 3.1 SDK [2] sının yüklü olması gerekmektedir. Bilgisayarınızda Visual Studio 2019 un güncel versiyonu bulunuyor ve yükleme sırasında .net core geliştirme ortamını işaretledi iseniz SDK hali hazırda yüklü demektir.

SDK yüklü olup olmadığını veya yüklü olan sürümü bulmak için komut satırına aşağıdaki kodu yazıp test edebilirsiniz.

dotnet --version

.Net Core SDK 3.1 [2] yüklemek için kaynakçada bulunan bağlantıdan bilgisayarınıza uygun sürümü indirip yükleyebilirsiniz.

Kaynak kodları bilgisayarınıza indirip Visual Studio içerisinde F5 komutu ile çalıştırabilirsiniz.

Geliştirme ortamı kullanmadan komut satırından aşağıdaki komutlar ile kaynak kodu derleyebilirsin.

dotnet build D:\election-sample\election-sample.sln

Derlenmiş uygulama proje klasörü içerisinde **.\bin\debug\netcoreapp3.1 \election-sample.exe** konumunda olacaktır.

Derleme ile ilgili problem yaşanır ise kaynak kodların derlenmiş hali GitHub da aşağıdaki url de bulunmaktadır.

https://github.com/ahmetaltay33/parallel-algorithms/blob/master/dotnetcore/election-sample/output/election-sample.zip

GÖZLEM ÇIKTILARININ ELDE EDİLMESİ

Program kaynak kodları derlendikten sonra election_sample.exe adında bir konsol uygulaması elde edilir. Program dışarıdan 2 adet parametre almaktadır.

İlk parametre: her sandık için oluşturulacak oy sayısıdır.

İkinci parametre: her oy oluşturma işlemi için geçecek olan işlem süresidir. Gerçek ortamda bu süre veri tabanına yazma ya da bir web servise veri gönderme işleminde geçecek zaman olarak değerlendirilir.

Program aşağıda verilen parametreler ile çalıştırılmış ve sonuçları ekran görüntüsü alınarak eklenmiştir.

Çalıştırılan Komut: election-sample.exe 5 3

```
PS D:\GitHub\ahmetaltay33\parallel-algorithms\dotnetcore\election-sample> .\bin\Release\netcoreapp3.1\election-sample.exe 5 1

Veriler sıralı algoritma ile oluşturuluyor... (Paralel yok, sıralı)
Veriler oluşturuldu. Kayıt sayısı: 39375 Geçen süre ms: 519710

Task kullanılarak sadece illerin paralelleştirildiği yöntem ile veriler oluşturuluyor... (Sadece iller paralel)
Veriler oluşturuldu. Kayıt sayısı: 39375 Geçen süre ms: 87995

Tüm döngüler Parallel.For ile çalıştırılarak veriler oluşturuluyor... (Tüm döngüler paralel)
Veriler oluşturuldu. Kayıt sayısı: 39375 Geçen süre ms: 49110
```

Çalıştırılan Komut: election-sample.exe 100 0

```
PS D:\GitHub\ahmetaltay33\parallel-algorithms\dotnetcore\election-sample> .\bin\Release\netcoreapp3.1\election-sample.exe 100 0

Veriler sıralı algoritma ile oluşturuluyor... (Paralel yok, sıralı)
Veriler oluşturuldu. Kayıt sayısı: 787500 Geçen süre ms: 2909

Task kullanılarak sadece illerin paralelleştirildiği yöntem ile veriler oluşturuluyor... (Sadece iller paralel)
Veriler oluşturuldu. Kayıt sayısı: 787500 Geçen süre ms: 1270

Tüm döngüler Parallel.For ile çalıştırılarak veriler oluşturuluyor... (Tüm döngüler paralel)
Veriler oluşturuldu. Kayıt sayısı: 787500 Geçen süre ms: 1531
```

Çalıştırılan Komut: election-sample.exe 1000 0

```
PS D:\GitHub\ahmetaltay33\parallel-algorithms\dotnetcore\election-sample> .\bin\Release\netcoreapp3.1\election-sample.exe 1000 0

Veriler sıralı algoritma ile oluşturuluyor... (Paralel yok, sıralı)
Veriler oluşturuldu. Kayıt sayısı: 7875000 Geçen süre ms: 31690

Task kullanılarak sadece illerin paralelleştirildiği yöntem ile veriler oluşturuluyor... (Sadece iller paralel)
Veriler oluşturuldu. Kayıt sayısı: 7875000 Geçen süre ms: 13205

Tüm döngüler Parallel.For ile çalıştırılarak veriler oluşturuluyor... (Tüm döngüler paralel)
Veriler oluşturuldu. Kayıt sayısı: 7875000 Geçen süre ms: 17379
```

Çalıştırılan Komut: election-sample.exe 1 10

```
PS D:\GitHub\ahmetaltay33\parallel-algorithms\dotnetcore\election-sample> .\bin\Release\netcoreapp3.1\election-sample.exe 1 10

Veriler sıralı algoritma ile oluşturuluyor... (Paralel yok, sıralı)
Veriler oluşturuldu. Kayıt sayısı: 7875 Geçen süre ms: 115489

Task kullanılarak sadece illerin paralelleştirildiği yöntem ile veriler oluşturuluyor... (Sadece iller paralel)
Veriler oluşturuldu. Kayıt sayısı: 7875 Geçen süre ms: 17588

Tüm döngüler Parallel.For ile çalıştırılarak veriler oluşturuluyor... (Tüm döngüler paralel)
Veriler oluşturuldu. Kayıt sayısı: 7875 Geçen süre ms: 12519
```

GÖZLEM SONUÇLARI

Gözlem çıktıları incelendiğinde her bir işlem için geçen zamanın artırılması paralel olan işlemlerin daha kısa sürede işi bitirmesine sebep oluyor. İşlem sayısının artması bir işlem için geçen sürenin sıfır olması ya da sıfıra yaklaşması durumlarında paralel işlemlerin çok fazla etkisinin olmadığı gözlemleniyor.

Sonuç olarak döngüsel bir paralel hesaplamada toplam geçen zamana olan etki; döngünün eleman sayısından çok, döngünün her bir turda harcadığı zamana bağlıdır.

GÖZLEM ORTAMI

Bu dokümanda bahsi geçen algoritmanın gözlemi aşağıdaki donanım özelliklerine sahip bir ortamda yapılmıştır.

- Intel Core i7 1. Nesil 1.7 Ghz Turbo ile max 2.4 Ghz 4 çekirdekli 8 kanallı işlemci (740QM)
- DDR3 1333 Mhz Clock 9 16 GB Ram
- SSD disk
- Paylaşımsız ekran kartı
- Diz üstü bilgisayar

KAYNAKÇA

[1] P. D. M. DURSUN, *Paralel Hesaplama,* Ankara: Gazi Üniversitesi Bilişim Enstitüsü, 2020.

[2] MSDN, «.Net Core SDK 3.1,» Microsoft, [Çevrimiçi]. Available: https://docs.microsoft.com/tr-tr/dotnet/?view=netcore-3.1. [Erişildi: 23 04 2020].

[3] A. ALTAY, «Election Sample,» 12 04 2020. [Çevrimiçi]. Available: https://github.com/ahmetaltay33/parallel-algorithms/tree/master/dotnetcore/election-sample. [Erişildi: 23 04 2020].

[4] MSDN, «Task.Run Yöntem,» [Çevrimiçi]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.run?view=netcore-3.1. [Erişildi: 23 04 2020].

[5] MSDN, «ConcurrentBag<T> Class,» [Çevrimiçi]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentbag-1?view=netcore-3.1. [Erişildi: 23 04 2020].

[6] MSDN, «Parallel.For Method,» [Çevrimiçi]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.parallel.for?view=netcore-3.1. [Erişildi: 23 04 2020].