T.C.

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

Paralel ve Dağıtık Sistem Programlama

Tek Boyutlu Bir Dizide ki En Büyük Elemanın MPI Kütüphanesi Kullanılarak Bulunması

Oğulcan Uçarsu

185172014

ÖĞRETİM GÖREVLİSİ

DOÇ. DR. Hikmet Hakan GÜREL

Ocak 2019

1.Problem Tanımı

MPI kütüphanesi kullanılarak, tek boyutlu bir dizide ki, dizinin en büyük elemanı nasıl bulunur? Dizi elemanları 'random' bir şekilde oluşturulduktan sonra, işlemci(core) sayısına göre diziyi parçalara bölerek, bölünen diziler arasında her bir işlemcinin(core) 'local' en büyük değeri bulunur. Her bir işlemcinin bulduğu 'local' en büyük değerlerden sonra 'master' işlemci üzerinde 'local' en büyük değerler arasında karşılaştırma yapılarak dizinin en büyük değeri bulunur.

2.Yöntem

MPI kütüphanesi ile Paralel Programlama yapabilmek için, Windows bir bilgisayarda 'Oracle VM Virtual Box Programı' kullanılarak Linux işletim sistemi oluşturulmuştur. Oluşturulan Ubuntu Linux Bilgisayar'ın özellikleri şu şekildedir.

- İşletim Sistemi: Ubuntu 64 bit
- Bellek: 8 GB

Ubuntu sanal bilgisayarı kurduktan sonra, tek boyutlu dizinin en büyük elemanını bulmak için, 'GCC' derleyicisinin ve 'Open MPI' kütüphanesinin indirilmesi gerekmektedir. Sırası ile terminal ekranına,

- 'sudo apt install gcc' komutu ile GCC derleyicisi indirilir.
- 'sudo apt install mpich' komutu ile Open MPI kütüphanesi ile indilir.

Yukarıda anlatılan kurulumlar, ilgili dersin Vize sınavı kapsamında yapılan proje ödevinde kurulmuştur.

2.1 Dizinin en büyük elemanını bulan algoritma ve çekirdekler arasında ki yük dağılımı

İlgili işlem yapılırken C programlama dili kullanılarak yapılmıştır. İlk önce tek boyutlu dizi oluşturularak elemanları random bir şekilde atanır. Daha sonra çekirdek sayısına göre orantılı bir şekilde dizi parçalanarak, parçalanmış dizilerin en büyük elemanları bulunur.

```
MPI_Comm_size(MPI_COMM_WORLD, &num_proc);
quotient = array_len / num_proc;
rem = array_len % num_proc;
```

Şekil – 1 Dizi boyutuna göre dizi elemanlarını işlemcilere orantılı şekilde dağıtmak için bölüm ve kalan hesaplanır.

```
for (i=0;i<array_len;++i){
search_array[i] = rand();
```

Şekil – 2 Random bir şekilde dizinin elemanları oluşturulur.

Eğer dizi boyutu ile işlemcilere dağıtılırken, dizi boyutu işlemci sayısı tam bölünmez ise, tam bölünmeyen dizinin elemanları ve sonrasında tam bölünecek şekilde diziler parçalanarak, parçalanmış diziler işlemcilere gönderilir[1, 2, 3].

```
for (rank=1; rank < rem; ++rank){
    sub_len = quotient+1;
    sub_start = rank*quotient+rank;
    MPI_Send(&sub_len,1,MPI_INT,rank,0,MPI_COMM_WORLD);
    MPI_Send(&(search_array[sub_start]),sub_len, MPI_INT, rank, 0, MPI_COMM_WORLD);
}
for (rank=1; rank < num_proc; ++rank){
    sub_len = quotient;
    sub_start = rank*quotient+rem;
    MPI_Send(&sub_len,1,MPI_INT,rank,0,MPI_COMM_WORLD);
    MPI_Send(&(search_array[rank*quotient+rem]),quotient, MPI_INT, rank, 0, MPI_COMM_WORLD);
```

Şekil – 3 İşlemci sayısına göre parçalanmış dizilerin işlemcilere gönderilmesi

Parçalanmış şekilde işlemcilere bölünmüş dizilere, kendi içlerinde ki en büyük elemanları bulmak için 'find_max' metodunu çağırırlar. Bu metot parametre olarak parçalanmış diziyi ve o dizinin boyutunu alır. Parçalanmış dizide ki en büyük elemanı bulmak için, yapılan işlemler ise şu şekildedir. İlk önce dizinin ilk elemanı en büyük değer(max) olarak belirlenir. Daha sonra 'for' döngüsü ile dizide ki elemanları sırası ile ilgili 'max' elemandan büyük mü değil mi kontrolü sağlanır. Eğer ilgili indeks elemanı 'max' değerinden büyük ise o indekste ki eleman yeni 'max' değeri olarak atanır[4].

```
int find_max(int local_array[ ],int length)
{
    int i;
    int max;
    max = local_array[0];
    for (i=1;i<length;++i)
    if (local_array[i] > max){
        max = local_array[i];
    }
    return max;
}
```

Şekil – 4 Parçalanmış dizide ki en büyük elemanın bulunması

Bu işlemlerin en sonunda ise MPI kütüphanesi yaratmış olduğu çekirdeklerde ki dizinin en büyük elemanlarını alarak 'local max' değerler arasından dizinin en büyük elemanı bulunur.

```
for (rank=1;rank<num_proc;++rank){
    MPI_Recv(&local_max,1,MPI_INT,MPI_ANY_SOURCE,0,
                    MPI_COMM_WORLD, &status);
    if (local_max > global_max)
        global_max = local_max;
}
```

Şekil – 5 MPI kütüphanesinin ilgili çekirdeklerden 'local max' değerleri alarak dizide ki en büyük elemanı bulur.

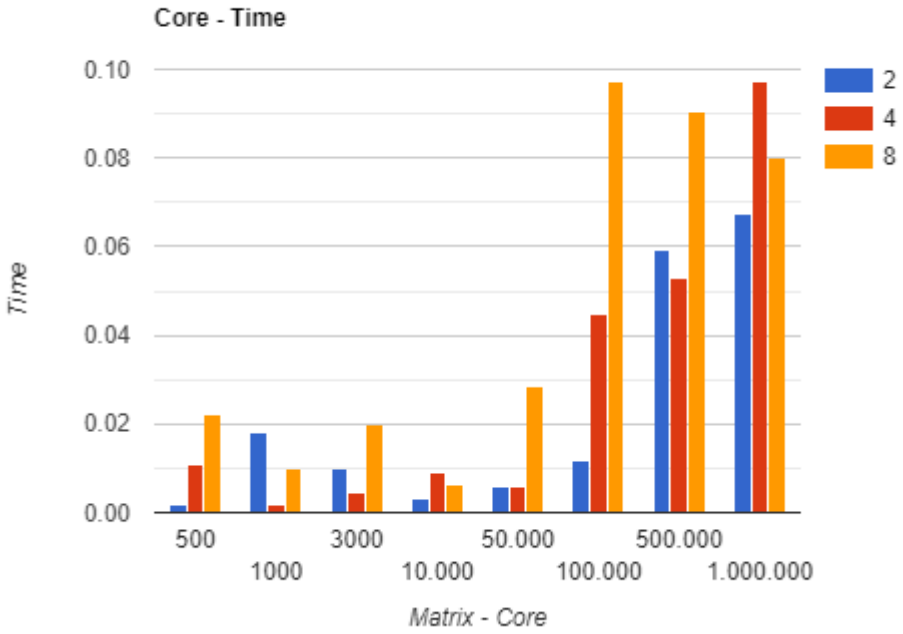
2.2 Dizi büyüklükleri ve çekirdek sayılarına göre işlem sürelerinin ölçülmesi

Dizi boyutları belirlenirken Bilgisayar belleğinin oluşturabileceği en büyük matris boyutuna kadar diziler oluşturulmuştur. İşlemler 2, 4 ve 8 çekirdek üzerinde denemiştir.

İşlem yapılan matris boyutları;

- 500
- 1000
- 3000
- 10.000
- 50.000
- 100.000
- 500.000
- 1.000.000
- 2.000.000 (Matris oluşamadı. Bellek hatası verdi.)
- 5.000.000 (Matris oluşamadı. Bellek hatası verdi.)

Yapılan işlemler ve elde edilen sonuçların birleştirilmesi ile birlikte aşağıda ki grafik elde edilmiştir.



Grafik – 1 Matris boyutu ve çekirdek sayısına göre elde edilen sonuçlar

Küçük denilebilecek matris boyutlarında işlem süreleri birbirine yakın çıksa bile matris boyutu büyüdükçe işlem süresi 'exponensiyel' artıyor denilebilir. Küçük matrislerde çekirdek sayısı değiştiğinde matris boyutunun küçük olması sebebi ile işlem süresinde çok büyük bir değişiklik olmamaktadır. Fakat matris boyutu büyüdüğünde ve çekirdek sayısı arttığında işlem süresinin arttığı söylenebilir. Ayrıca işlem süresi hesaplanırken MPI ile paralel programlama yapılmaya başlandığı an ve bittiği an arasında ki zaman üzerinden işlem süresi hesaplanmıştır.

2.3 Sonuçlar

MPI kütüphanesi ile Paralel programlama yapılarak, tek boyutlu bir dizinin en büyük elemanı bulunmaya çalışılmıştır. Matris boyutu belirlenirken Bilgisayar belleğinde oluşturulabilecek en büyük boyuta kadar matrisler oluşturulmaya çalışılmıştır. İlgili matris oluşturulduktan sonra, çekirdek sayısına göre matris bölünerek local matrisler elde edilmiştir. Çekirdek sayısına göre oluşan local matrisler, MPI kütüphanesi ile çekirdekler aracılığı ile en büyük elemanı bulan fonksiyona giderek, her bir local matris kendi en büyük elemanını bulduktan sonra master çekirdek, local matrislerde ki en büyük değerler arasından, dizinin en büyük elemanını bulur ve işlem tamamlanır. Performans iyileştirme anlamında, local matrislerde ki en büyük elemanı bulan fonksiyon içerisinde 'Big(o)' notasyonu düşünülerek, algoritma üzerinde değişiklikler yapılarak daha performanslı bir algoritma geliştirilebilir.

Referanslar

1. https://www.open-mpi.org/faq/?category=mpi-apps

2. http://www.csharpnedir.com/articles/read/?id=473

3. http://www.csharpnedir.com/articles/read/?id=481

4. https://www.quora.com/What-is-the-best-algorithm-for-finding-the-max-in-an-array-and-what-is-its-complexity