SUNUCU İSTEK YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ

Atakan İBİŞ, Hanım Sude ŞANLI

170202047, 160202014
Bilgisayar Mühendisliği

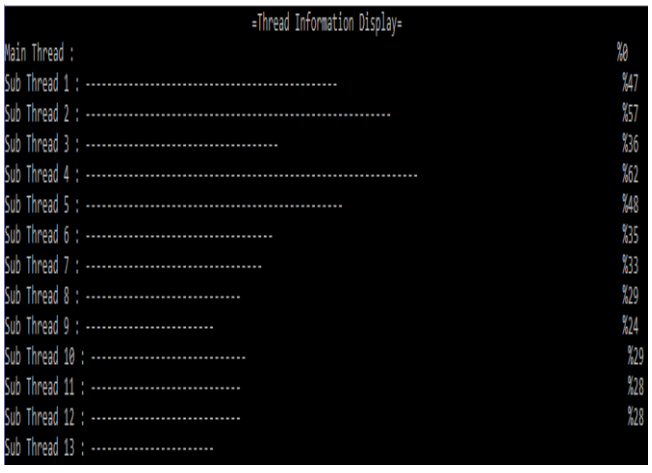
Kocaeli Üniversitesi
atakanibis1@gmail.com, sanlihanimsude@gmail.com

Özet

Projenin amacı, bir sunucuya gelen isteklerdeki aşırı yoğunluğu, multithread kullanarak alt sunucularla birlikte azaltmaktır.

1. Giriş

Program başlatıldığında konsolda Thread İnformation Display üzerinde Main Thread ve Sub Threads bilgilerini görebilmekteyiz. Kapasitelerinin doluluk oranları, % kaç oranda olduğunu ve toplamda kaç thread olduğunu buradan kontrol edebilmekteyiz.



2. Temel Bilgiler

Program, C# programlama dilinde yazılmış olup, geliştirme ortamı olarak Visual Studio kullanılmıştır.

3. Tasarım

Sunucu İstek Yoğunluğunun Multithread İle Kontrolü programının programlanma aşamaları aşağıdaki başlıklar altında açıklanmıştır.

3.1 Yazılım Mimarisi ve Algoritma

Main thread ve sub threadler için class oluşturulmuştur.
Main thread sınıfında Threading sınıfı eklenmiştir. Buradaki threadlerin özellikleri olarak istek kapasitesi olarak 10000, request time 500 ve response time 200 olarak belirlenmiştir. İlgili fonksiyonların çalışması sonucunda oluşan sub threadler 'SubThreads' isimli bir listede tutulmaktadır. Main Thread'den oluşturulacak olan nesne için bir tane response thread ve bir tane de request thread oluşturulup, başlatılmaktadır. Başlatılan threadler ResponseFunction ve RequestFunction ile yapılacak işlemler sonucunda [1,100] arasında istek kabul etmektedir ve [1,50] arasında da rastgele sayıda isteğe geri dönüş yapmaktadır.
İlk başta belirtildiği üzere 2 adet sub thread olmalıdır. Bunlar SubThread sınıfımızdan oluşturulup sub thread listesine eklenmektedir.
SubRequestThread adlı thread oluşturulup SubRequestFunction ile SubThread listesindeki threadler için [1,50] arasında rastgele isteğe geri dönüş yapmaktadır.
SubThreadCreator thread ise SubThreadControl fonksiyonu ile birlikte çalışmaktadır. Bu fonksiyon ile birlikte mevcut olan alt sunucuları kontrol edilmektedir. Eğer herhangi bir alt sunucunun kapasitesi %70 ve üzerinde ise yeni bir alt sunucu oluşturulmaktadır ve kapasitenin yarısını yeni oluşturduğu alt sunucuya göndermektedir. Eğer

herhangi bir alt sunucunun kapasitesi %0 a ulaşır ise mevcut olan alt sunucu sistemden kaldırılmaktadır. Bunların yanı sıra her şekilde en az iki alt sunucu çalışır durumda kalmaktadır. Sunucuyu takip edebilmek amacıyla PrinterThread oluşturulmuştur. Interface fonksiyonu ile mevcut olan tüm threadleri, kapasitelerini, % oranlarını canlı olarak görebilmekteyiz.
SubThread sınıfında ise max_capacity olarak 5000, request time ise 500 olarak belirlenmiştir. Response time ise 300 ms olarak oluşturulmuştur. Başlatılan ResponseThread, ReponseFunction ile [1,50] arasında rastgele sayıda geri dönüş yapmaktadır.

3.2 Kullanılan Bazı Fonksiyonlar

1-50 arasında geri dönüş yapımasını saplayan ResponseFunction:

```
public void ResponseFunction()
{
    while (true)
    {
        Random rand = new Random();
        capacity -= rand.Next(1, 50);
        if (capacity < 0)
            capacity = 0;
        Thread.Sleep(response_time);
    }
```

Aynı şekilde RequestFunction ile [1,100] arasında istek alınmaktadır:

```
public void RequestFunction()
{
    while (true)
    {
        Random rand = new Random();
        capacity += rand.Next(1, 100);
        if (capacity >= max_capacity)
            capacity = max_capacity;
        Thread.Sleep(request_time);
    }
```

SubThreadControl fonsiyonu ile mevcut olan alt sunucuları kontrol ediliyor. Eğer herhangi bir alt sunucunun kapasitesi %70 ve üzerinde ise yeni bir alt sunucu oluşturup, kapasitenin yarısını yeni oluşturduğu alt sunucuya gönderir. Eğer herhangi bir alt sunucunun kapasitesi %0 a ulaşır ise mevcut olan alt sunucu sistemden kaldırılır. En az iki alt sunucu sürekli çalışır durumda kalmasını da sağlamaktadır.

```
public void SubThreadControl()
{
    while (true)
    {
        for (int i = 0; i < SubThreads.Count; i++)
        {
            if (SubThreads[i].Capacity == 0 && SubThreads.Count > 2)
            {
                SubThreads.Remove(SubThreads[i]);
            }
            else if (SubThreads[i].Capacity >= SubThreads[i].Max_capacity * 7 / 10)
            {
                SubThreads[i].Capacity /= 2;
                SubThread subthread3 = new SubThread
                {
                    Capacity = SubThreads[i].Capacity
                };
                SubThreads.Add(subthread3);
```

4. Genel Yapı

Projemiz genel olarak bizden istenilen tüm isterleri gerçekleştirebilmektedir. Sunucudaki istek yoğunluğunu multithread yöntemiyle kontrol edebilmekteyiz.

5. Kaynaklar

[1] https://www.tutorialspoint.com/csharp/
Kazanımlar: C#
[2] https://stackoverflow.com/
[3] https://docs.microsoft.com/tr-tr/dotnet/api/system.threading.thread?view=netframework-4.8
Kazanımlar: Threading
[4] https://www.tutorialspoint.com/csharp/csharp_multithreading.htm
[5] https://berkarat.com/c-thread-ve-multithread-islemleri/
[6] https://www.javatpoint.com/c-sharp-multithreading