Suleyman Demirel Universitesi
Muhendislik Fakultesi
Bilgisayar Muhendisligi

Paralel Programlama Dersi

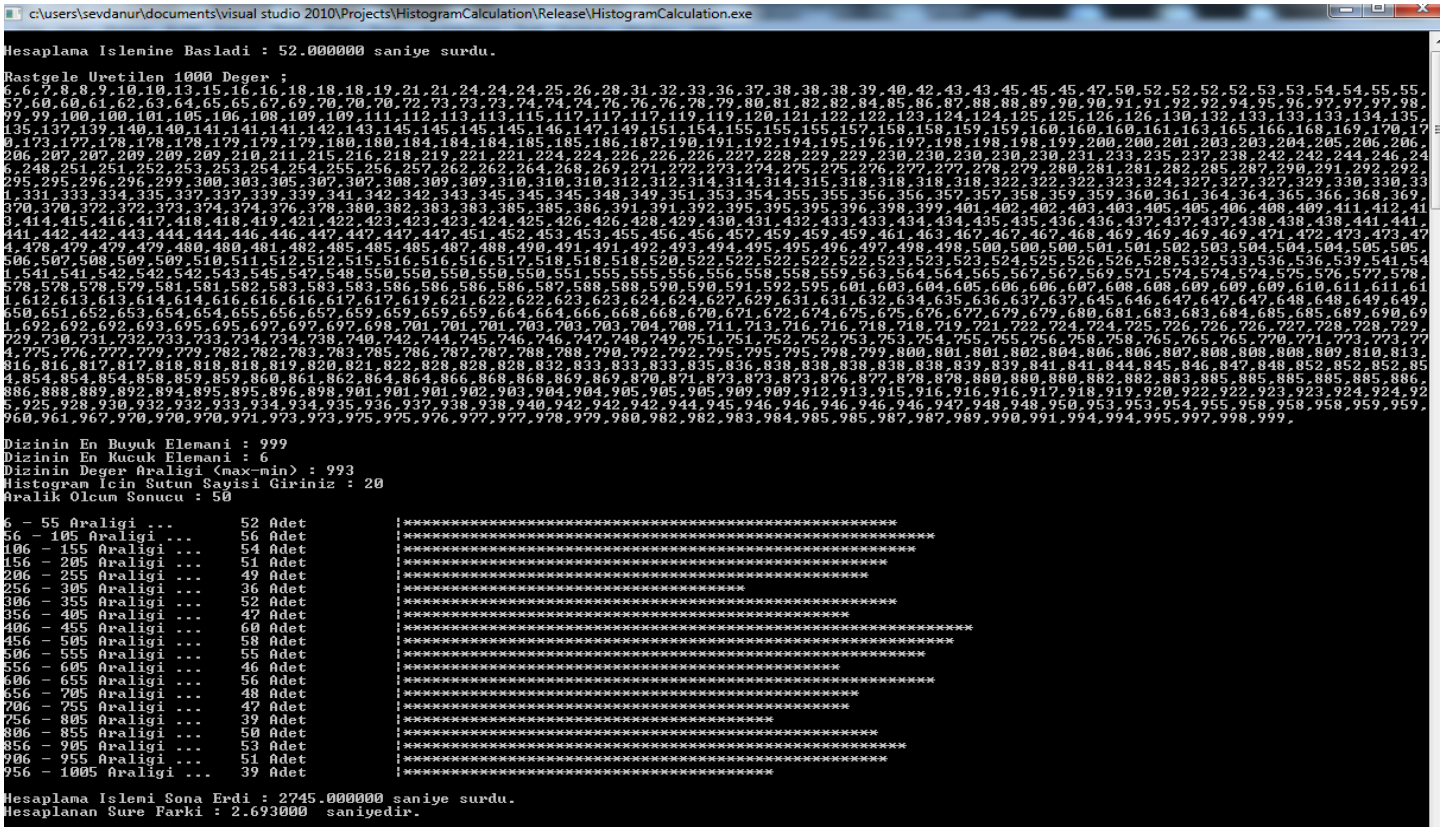
Sevdanur GENC - 0921012067

Paralel Programlama

Histogram Hesaplama - Seri ve Paralel Programlama Ile Analiz Sonuclari

Seri Programlama Kod Ve Analizleri

Seri Programlama kodlari ve ornek ekran ciktisi su sekildedir.



Dizi boyutu 100 olarak verilmistir, Sutun sayilari ise en fazla 20 olarak tanimlanmistir.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <float.h>
#include <conio.h>
#include <string.h>
#include <iostream>
#include <math.h>

using namespace std;

int main(int argc, char *argv[])
{
	int sayac = 0, i, sutunSayisi = 0, yenidizi, min, max, degerAraligi, aralikOlcumu,
baslangic[21], bitis[21];
```

```
int dizi[1000];

clock_t HesaplamayaBasla, HesaplamayiBitir;
double SureFarki;

srand(time(NULL));

printf("\nHesaplama Islemine Basladi : ");
HesaplamayaBasla = clock();
printf("%f saniye surdu. \n", (double) HesaplamayaBasla );

printf("\n");
for(i=0;i<1000;i++)
{
	dizi[i]= rand()%1000;
}

min = dizi[0];
max = dizi[0];

for(i=1;i<1000;i++)
{
	if(dizi[i]<min) min = dizi[i];
	if(dizi[i]>max)	max = dizi[i];
}

for(i=1; i<1000; i++)
	for(int j=i; j<1000;j++)
	{
		if(dizi[i]>dizi[j])
		{
			yenidizi = dizi[i];
			dizi[i]=dizi[j];
			dizi[j]=yenidizi;
		}
	}

cout << "Rastgele Uretilen 1000 Deger ; " << endl;
for(i = 1; i<1000; i++)
	cout << dizi[i] << ",";

printf("\n");
cout << "\nDizinin En Buyuk Elemani : " << max << endl;
cout << "Dizinin En Kucuk Elemani : " << min << endl;

degerAraligi = max - min;
cout << "Dizinin Deger Araligi (max-min) : " << degerAraligi << endl;

cout << "Histogram Icin Sutun Sayisi Giriniz : ";
cin >> sutunSayisi;

aralikOlcumu = ((float) degerAraligi) / ((float) sutunSayisi) + 0.5f;
cout << "Aralik Olcum Sonucu : " << aralikOlcumu << endl;

baslangic[0] = min;

for(i = 0; i<sutunSayisi; i++)
{
```

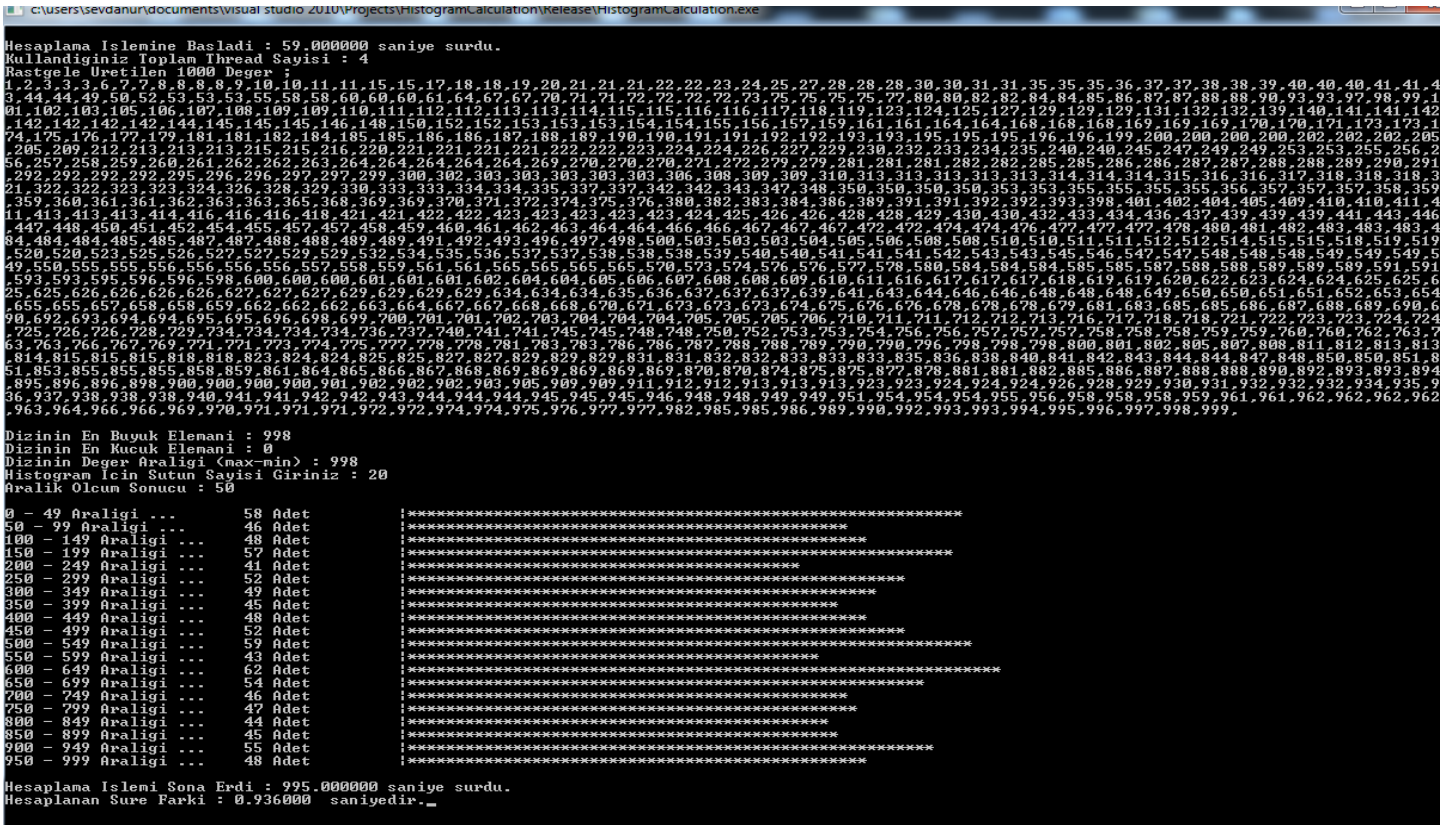
```
		baslangic[i+1] = (baslangic[i] + aralikOlcumu);
		bitis[i] = (baslangic[i+1] - 1);
		bitis[i+1] = (bitis[i] + aralikOlcumu);
	}

	for(int j = 0; j<sutunSayisi; j++)
	{
		sayac = 0;
		for(i = 1; i<1000; i++)
		{
			if ((dizi[i] >= baslangic[j]) && (dizi[i] <= bitis[j]))
				sayac++;
		}
			printf("\n");
			printf("%d - %d Araligi ... \t %d Adet \t |", baslangic[j],
bitis[j], sayac);
			for(int k=0; k<sayac; k++)
					printf("*");
	}

	printf("\n");
	printf ("\nHesaplama Islemi Sona Erdi : ");
	HesaplamayiBitir = clock();
	printf("%f saniye surdu. \n",(double) HesaplamayiBitir );
	SureFarki = ((double) (HesaplamayiBitir - HesaplamayaBasla)) / CLOCKS_PER_SEC ;
	printf("Hesaplanan Sure Farki : %f  saniyedir.", SureFarki);

	getche();
	return 0;

}
```

SERİ PROGRAMLAMA ANALİZ SONUCU;

Seri bir sekilde kodlanmis olan histogram hesaplama uygulamasinda tek thread'in baslangic ve bitis surelerini birer degisken de tutttuktan sonra farklarini alarak calisma surelerini belirlemis oldum. Ornek ekran ciktisinda yapilan islemlerin suresi 2.693 saniye surmustur.

Paralel Programlama Kod Ve Analizleri

Paralel Programlama kodlari ve ornek ekran ciktisi su sekildedir.




```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <float.h>
#include <conio.h>
#include <string.h>
#include <iostream>
#include <math.h>
#include <omp.h>

using namespace std;

int main(int argc, char *argv[])
{
	int sayac = 0, i, j, sutunSayisi = 0, temp, min, max, degerAraligi, aralikOlcumu;
	int dizi[1000],baslangic[21],bitis[21];
	int ToplamThreadSayisi, ThreadID, chunk = 10;

	clock_t HesaplamayaBasla, HesaplamayiBitir;
	double SureFarki;

	srand(time(NULL));
```

```
	printf("\nHesaplama Islemine Basladi : ");
	HesaplamayaBasla = clock();
	printf("%f saniye surdu. \n", (double) HesaplamayaBasla );

	#pragma omp parallel shared(dizi,baslangic,bitis,ToplamThreadSayisi,chunk)
private(ThreadID,i,j,temp,sutunSayisi,sayac,min,max,degerAraligi,aralikOlcumu)
	{
		ThreadID = omp_get_thread_num();

		if (ThreadID == 0)
		{
			ToplamThreadSayisi = omp_get_num_threads();
			printf("Kullandiginiz Toplam Thread Sayisi : %d
\n",ToplamThreadSayisi);
		}

	#pragma omp parallel for schedule (static, chunk)
		for(i=0;i<1000;i++)
		{
			dizi[i] = rand()%1000;
		}

		min = dizi[0];
		max = dizi[0];

	#pragma omp parallel for schedule (static, chunk)
		for(i=1;i<1000;i++)
		{
			if(dizi[i]<min) min = dizi[i];
			if(dizi[i]>max)	max = dizi[i];
		}

	#pragma omp parallel for schedule (static, chunk)
		for(i=1; i<1000; i++)
			for(j=i; j<1000;j++)
			{
				if(dizi[i]>dizi[j])
				{
					temp = dizi[i];
					dizi[i]=dizi[j];
					dizi[j]=temp;
				}
			}

	#pragma omp single
	{
		  cout << "Rastgele Uretilen 1000 Deger ; " << endl;

			for(i = 1; i<1000; i++)
				cout << dizi[i] << ",";

			printf("\n");

			cout << "\nDizinin En Buyuk Elemani : " << max << endl;
			cout << "Dizinin En Kucuk Elemani : " << min << endl;

			degerAraligi = max - min;
```

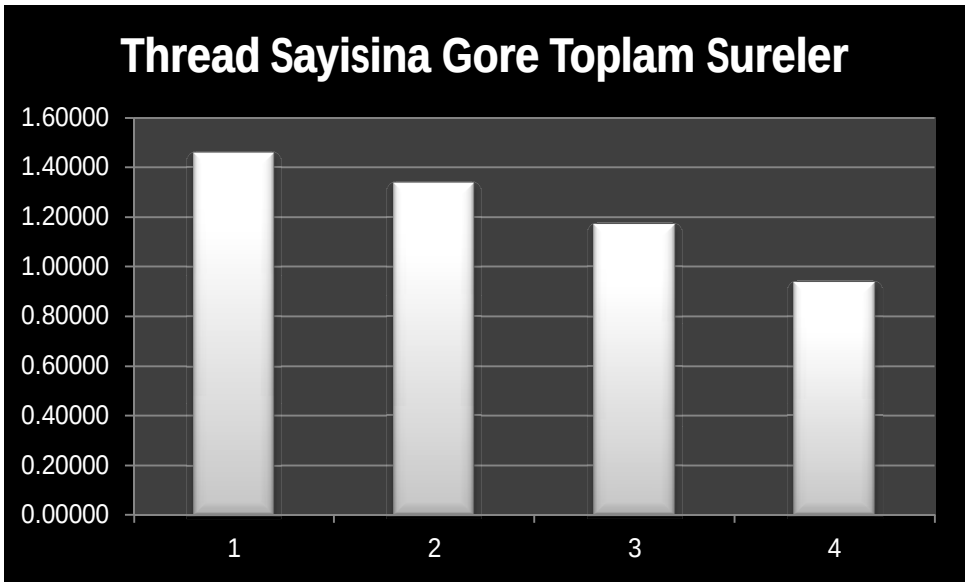
```
                    cout << "Dizinin Deger Araligi (max-min) : " << degerAraligi <<
endl;

                    cout << "Histogram Icin Sutun Sayisi Giriniz : ";
                    cin >> sutunSayisi;

                    aralikOlcumu = ((float) degerAraligi) / ((float) sutunSayisi) +
0.5f;
                    cout << "Aralik Olcum Sonucu : " << aralikOlcumu << endl;
        }

        baslangic[0] = min;

        #pragma omp parallel for schedule (static, chunk)
                for(i = 0; i<sutunSayisi; i++)
                {
                        baslangic[i+1] = (baslangic[i] + aralikOlcumu);
                        bitis[i] = (baslangic[i+1] - 1);
                        bitis[i+1] = (bitis[i] + aralikOlcumu);
                }

        #pragma omp parallel for schedule (static, chunk)
                for(j = 0; j<sutunSayisi; j++)
                {
                        sayac = 0;
                        for(i = 1; i<1000; i++)
                        {
                                if ((dizi[i] >= baslangic[j]) && (dizi[i] <= bitis[j]))
                                        sayac++;
                        }
                                printf("\n");
                                printf("%d - %d Araligi ... \t %d Adet \t |", baslangic[j],
bitis[j], sayac);
                                for(int k=0; k<sayac; k++)
                                                printf("*");
                }
        }

        printf("\n");
        printf ("\nHesaplama Islemi Sona Erdi : ");
        HesaplamayiBitir = clock();
        printf("%f saniye surdu. \n",(double) HesaplamayiBitir );
        SureFarki = ((double) (HesaplamayiBitir - HesaplamayaBasla)) / CLOCKS_PER_SEC ;
        printf("Hesaplanan Sure Farki : %f  saniyedir.", SureFarki);

        getche();
        return 0;
}
```

PARALEL PROGRAMLAMA ANALİZ SONUCU;

Paralel bir sekile donusturulmus olan histogram hesaplama uygulamasinda dort thread'le birlikte calisilmis olup, baslangic ve bitis surelerini birer degisken de tutttuktan sonra farklarini alarak calisma surelerini belirlemis oldum. Ornek ekran ciktisinda yapilan islemlerin suresi 0.936 saniye surmustur.

Paralel Programlama Analiz Sonuclari

Thread Sayisi	Paralel Prog. Toplam Suresi
1	1.45800
2	1.33700
3	1.16800
4	0.93600



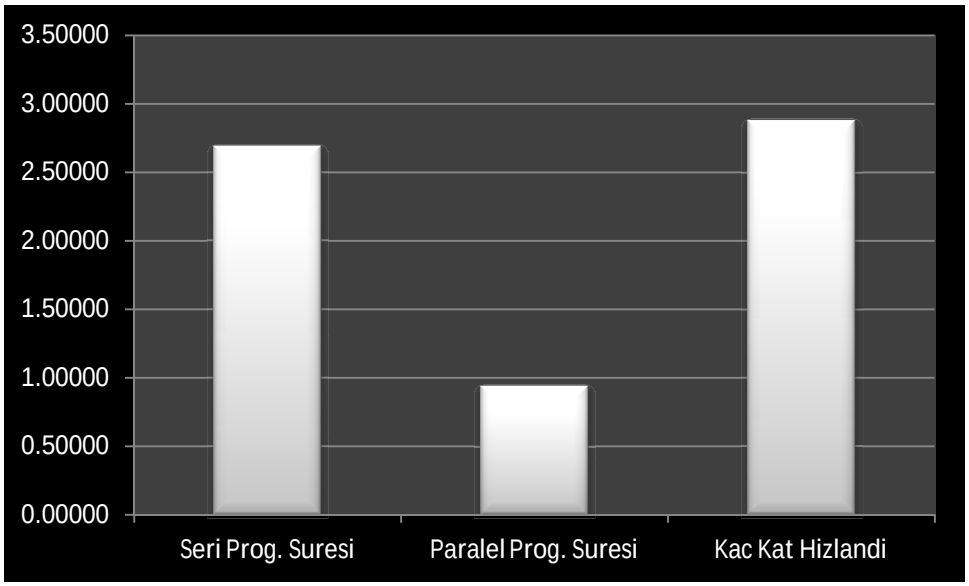
Paralel koda cevrilmis uygulamamizi Omp_Num_Threads=1 degerine gore 1.458 saniye , Omp_Num_Threads=2 degerine gore 1.3370 saniye, Omp_Num_Threads=3 degerine gore saniye ve Omp_Num_Threads=4 degerine gore de 0.9360 saniyeye kadar thread'lerin calisma surelerinin geriledigini gozlemleyebiliriz.

Seri/Paralel Programlama'da Hiz Karsilastirilmasi (Paralel'de 4 Thread)

Seri kodla yazmis oldugum bir uygulamayi paralel bir kodla yeniden duzenleyip derledikten sonra hiz konusunda ne kadar fazla artabilecegini gostermek icin mazx 4thred ile calisarak bir analiz yapacak olursak;

Seri / Paralel Programlama'da Hiz Karsilastirmasi (Paralel'de 4 Thread)

Seri Prog. Suresi	Paralel Prog. Suresi	Kac Kat Hizlandi
2.69300	0.93600	**2.87714**



Histogram Hesaplama'da 1000 tane uretilmis rastgele sayilarin seri kod ile paralel kod seklinde calistiginda aralarindaki sure farkinin saniye bakiminda gozlemlendiginde yaklasik 2.5 ile 3 saniye arasinda degistigini gozlemleyebiliriz.