Paralel ve Sıralı Brute Force Algoritmasının Karşılaştırılması

Özet: Günümüzün hızla gelişen teknolojisi olan bilgisayarlar kişisel, bilimsel, ticari, askeri gibi birçok alanda kullanılmaktadır. Araştırmacılar kullanıcıların daha kısa zamanda daha fazla işlem yapmasına olanak sağlamak için bilgisayarların hızlarını arttırmaya çalışmaktadırlar. Günümüzde bilgisayarların hızları fiziksel limitlere yaklaştığı için donanım üreticileri birden fazla işlemci içeren çok çekirdekli işlemciler üretmektedirler. Fakat günümüzde kullanılan yazılımların büyük bir kısmı sıralı olarak programlanmıştır. Bu sebeple bu yazılımlar çok işlemcili bilgisayarların kapasitesini verimli olarak kullanamamaktadır. Paralel programlama büyük problemleri küçük parçalara ayırıp çözmekte, bu sayede küçük problemler farklı işlemcilerde eşzamanlı olarak hesaplanabilmektedir. Bu araştırmanın amacı, sıralı ve paralel Brute Force Algoritmasının çalışma zamanı farklılıklarını incelemektir.

Anahtar Sözcükler: Sıralı Programlama, Paralel Programlama, Algoritma, Brute Force, İşlemcik

Comparison of Parallel and Sequential Brute Force Algorithm

Abstract: Computers, one of today's rapidly evolving technology, are used in many areas such as personal, scientific, commercial and military. Researchers are working to increase the speed of computers to enable the users perform more processing in less time. As processors speeds is approaching the physical limits now, hardware manufacturers produce multi-core processors which includes more than one core inside. However, most of today's software is programmed in traditional sequential way. Therefore these software can not use the capacity of multi-core computers efficiently. Parallel programming solves large problems by taking into smaller pieces, so that minor problems can be calculated simultaneously in different processors. The purpose of this study is to analyze the run-time differences of sequential and parallel Brute Force algorithm.

Keywords: Sequential Programming, Parallel Programming, Algorithm, Brute Force, Threads

1. Giriş

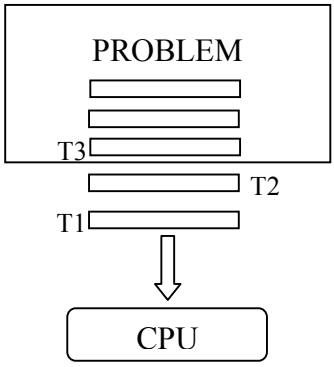
Günümüzün hızla gelişen teknolojisi olan bilgisayarlar kişisel, bilimsel, ticari, askeri gibi birçok alanda kullanılmaktadır. Araştırmacılar kullanıcıların daha kısa zamanda daha fazla işlem yapmasına olanak sağlamak için bilgisayarların hızlarını arttırmaya çalışmaktadırlar. Bir bilgisayarın hızı genellikle işlemcisinin hızıyla; işlemcisinin hızı ise saniyede yapabildiği işlem sayısıyla ölçülür [1]. Moore'un kuralına göre (Moore's Law) işlemcilerin hızları her iki yılda bir iki katına çıkmaktadır [2]. Peki bu artış ne kadar devam edebilir? İşlemcilerin hızı içerisinde kullanılan yarı iletkenlerin fiziksel özellikleriyle sınırlıdır ve günümüzde işlemci hızları neredeyse fiziksel limitlere dayanmıştır [3]. Donanım üreticileri bu problemi aşmak için bir chip'in içerisine birden fazla işlemci yerleştirmişlerdir. Bu sayede bilgisayarlar gerçek anlamda aynı anda birkaç işlem yapabiliyorlar. Fakat günümüzde kullandığımız yazılımların birçoğu sadece tek işlemci üzerinde çalışacak şekilde programlandığı için bilgisayarların kapasitesini kullanamamaktadır [4].

Paralel programlanmış algoritmalar günümüzde sıralı algoritmalara göre çok daha iyi sonuçlar ortaya koymaktadır [5]. Bu çalışmada sıralı Brute Force (BF) algoritması paralel olarak programlanmıştır. Her iki algoritmanında çalışma zamanları process explorer [6] programıyla ölçülerek paralel programlamanın performansı araştırılmıştır. Bu algoritmalar 4 GB RAM'a sahip İntel E7500 (Core 2 Duo, 2,93 GHz.) işlemcili bir masaüstü bilgisayarında denenmiştir. Dört, beş ve altı haneli 5'er adet olmak üzere toplam 15 adet şifre sıralı ve paralel algoritmalarla çözülmüş ve process explorer programı yardımıyla çalışma zamanları kaydedilmiştir.

Çalışmanın ikinci bölümünde paralel programlama ve BF algoritmaları hakkında bilgi verilmiştir. Üçüncü bölümde sıralı ve paralel olarak programlanmış algoritmalar çalıştırılarak, çalışma zamanları karşılaştırmalı olarak incelenmiştir. Son bölümde ise araştırmadan elde edilen sonuçlar ve daha sonra yapılabilecek araştırmalarla ilgili öneriler yer almaktadır.

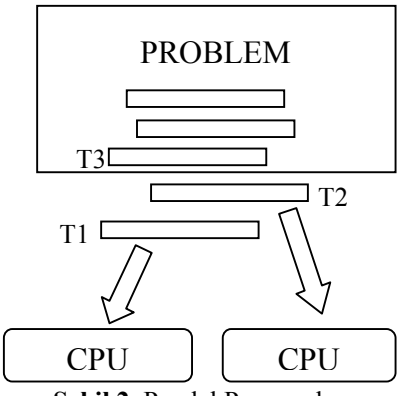
2. Paralel Programlama ve Brute Force

Yapılacak olan işlemin tek bir bilgisayarda ve tek işlemci üzerinde çalıştırılmasına "Sıralı Programlama" denir.



Şekil 1. Sıralı Programlama

Paralel programlama ise, bir problemi çözmek için birden fazla bilgisayar kaynaklarının aynı anda kullanılmasıdır.



Şekil 2. Paralel Programlama

Şekil 2. de görüldüğü gibi paralel programlamada bir problem birden fazla işlemci kullanarak ya da çoklu çekirdek

teknolojisine sahip bir işlemci yardımıyla çözülür [7].

BF araması ya da kapsamlı aramanın, bilgisayar bilimlerinde bilinen diğer bir adı da "oluştur ve test et" aramasıdır. Bu arama algoritması genel bir problem çözme tekniğidir. Bu teknikte bir problemi çözmek için bütün olası ihtimaller oluşturulur ve problem deneme yanılma yoluyla çözülmeye çalışılır [8].

2.1 Paralel Programlama

Günümüzde bilgisayar donanımları, yazılımların ihtiyaçlarına cevap vermekte zorlanmaktadırlar. Yazılımlar her geçen gün daha fazla hafıza ve daha hızlı bilgisayarlara ihtiyaç duymaktadırlar. Bilgisayar hafızaları daha fazla yarıiletken kullanarak arttırılabilirken, bilgisayarların hızı fiziksel limitlere dayandığı için arttırmak neredeyse imkânsızdır. Bilgisayar donanımlarını üreten mühendisler birden çok bilgisayar işlemcisini birbirlerine paralel çalışabilecek şekilde üreterek bu sorunu aşmışlardır. Günümüzde akıllı cep telefonlarında ve tablet bilgisayarlarda bile çift/dört çekirdekli işlemciler oldukça yaygın olarak kullanılmaktadır. Bu paralel işlemci mimarisini etkin olarak kullanabilmek için yazılımların paralel olarak programlanması gerekmektedir. Sadece tek işlemci üzerinde çalışacak şekilde sıralı olarak programlanmış algoritmalar çok çekirdekli işlemci mimarisinin gücünü kullanamamaktadır. Bu sebeple sıralı bir algoritmayı paralel hale getirmek algoritmanın performansını her zaman arttırır [9].

Paralel program yazmanın temelde "Görev ve Veri Paralelleştirme" şeklinde 2 yöntemi vardır. Görev paralelleştirme (Task Parallelism), bir problemin çözümünde kullanılan çeşitli işleri farklı işlemcilere dağıtmaktır. Veri paralelleştirme (Data Parallelism), bir problemin çözümünde kullanılan verileri farklı işlemcilere dağıtmaktır. Görev paralelleştirmede işlemciler algoritmanın farklı kodlarını çalıştırırken, veri paralelleştirmede işlemciler neredeyse aynı kodları farklı veriler üzerinde kullanırlar[8].

2.2 Brute Force (BF)

BF algoritması en temel, basit algoritma olarak görülebilir. Bir şifreyi çözmek için olası bütün ihtimalleri deneyerek şifreyi bulmaya çalışır[10]. BF algoritması temel metot olduğu için diğer algoritmaları kıyaslarken de kullanılabilir.

Bu arama algoritmasının ne kadar zaman ve hafıza harcayacağı olası muhtemel çözüm kümesi adaylarının çokluğuyla ilgilidir. Bir çok problem de de bu çözüm kümesi adayları çok devasa boyutlara ulaşabilir. Bu sebeple, BF algoritması 128 bit ve üzeri şifreler için uygun değildir [11]. BF algoritması genellikle problemin boyutu sınırlı ve az olduğu zamanlarda kullanılır. Bazı durumlarda da çözüm kümesi adaylarının makul miktarlara düşürülebileceği probleme özel sezgisel yöntemlerle de kullanıldığı olmaktadır.

```
Brute_Force(…)
{ C = ilk(P);
  While ( C ≠ son(P) )
  {  if (C=sifre) return (C);
     C=sonraki;
  }
}
```

Şekil 3. BF Algoritması Kaba Kodu

Bu algoritmayı uygulamanın 4 temel prosedürü vardır: ilki, sonraki, kontrol ve çıkış. Program ihtimalleri denemeye ilk elemandan başlar. **İlk** eleman C değişkenine aktarılır. Program C değişkeninin şifre olup olmadığını **kontrol** eder. Eğer ilk eleman şifre değilse **sonraki** eleman C değişkenine aktarılır ve while döngüsü tekrar çalışır. Şifre bulunasıya kadar ya da dizinin son elemanına kadar bu işlem devam eder ve ardından programdan **çıkar** [8].

BF algoritmasının en temel dezavantajı, gerçek problemlerin birçoğunda olası çözüm kümesi adayları setinin caydırıcı oranda geniş olmasıdır. Olasılıklar, ihtimaller arttıkça çözüm kümesi adaylarının sayısı katlanarak artmaktadır. Örneğin 29 harfin kullanıldığı 10 basamaklı bir şifrenin çözüm kümesi adaylarının sayısı 29^{10} dur. Buda en az 10^{15} ihtimalin denenmesi demektir. Basamak sayısı 2 artarsa (12 basamak) çözüm kümesi adaylarının sayısı 29^{12} yani en az 10^{18} ihtimalin denenmesi demektir. Basamak sayısı 2 artmasına rağmen çözüm kümesi adaylarının sayısı yaklaşık olarak 1000 kat artmıştır. Bu da 10 basamaklı bir şifre 1 günde çözülüyorsa, 12 basamaklı bir şifre 1000 günde çözülür demektir. Bu istenmedik duruma "kombinasyonel patlama" adı verilir [7].

Bu araştırmada kullanılan veri seti büyük ve küçük harfleri, sayıları ve özel karakterleri içermektedir. Hazırlanan programda, 67 farklı karakter kullanılan 5 haneli bir şifre yaklaşık olarak 1 dakikada çözülürken, 6 haneli bir şifre yaklaşık olarak 60 dakikada çözülmektedir. 7 haneli bir şifrenin teorik olarak 3 günde çözülmesi beklenmektedir.

BF algoritmasının çalışma zamanı adayların hangi sırada test edileceği ile doğrudan ilgilidir. Bu araştırmada kullanılan sıralı ve paralel algoritmalar aynı veri setini aynı sırada kullanmışlardır.

3. Uygulama

Bu bölümde araştırmada kullanılan sıralı ve paralel BF uygulamasının algoritmaları açıklanarak programdan elde edilen sonuçlar karşılaştırılmıştır.

3.1 Sıralı Brute Force Algoritması

Şekil 4. de kaba kodu verilen BF algoritması temelde "kullanılanKarakerler" dizisindeki bütün karakterleri belirli bir sırayla deneyerek herhangi bir şifreyi çözmeye çalışır. Algoritmada kullanılacak karakterler "kullanılanKarakterler" adındaki bir dizide tanımlanıyor. Ardından program tek karakter olarak dizideki bütün elemanları "kontrol et" fonksiyonuna gönderiyor. Eğer bulunacak şifre tek karakterli değilse "tahminiSifreUzunluğu" değişkeni bir arttırılıyor ve iki karakterli anahtarlarla şifre çözülmeye çalışılıyor. Eğer şifre çözülmezse("eşleşti" değişkeni "false" ise) while döngüsü tekrar döner ve "tahminiSifreUzunlugu" değişkeni bir arttırılarak 3 karakterli şifreler denenir. Şifre bulunasıya kadar "tahminiSifreUzunlugu" bir arttırılarak şifre çözülmeye çalışılır [12].

```
Seri_Brute_Force()
{ kullanılanKarakterler= {… ,
  … , … , … };
  tahminiSifreUzunlugu=0;
  while (!eslesti)
  { tahminiSifreUzunlugu++;
    anahtarKarakterler = new
    char[tahminiSifreUzunlugu]
    select karakterSeti[0];
    kontrolEt (KarakterSeti,
    sifre);
  }
}
```

Şekil 4. Sıralı BF Algoritması Kaba Kodu

Hazırlanan brute force algoritmasında şifreyi kontrol etmek için daha etkili ve hızlı çalışan rekursif bir algoritma kullanılmıştır. Kullanılan algoritmanın kaba kodu şekil 5. de verilmiştir.

```
Kontrol_et(…)
{  for (karakterUzunlugu)
   {  if (mevcut<son)
        KontrolEt(mevcut++);
      elseif (mevcut==sifre)
      {   eslesti = true;
          sonuc = mevcut;
      }
      return;
   }
}
```

Şekil 5. Şifre kontrolü Algoritması Kaba Kodu

Algoritma 0'dan başlayarak dizideki bütün elemanları rekürsif olarak çağırmaktadır. Dizideki son karaktere ulaşıldığında oluşan anahtarları tek tek istenilen "şifre" ile karşılaştırmaktadır. Şifre bulunduğunda ise "eslesti" değişkeninin değeri "true" yapılarak döngüden çıkması sağlanmaktadır.

Yukarıda ifade edildiği gibi döngülerin çalışması için kendisinden önceki döngünün bitmesi beklenmektedir. 2 haneli şifrelerin denenmesi için bilgisayar öncelikle tek haneli şifreleri kontrol etmektedir. İki yada dört çekirdekli bilgisayar da kullanılsa bu algoritmanın tasarımından dolayı bilgisayar bu yazılımı sıralı olarak, tek bir çekirdekte çalıştırmaktadır.

3.2 Paralel Brute Force Algoritması

BF algoritmasının paralel hale getirmek için veri paralelleştirme yöntemi uygulanmıştır. Algoritmanın şifreyi çözmekte kullanacağı anahtar karakterler bölünmüştür. Bölüm 3.1'de anlatılan sıralı programlanmış BF Algoritmasının paralel programlanması için denenen şifreler basamak sayısına göre "Thread A" ve "Thread B" olmak üzere ikiye ayrılmıştır. Programın bir bloğu(Thread A) tek haneli şifreleri(1,3,5,…) denerken, diğer program bloğu (Thread B) çift haneli (2,4,6,…) şifreleri denemektedir. Bu sayede bilgisayarın iki çekirdeği de eş zamanlı olarak çalışmaktadır.

```
Thread A( )
{ kullanılanKarakterler= {… ,
  … , … , … };
  tahminiSifreUzunlugu=  -1 ;
  while (!eslesti)
 {  tahminiSifreUzunlugu+=2;
    anahtarKarakterler = new
    char[tahminiSifreUzunlugu]
    select
kullanılanKarakterler[0];
kontrolEt (anahtarKarakterler,
sifre);
} }
```

Şekil 6. Thread A Fonksiyonu Kaba Kodu

Şekil 6' da verilen "Thread A" program bloğunun kaba kodunda "tahmini şifre uzunluğu değişkeninin değeri 2 şer arttırılarak tek haneli anahtar karakterlerle şifre çözülmeye çalışılmıştır. "Thread B" program bloğunda "Thread A"dan farklı olarak "tahminiSifreUzunluğu"nun değeri 0 olarak atanmaktadır. "Thread B" program bloğu da bu sayede çift haneli anahtarlarla şifreyi çözmeye çalışmaktadır. Şifre çözülünce "eslesti" değişkeninin değeri "true" olmaktadır ve threadler çalışmayı durdurmaktadır.

3.3 Verilerin Analizi

Sıralı ve paralel olarak programlanmış algoritmalar aynı bilgisayar üzerinde çalıştırıldığında bazı şifrelerde sıralı programlanmış algoritmanın, bazı şifrelerde ise paralel programlanmış algoritmanın daha hızlı çalıştığı gözlenmiştir. Bellek kullanımları incelendiğinde paralel programlanmış algoritmanın yaklaşık %10 oranında daha fazla bellek harcadığı tespit edilmekle birlikte, 6 haneli şifrelerde bile 25 mb.'ı geçmediği belirlenmiştir. Günümüz bilgisayarlarının hafızaları düşünülünce bu farkın hız kadar önemli olmadığı düşünülebilir.

Tablo 1.de 4, 5 ve 6 haneli toplam 15 adet şifrenin sıralı ve paralel çalışma süreleri ve hızları yer almaktadır.

Paralel programlanmış algoritmanın ne kadar hız kazandırdığını ya da kaybettirdiğini belirlemek için sıralı programlanmış algoritmadan elde edilen zaman paralel programlanmış algoritmadan elde edilen zamana bölünmüştür (formül 1).

Tablo 1.de 4, 5 ve 6 haneli toplam 15 adet şifrenin sıralı ve paralel çalışma süreleri ve hızları yer almaktadır.

Tablo 1. Sıralı ve paralel program süreleri

		Sıralı Program Süresi (sn.)	Paralel Program Süresi (sn.)	Hız
4 haneli şifreler	6T$l	1,092	1,591	0,686
	pAu@	1,060	0,398	2,663
	zrS3	0,982	0,655	1,499
	#fHZ	1,014	1,357	0,747
	jYS6	1,045	0,296	3,530
	Ortalama	1,039	0,859	1,209
5 haneli şifreler	3r2!D	50,403	61,113	0,825
	cK$iZ	51,480	3,104	16,585
	8cz1J	57,642	68,781	0,838
	Li5fQ	50,013	41,387	1,208
	a4!i3	55,458	1,443	38,432
	Ortalama	52,999	35,166	1,507
6 haneli şifreler	b2!YsW	3445,531	138,498	24,878
	YZ-33d	3675,882	5764,744	0,638
	Rl4qs@	3494,219	3216,748	1,086
	dQ1Vts	3434,705	304,615	11,276
	9t3Nsa	3320,918	4729,779	0,702
	Ortalama	3474,251	2830,877	1,227

$$\text{Hız} = \frac{\text{Sıralı Çalışma Süresi}}{\text{Paralel Çalışma Süresi}} \qquad (1)$$

Paralel programlama BF Algoritmasını rastgele seçilen 4 haneli şifrelerde ortalama %21, 5 haneli şifrelerde ortalama %51, 6 haneli şifrelerde ise ortalama %23 oranında hızlandırmıştır.

Elde edilen değerler ayrı ayrı incelendiğinde bulunacak şifrenin ilk harfi veri setinde önlerde ya da ortalarda tanımlanmışsa paralel programlamayla, sonlarda tanımlanmışsa sıralı programlamayla daha hızlı çözüldüğü görülmüştür.

Sıralı programlanmış algoritma kullanıldığında program şifreyi çözesiye kadar bir döngüye girdiği için "yanıt vermiyor" hatasına sebep olmakta ve kullanılan arayüz üzerinde kontrolü kaybetmektedir. Paralel olarak programlanmış algoritma her iki işlemciyi de kullanmasına rağmen, threadleri kullandığı için ve döngüye giren bu threadler olduğu için "yanıt vermiyor" hatası oluşmamakta ve arayüz ekranı kilitlenmemektedir.

4. Sonuç ve Öneriler

Günümüzde masaüstü bilgisayar, cep telefonu ve tablet bilgisayarlarda genellikle çok çekirdekli işlemciler kullanılmaktadır. Fakat hali hazırda kullandığımız yazılımların çok az bir kısmı paralel olarak programlanmıştır. Bu sebeple yazılımların büyük bir çoğunluğu bilgisayar işlemcisinin gücünü yeteri kadar kullanamamaktadır. BF algoritmasının kalitesini belirleyen iki önemli etken vardır; en kısa sürede ve en az bellek kullanarak doğru şifreyi bulmak. BF algoritmasının paralelleştirilmesi sayesinde şifreler çok daha kısa sürelerde çözülebilmektedir.

Bu araştırmada paralel programlamanın şifreyi daha hızlı çözmesinin yanında veri setinde tanımlanan karakterlerin sırasının şifre çözümünde ne kadar etkili olduğu da görülmüştür. Veri setinde ilk başta tanımlanan küçük harfli alfabeler (a-z) daha çabuk çözülürken sonlarda tanımlanan sayı ve özel işaretler(0-9, #,!,-,@,$) ise daha geç çözülebilmiştir.

Araştırma sonuçları incelendiğinde 4 haneli şifrelerde %20, 5 haneli şifrelerde %50 ve 6 haneli şifrelerde %22 oranında hızlanma tespit edilmiştir. Kullanılan şifreler paralel programlama yardımıyla ortalama %31 oranında daha hızlı çözülmüştür. Paralel programlamada kullanılan yük dağılımı daha

iyi yapılırsa 2 kata kadar (%100) hızlanma gerçekleştirilebilir[13].

Ayrıca paralel programlandığında algoritmanın kernel threadleri kullandığı, sıralı programlandığında ise user threadler oluştuğu tespit edilmiştir.

Burada sunulan çalışmada BF algoritması paralel olarak programlanmış ve sıralı algoritmaya göre etkililiği incelenmiştir. İleride yapılacak çalışmalarda bu algoritma 4 yâda 8 çekirdek üzerinde çalışacak şekilde programlanırsa daha etkili olabilir. Ayrıca bu araştırmada kullanılan yük dağılımı dinamik hale getirilirse işlem zamanları daha tutarlı hale getirilebilir. Farklı algoritmalarında paralel olarak programlanmasının hızı arttıracağı ve donanım kaynaklarını daha verimli kullanacağı düşünülmektedir.

5. Kaynaklar

[1] İşlemciler(CPU), Bilişim Teknolojisi, megep.meb.gov.tr/mte_program_modul/modul_pdf/481BB0009.pdf, **Milli Eğitim Bakanlığı**, Ankara (2012).

[2] Gordon Moore, Moore's Law, Wikipedia en.wikipedia.org/wiki/Moore's_law#cite_note-IntelInterview-3 , (2012).

[3] Jonathan Kang , online erişim: 10/12/2012, www.quora.com/Why-havent-CPU-clock-speeds-increased-in-the-last-5-years, (2011).

[4] Techtarget, online erişim: 10/12/2012, www.bitpipe.com/detail/RES/1282247334_650.html, (2012)

[5] Jagna, A & Bhima,, K., "An improved order independent paralel thinning algorithm for image thinning", **International Journal of Computer Applications,** Vol. 46(3), (2012)

[6] Russinovic, M., Process Explorer V15.23 http://technet.microsoft.com/tr-tr/sysinternals/bb896653.aspx, **Microsoft,** (2012)

[7] Pacheco, P., "An Introduction to Parallel Programming", **Elsevier Press**, Massachusetts, (2011).

[8] Brute Force Search, online erişim: 10/12/2012,http://en.wikipedia.org/wiki/Brute-force_search, Wikipedia, (2008).

[9] Singh, D. P. & Khare, N., "A Study of Different Parallel Implementations of Single Source Shortest Path Algorithms", **International Journal of Computer Applications** 54(10), September (2012).

[10] Paar, C. & Pelzl, J., "Understanding Cryptography: A Textbook for Students and Practitioners", **Springer Press**, p.7, (2010).

[11] Rayarikar, R, Upathyay, S. and Shah, D., "An Encryption Algorithm for Secure Data Transmission", **International Journal of Computer Applications,** Vol. 40(7), February (2012).

[12] Woschitz, J., "A simple Brute Force Algorithm in C#", https://janosch.woschitz.org/a-simple-brute-force-algorithm-in-c-sharp/#bruteforce-source, Berlin, (2010).

[13] Sharma, S.K. & Gupta, K., "Performance Analysis of Parallel Algorithms on Multi-Core System using OpenMP", **International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)**, Vol 2(5), October (2012).