WEB ARAMA MOTORLARI İÇİN BAĞLANTI TEMELLİ BİR SIRALAMA ALGORİTMASININ PARALEL GERÇEKLENMESİ*

Adnan Burak Gürdağ
burak.gurdag@o2.com.tr
Oksijen Teknoloji

Can Özturan
ozturaca@boun.edu.tr
Boğaziçi Üniversitesi

Özetçe

Bağlantı temelli sıralama, Web sayfalarının, yardımlı metin linkleri ile kendi aralarında oluşturdukları bağlantı yapısına göre sıralandırılmasına dayanan bir tekniktir. Günümüzün en iyi arama motoru olarak gösterilen Google, başarısının büyük bölümünü PageRank isimli bağlantı temelli sıralama algoritmasına borçludur. PageRank, yöntem olarak aynı zamanda kişiselleştirilmiş aramaya imkan vermektedir. Ancak tüm web sayfalarının oluşturduğu çizge üzerinde koşan bu algoritmanın farklı her kullanıcı profili için çalıştırılması yüksek hesaplama gücü gerektirmektedir. Bunun yanında artan Web sayfaları için bu hesaplamanın yapılabilmesi için ölçeklenebilir bir yapı gerekmektedir.

Bu bildiride PageRank algoritmasının MPI kullanılarak paralel gerçeklenmesi ve bu gerçeklemenin iki Linux PC öbeği üzerindeki performansı ve ölçeklenebilirliği incelenmektedir.

Giriş

Web sayfalarının günden güne ve hızla artışı arama motorlarını önemini daha da arttırmaktadır. Sıradan bir Internet kullanıcısının bir arama motoruna başvurmadan aradığı kalitede bilgiye ulaşması artık iyice zorlaşmıştır. Son zamanların popüler arama motoru Google [1], uyguladığı ileri teknikler sayesinde kısa sürede en iyi arama motoru olma başarısını yakalamıştır. Bu tekniklerden bir tanesi arama sonuçlarının sıralanmasında kullanılan ve PageRank [2] adı verilen bir algoritmadır. PageRank, arama sorgusun-dan bağımsız, Web sayfalarının birbirlerine olan bağlantıları-nı gözönüne alarak her sayfa için bir önem değeri bulan bir yöntemdir. Web sayfaları yine PageRank adı verilen bu önem değerlerine göre kendi aralarında sıralanırlar. PageRank, normalde arama motorunun her dizinleme işlemi için bir kere çalıştırılır ve tüm sorgulamalarda aynı önem değerleri kullanılır.

PageRank yöntem olarak kişisel tercihlere göre Web sayfalarının önem sıralamasının değiştirilmesine, dolayısıyla kişiselleştirilmiş aramaya imkan tanımaktadır. Web'deki içeriğin hızlı bir şekilde artması kişiselleştirilmiş aramanın zorunlu bir servis haline gelmesini kaçınılmaz kılmaktadır. Ancak PageRank'in kişisel tercihlere göre tekrar tekrar çalıştırılması önemli miktarda hesaplama gücü gerektirmektedir. Dolayısıyla sayıları hızla artan Web sayfalarının oluşturduğu çok büyük çizgeyi kısa sürede işleyecek bir PageRank gerçeklemesi gerekmektedir. Bu durum gerçeklemenin hem yüksek performanslı hem de ölçeklenebilir olmasını gerektirmektedir.

Bu çalışmada PageRank algoritması, dağıtık bellek mimarisinde çalışacak şekilde MPI kullanılarak gerçeklendi. Bu gerçekleme, biri Fast Ethernet diğeri Myrinet ağ altyapısına sahip iki Linux PC öbeği üzerinde çalıştırılarak performansı ve ölçeklenebilirliği incelendi.

İlgili Çalışmalar

Haveliwala [3], PageRank hesaplamasının kişisel bilgisayarlarda yapılmasını önermiş ve bu hesaplamanın verimli yapılmasıyla ilgili bazı yöntemler önermiştir. Hesaplamanın arama motoru kullanıcılarının bilgisayarlarında yapılması düzenli aralıklarla kullanıcıların tüm Web çizge bilgisini güncellemelerini gerektirmektedir. Gigabaytlar tutan bu bilginin kullanıcı tarafına gönderilmesi ancak DVD-ROM gibi yüksek kapasiteli saklama ortamlarıyla mümkündür. Ancak insanların kişisel arama servisi alabilmek için böyle bir uğraşın içine girmeleri çok olası gözükmemektedir.

Dizisel PageRank Algoritması

PageRank tanım olarak Web sayfalarındaki yardımlı bağlantıları takip ederek dolaşan birinin herhangi bir t anında bir Web sayfasını ziyaret etme olasılığıdır [4]. Bu tanıma göre Web sayfalarının PageRank değerleri toplamı birdir. Dolayısıyla PageRank tüm Web sayfaları için bir olasılık dağılımını gösterir.

Bir Web sayfasının PageRank'i bu sayfada bulunan bağlantıların gösterdikleri sayfalara eşit olarak dağıtılır. Dolayısıyla, bir sayfanın PageRank'i, kendisine bağlantı vermiş olan sayfalardan gelen kısmi PageRank değerlerinin toplamına eşittir. Bir sayfanın PageRank'i şu eşitlikle ifade edilebilir:

$$R_v = \sum_{u \in B_v} \frac{R_u}{d_u}$$

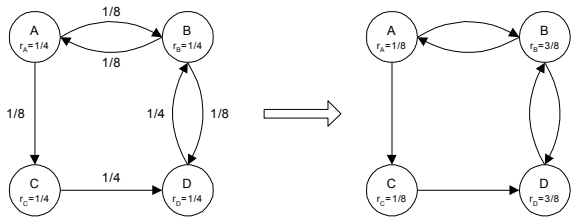
Burada R_u ve R_v, sırasıyla u ve v sayfalarının PageRank değerlerini, d_u, u sayfasında bulunan farklı yardımlı metin linklerinin sayısını ve B_v, v sayfasına bağlantı vermiş sayfaların kümesini göstermektedir.

n tane Web sayfasının PageRank'i şöyle hesaplanır:

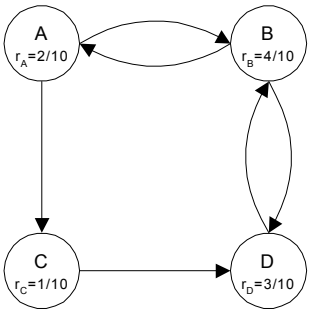
- Başlangıçta her sayfaya PageRank değeri olarak 1/n atanır.
- Döngüsel olarak yukarıda bahsedilen PageRank akışı gerçekleştirilir.
- Döngünün sonlanma şartı son iki döngüde elde edilen PageRank değerlerinin oluşturdukları vektörlerin aralarındaki Öklid mesafesinin belli bir ε değerinden küçük olmasıdır.

Şekil-1'de 4 sayfadan oluşan bir Web çizgesinde ilk döngüdeki PageRank akışı gösterilmiştir. Şekil-2'de bu akışın sonucunda her sayfanın almış olduğu PageRank değerleri görülebilir (Bu hesaplamada $\varepsilon = 10^{-4}$ alınmıştır). Buna göre bu örnekteki en popüler sayfa B sayfasıdır.

* Bu çalışma TÜBİTAK 199E009 projesi ve Boğaziçi Üniversitesi Araştırma Fonu Projesi 99A101 kapsamında kısmen desteklenmiştir.



Şekil-1: Örnek PageRank akışı (ilk döngü)



Şekil-2: Örnek PageRank akışı sonucu (13 döngüden sonra)

Yukarıda anlatılan PageRank hesaplaması aşağıdaki vektör matris çarpımı ile ifade edilebilir:

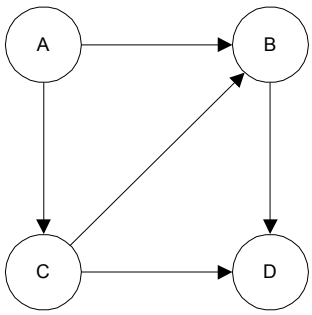
$$r^{(t)} = r^{(t-1)} \times P$$

Burada $r^{(t)}$ *n* sayfanın *t* anındaki PageRank dağılımını gösteren 1×n boyutunda bir vektördür. *P* ise n×n boyutunda satır-stokastik bir durum geçiş matrisidir. Yani, eğer *i* sayfasından *j* sayfasına bağlantı verilmişse, $P_{ij} = 1/d_i$ olmaktadır. Aksi takdirde P_{ij} 0 değerini alır. Burada her Web sayfası birer duruma, Web sayfaları arasındaki yardımlı metin bağlantıları da durumlar arası geçiş yaylarına denk düşmektedir. Özetle, PageRank hesaplaması aynı zamanda Web sayfalarının temsil ettiği bir Markov zincirinin düzgün durum dağılımının hesaplamasıdır.

Buraya kadar anlatıldığı haliyle PageRank hesaplama döngüsünün bir çözüme yakınsamasını engelleyen iki problem vardır. Bunlardan ilkine sıra sızıntısı (rank leakage), diğerine ise sıra kapanı (rank sink) denmektedir.

Sıra Sızıntısı

Başka sayfalara bağlantı vermeyen Web sayfaları, sıra değerlerini (ya da PageRank'lerin) diğer sayfalara aktarmadıkları için, sıra sızıntısına sebep olurlar. Bu durumda normalde bir olması gereken PageRank değerlerinin toplamı her döngüde azalır. Böylece sağlıklı bir sıralama yapılama-mış olur. Şekil-3'te bu problem için bir örnek verilmiştir. Bu problem için üç çözüm önerilmiştir. Bir çözümde, hiç bağlantı içermeyen sayfaların özyinelemeli olarak sistemden çıkarılıp PageRank hesaplamasının yapılması ve daha sonra bu sayfalara bir ayarlamayla PageRank değerleri atanması önerilmiştir [5]. Diğer bir çözüm olarak döngü sayısını sınırlamak gösterilmiştir [5]. Önerilen son ve hesaplama olarak en makul görünen çözüm, diğer sayfalara bağlantısı olmayan sayfalardan tüm sayfalara birer sanal bağlantı verilmesidir [6]. Böylece hesaplama sırasında sıra değerlerinin sistem içinde tutulması garantilenmiş olur. Gerek uygulama kolaylığı gerekse saklama ve hesaplama gereksiniminin azlığı sebebi ile son çözümün uygulanması makul gözükmektedir. Bu çalışmada, kullanılan örnek Web çizgelerinde böyle bir durum oluşmadığı için bundan sonraki bölümlerde bu çözüm açıkça uygulanmayacaktır.



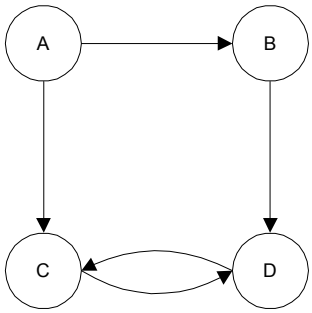
Şekil-3: Sistem D sayfası yüzünden PageRank kaybetmektedir.

Sıra Kapanı

Bazı sayfalar sadece birbirlerine bağlantı vererek kendi aralarında kapalı gruplar oluştururlar. Bu tür gruplar diğer sayfalardan bağlantı alır ama geriye bağlantı vermezler. Bu durum, bir süre sonra sistemdeki toplam PageRank'in bu gruplarda birikmesine ve hapis olmasına neden olur. Bu problem Şekil-4'te gösterilmektedir. Bu problem için önerilen çözüm şudur: PageRank modelindeki Web gezgini hangi sayfaya gideceğine karar verirken artık şöyle bir yöntem izler: *D* olasılıkla şu anda bulunduğu sayfadaki bağlantıların gösterdiği sayfalardan birine, 1-*D* olasılıkla rastgele ya da belirli bir dağılıma göre başka bir sayfaya gider (şu anda bulunduğu sayfada kalma ihtimali de söz konusudur). Buna PageRank hesaplaması aşağıdaki hali alır:

$$r^{(t)} = r^{(t-1)} \times DP + (1-D) \times p$$

Burada *p*, 1×n boyutunda bir dağılım vektörüdür. p_u, herhangi bir anda *u* sayfasına geçiş olasılığını göstermektedir. Eğer rastgele geçiş olasılığı kullanılırsa, p_u her sayfa için 1/n değerini alacaktır. *p* vektörüne *kişiselleştirme vektörü (personalization vector)* de denmekte-dir.



Şekil-4: C ve D sayfaları bir sıra kapanı oluşturmaktadırlar.

Başka sayfaya gitme seçiminde bir olasılık dağılımı kullanılması, tercihe göre farklı PageRank hesaplamaları yapılmasına imkan tanımaktadır. Dolayısıyla bu özellik sayesinde kişiselleştirilmiş sıralamalar elde edilebilmektedir. Örnek olarak bir arama motoru kullanıcısı kişiselleştirilmiş arama servisini kullanmak istediğinde, ilgilendiği konular ve web sayfalarını arama motoruna bildirip kendisine özel bir PageRank hesaplaması yapılmasını sağlar. Böylece bundan sonra bu kişinin yapacağı aramaların sonuçları kendisi için hesaplanmış PageRank değerlerine göre sıralanacaktır. Dolayısıyla bu kullanıcıya kişiselleştirilmiş arama servisi verilmiş olacaktır. Girişte de belirtildiği üzere, PageRank'in kişiselleştirilmiş aramaya olanak tanıması bu çalışmanın ana motivasyonlarından birini oluşturmaktadır.

Paralel PageRank Algoritması

Şekil-5'te bu çalışmada geliştirilen paralel PageRank algoritması görülmektedir.

```
Pi   : i numaralı işlemci
Vi   : Pi üzerindeki sayfalar
Fu   : u sayfasındaki bağlantıların
       gösterdiği sayfalar
Pid(v): v sayfasının bulunduğu işlemci
her işlemci Pi için
  iletişim bilgilerini oku
  kişiselleştirme vektörünü oku (p)
  her sayfa u ∈ Vi için
    ru(0) = 1/n
  kalan = ∞
  t = 0
  (kalan > ε) oldukça
    her sayfa u ∈ Vi için
      her sayfa v ∈ Fu için
        eğer Pi == Pid(v)
          rv(t+1) = rv(t+1) + ru(t)/du
        ya da
          ru(t)/du değerini Pid(v)'ye
           göndermek için bir yastıkta
           sakla
    gönderilecek her yastığı işlemciye
     gönder
    bu işlemciye gönderilen her yastığı
     diğer işlemcilerden al
    alınan her yastık için
     v için gönderilen her ru(t)/du değeri
      için
       rv(t+1) = rv(t+1) + ru(t)/du
    ltoplam = 0
    her sayfa v ∈ Vi için
      rv(t+1) = D*rv(t+1) + (1-D)*pv
      ltoplam += (rv(t+1) - rv(t))2
    gtoplam = Σ(tüm işlemcilerdeki
                ltoplam'lar)
    kalan = √(gtoplam)
    r(t) = r(t+1)
    t = t + 1;
```

Şekil-5: Paralel PageRank Algoritması

Paralel PageRank algoritmasında işlemciler daha önce hazırlanmış iletişim bilgilerini kullanarak birbirleriyle PageRank alışverişinde bulunurlar. İletişim bilgileri işlemcilerin hepsinin katılımıyla yine paralel olarak koşan bir ön hesaplama işlemi sonucunda oluşturulur. Her işlemci kendisiyle ilgili bilgileri yerel bir dosyada tutar. Bu işlem her Web çizgesi için sadece bir kez yapılır. İletişim bilgileri arasında aşağıdakiler bulunmaktadır:

a. PageRank gönderilecek işlemcilerin sayısı ve listesi
b. a'daki işlemcilerin her birine gönderilecek PageRank'lerin sayısı
c. PageRank alınacak işlemcilerin sayısı ve listesi
d. c'deki işlemcilerin her birinden alınacak PageRank'lerin sayısı ve bunların gönderildiği sayfaların listesi (işlemci başına bir liste)
e. yerel sayfaların sayısı ve listesi

Paralel PageRank yürütmesinin sonucunda her işlemci, hesapladığı kısmi PageRank vektörünü bir dosyaya yazar. Daha sonra işlemcilerden bu dosyalar toplanabilir ya da doğrudan sorgulanabilir.

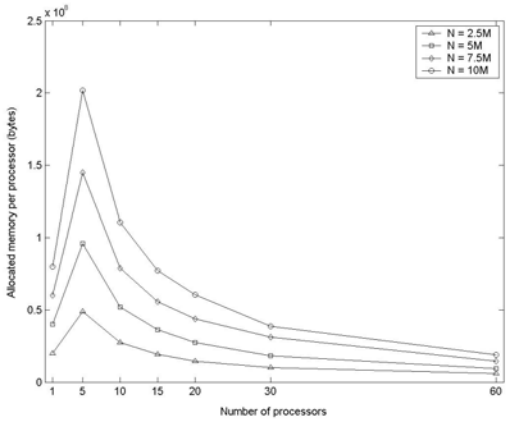
Çizge Bölümleme

PageRank hesaplaması yapılmadan önce Web çizgesinin işlemcilere uygun şekilde bölünmesi gerekmektedir. Burada önemli olan, bu bölümleme sonucunda işlemciler arasındaki PageRank akışlarının en az olması ve işlemcilerin hesaplama yüklerinin eşit dağılmasıdır. Şu anda Web çizgelerini bu kriterlere göre bölecek bir bölümleme algoritması bulunmamaktadır. Başlı başına bir araştırma konusu olabilecek bu konu bu çalışmanın kapsamı dışındadır. Bu yüzden bu çalışmada sayfalar, üzerlerindeki yardımlı metin bağlantılarıyla beraber işlemcilere eşit dağıtılmışlardır. İlginç bir şekilde, eşit dağıtım yapıldığında işlemcilerin hesaplama yüklerinin neredeyse eşit dağıldığı gözlemlenmiştir. Ancak bazı çizgelerde bazı işlemcilere fazla hesaplama yükü geldiği de görülmüştür. Deneylerde kullanılan örnek Web çizgelerinin üretildiği de göz önüne alınırsa sayfaların eşit dağılımının, tüm Web çizgeleri için hesaplama yükünü eşit dağıtacağı gibi bir genelleme yapılamaz.

Karmaşıklık Analizi

Bağlantı sayısı m olan bir Web çizgesinde dizisel PageRank algoritması O(m) zaman karmaşıklığına sahiptir. Buna karşılık paralel PageRank algoritması p işlemci ile O((m/p)log(m/p)) zaman karmaşıklığına sahiptir [7]. Burada-ki log faktörü yerel dizinlerinin global dizinlere dönüştürül-mesinde ve sayfaların işlemcilerinin bulunmasında kullanılan ikili aramalardan kaynaklanmaktadır. Web çizgelerinin yapısal özelliğine göre sayfa sayısı ve bağlantı sayısı arasın-da doğrusal bir ilişki vardır. Dolayısıyla karmaşıklık ifadelerinde m yerine n kullanılabilir. Nitekim deneylerde problem büyüklüğü olarak sayfa sayısı (n) temel alınmıştır.

Yer karmaşıklığı analizi [7]'de detaylı olarak yapılmıştır. Şekil-6'da dizisel ve paralel PageRank algoritmalarının gerçeklemelerinin ana bellek kullanımları grafik halinde sunulmaktadır.



Şekil-6: İşlemci başına harcanan ana bellek miktarları

Deneysel Tasarım

Amaç

Bu çalışmada yapılan deneylerde, geliştirilen paralel kodun örnek sistemler üzerindeki performansını ve ölçeklenebilirliğini gözlemlemek amaçlanmıştır.

Metrikler

Deneylerde gözlenen performans metrikleri şunlardır:

a. PageRank hesaplama döngüsünde harcanan toplam zaman

b. Dizisel algoritmanın gerçeklemesine göre paralel gerçeklemenin hızlanması (speedup). Hızlanma, dizisel çalışma zamanının paralel çalışma zamanı-na oranı olarak hesaplanır.

c. Paralel kodun verimliliği (efficiency). Verimlilik, hızlanmanın işlemci sayısına bölünmesiyle bulunmaktadır. Deney sonuçlarında yüzde olarak ifade edilmiştir.

Bu çalışmada paralel kodun ölçeklenebilirliğini gözlemek için, işlemci sayısı ve problem büyüklüğü artarken, verimliliğin ne kadar sabit kaldığı gözlenmiştir. Verimliliğin bu şartlarda sabit kalması sistemin ölçeklenebilirliğini göstermekte bir ölçü kabul edilmektedir [8].

Hızlanma, verimlilik ve ölçeklenebilirlik gözlemleri için referans alınmak üzere dizisel PageRank algoritması da gerçeklenmiştir.

Deney Ortamı

Geliştirilen paralel kod iki Linux PC öbeği üzerinde koşulmuştur. Bu sistemlerin özellikleri Tablo-1'de verilmiştir. ASMA isimli öbek Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü'nde, MYRI isimli öbek ise ABD'de bir üniversitede bulunmaktadır. Deneylerde her iki sistem için $D = 0.85$ ve $\varepsilon = 10^{-4}$ alınmıştır.

Table-1: Deneylerde kullanılan Linux PC öbeklerinin sistem özellikleri

Sistem Özellikleri	ASMA	MYRI
Linux kernel versiyonu	2.2.14	2.4.12 SMP
C Derleyici	egcs 2.91.66	gcc 3.0.2
Derleyici seçenekleri	optimization level 2	optimization level 2
MPI kütüphanesi	MPICH 1.2.1 over TCP	MPICH-GM 1.2.1
Ağ anahtar ortamı	HP4000M Fast Ethernet Switch	M3-E64 Myrinet Switch
Ağ arayüz kartları (AAK)	Intel EtherExpress 100	Myrinet M3S-PCI64B
AAK ham bantgenişliği	100 Mb/s	1.92 Gb/s
Büyük mesajlar için AAK MPI bantgenişliği	60 Mb/s	0.5-1 Gb/s
AAK MPI küçük mesaj gecikmesi	120 μs	10 μs
Deneylerde kullanılan işlemci sayıları	5, 10, 15, 20	5, 10, 15, 20, 30, 60
Makine başına ana bellek	128 MB	1000 MB
Makinelerdeki işlemciler	PII-400 Mhz	Dual PIII-1000 Mhz
Makinelerdeki disk ulaşımı	yerel	NFS üzerinden merkezi

Örnek Web Çizgeleri

Deneylerde kullanılan örnek Web çizgeleri, bu çalışma kapsamında geliştirilen bir üreteç ile gerçek Web çizgelerine yakın yapılarda oluşturulmuşlardır [7]. Deneylerde kullanılan Web çizgelerinin sayfa ve bağlantı sayıları Tablo-2'de gösterilmiştir.

Tablo-2: Deneylerde kullanılan örnek Web çizgeleri

n	m
250000	1378610
500000	2549871
1250000	5895383
2500000	12748989
3750000	19795627
5000000	25927066
7500000	37250620
10000000	53435470

Deney Sonuçları

Bu bölümde yukarıda bahsedilen iki Linux PC öbeği üzerinde gerçekleştirilen deney sonuçları sunulmaktadır.

MYRI

Yürütme Zamanı

Tablo-3'te MYRI üzerinde alınan yürütme zamanları gösterilmektedir. Tek işlemcili yürütme zamanlarının 5 işlemcili zamanlardan iyi olmalarının sebebi paralelleştirmeden kaynaklanan hesaplama ek yüküdür. Bu ek yük aynı zamanda kodun verimini tayin eden en önemli etkendir.

Tablo-3: MYRI için yürütme zamanları (sn)

İşlemci sayısı (p)	Sayfa sayısı (n)			
	2.5M	5M	7.5M	10M
1	30.57	59.62	87.58	125.56
5	49.63	116.37	165.69	246.55
10	23.40	57.74	74.68	111.84
15	14.53	33.95	47.13	69.83
30	6.63	16.70	22.71	33.46
60	3.59	9.65	12.30	17.70

Hızlanma

Tablo-4'te Tablo-3'teki yürütme zamanlarından hesaplanan hızlanma değerleri gösterilmektedir. İdeal olan hızlanma değerlerinin işlemci sayısına eşit olmasıdır. Ancak hesaplama ve iletişim ek yükü genellikle buna izin vermez. Tablo-4'teki değerler ideal durumdan oldukça uzaktır. MYRI için bunun başlıca sebebi hesaplama yüküdür. İletişim ek yükü MYRI'nin yüksek performans ağ altyapısı nedeniyle kendini çok göstermemektedir.

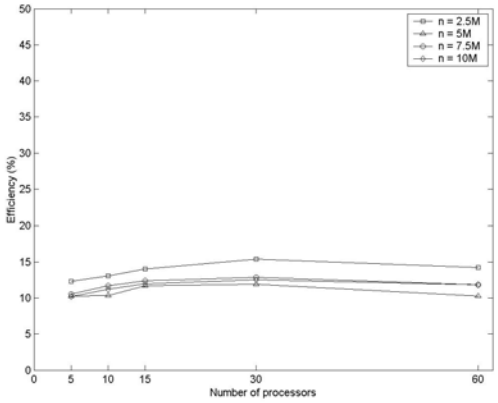
Asemptotik olarak bakıldığında, problem büyüklüğü arttıkça dizisel PageRank algoritmasının yürütme zamanı paralel olandan daha yavaş arttığından, hızlanma değerleri de aynı işlemci sayısı için problem büyüklüğü arttıkça azalmaktadır. Tablo-4'te 5M sayfa sayısındaki durumun bu gerçekle çeliştiği görülmektedir. Bunun sebebi, 5M sayfalık çizgenin bölümlemesinin bir işlemciye diğerlerinden çok daha fazla hesaplama yükü getirmesi dolayısıyla ortaya çıkan verimsiz işlemci gücü kullanımıdır.

Verimlilik

Şekil-7'de paralel PageRank gerçeklemesinin verimliliği grafik olarak gösterilmektedir. En yüksek verimlilik 2.5M sayfa ve 30 işlemcili hesaplamada görülmüştür (%15.37). Bir önceki bölümde hızlanma konusunda yapılan yorumlar verimlilik için de aynen geçerlidir.

Tablo-4: MYRI için hızlanma değerleri

İşlemci sayısı (p)	Sayfa sayısı (n)			
	2.5M	5M	7.5M	10M
1	1.00	1.00	1.00	1.00
5	0.62	0.51	0.53	0.51
10	1.31	1.03	1.17	1.12
15	2.10	1.76	1.86	1.80
30	4.61	3.57	3.86	3.75
60	8.51	6.18	7.12	7.10

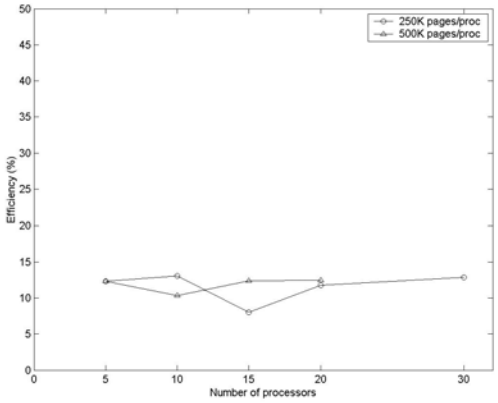


Şekil-7: MYRI için verimlilik yüzdeleri

Ölçeklenebilirlik

Şekil-8'de işlemci başına düşen sayfa sayısı iki farklı değerde(250 bin ve 500 bin) sabit tutulduğunda verimliliğin işlemci sayısına nasıl değiştiği gösterilmektedir. Burada görüldüğü gibi verimlilik iki nokta haricinde %12-13 civarında sabit kalmıştır. Bu iki noktadan bir tanesi yukarıda bahsedilen 5M sayfalık durum diğeri ise yine dağılım dengesizliği gözlenen 3.75M sayfalık durumdur.

Sonuç olarak gerçeklememizin işlemci başına 250 bin ve 500 bin sayfa için 60 işlemciye kadar ölçeklendiği görülmektedir.



Şekil-8: Paralel PageRank'in MYRI üzerinde ölçeklenebilirliği

ASMA

Yürütme Zamanı

Tablo-5'te ASMA'da elde edilen yürütme zamanları gösterilmektedir. Tabloda koyu yazıyüzü ile belirtilen zamanlar işlemcilerin ana bellek yetmeyerek getir götür yaptıkları yürütmelerin zamanlarıdır. Tek işlemcide 5M, 7.5M ve 10M sayfa için elde edilen zamanlarda anormal yükseklik göze çarpmaktadır (tabloda italik yazıyüzü ile belirtilen rakamlar). Normalde MYRI'de olduğu gibi zamanların doğrusal olarak artması gerekirken 2.5M sayfadan sonra yürütme beklenenden çok daha uzun sürmüştür. İncelemelerimiz sonucunda ASMA'da kullanılan Linux kernel üzerinde read() sistem çağrı sayısı belli bir sayının üzerine çıktığında çağrı sürelerinde bir artış olduğu görülmüştür. Birden fazla işlemci kullanıldığında işlemci başına yapılan çağrı sayısı muhtemelen bu sayının üzerine çıkmadığından elde edilen sonuçlarda bir gariplik görülmemektedir.

Tablo-5: ASMA için yürütme zamanları (sn)

İşlemci sayısı (p)	Sayfa sayısı (n)			
	2.5M	5M	7.5M	10M
1	53.38	*546.30*	*737.98*	*1044.57*
5	129.24	341.97	**827.66**	**2015.39**
10	65.41	145.73	195.35	**351.02**
15	43.45	99.22	130.06	191.61
20	39.43	75.09	96.29	147.11

Hızlanma

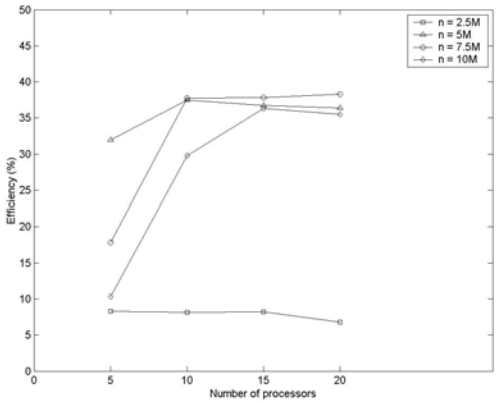
Tablo-6 ASMA'da paralel PageRank'in hızlanma değerlerini göstermektedir. Tek işlemcili sonuçlarda görülen normalin altında değerler nedeniyle 5M, 7.5M ve 10M sayfalık yürütmelerin hızlanma değerleri de normalin üzerinde görünmektedir. Bu değerler çalışmanın bütünlüğü açısından buraya konmuştur.

Tablo-6: ASMA için hızlanma değerleri

İşlemci sayısı (p)	Sayfa sayısı (n)			
	2.5M	5M	7.5M	10M
1	1.00	1.00	1.00	1.00
5	0.41	1.60	0.89	0.52
10	0.82	3.75	3.78	2.98
15	1.23	5.51	5.67	5.45
20	1.35	7.28	7.66	7.10

Verimlilik

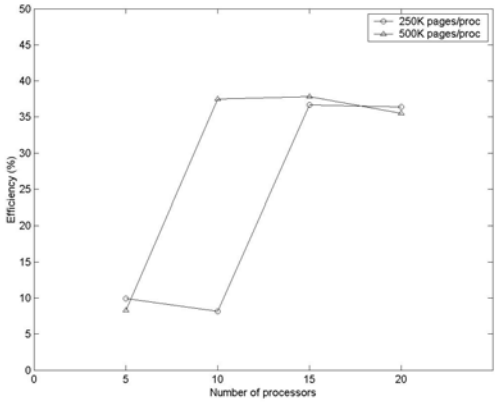
ASMA için verimlilik yüzdeleri Şekil-9'da gösterilmektedir. Bahsi geçen anormal yürütme zamanlarının etkisini şekilde görmek mümkündür. 2.5M sayfalık yürütme zamanlarının normal olduğu gözönüne alınırsa verimliliğin aslında %8 civarlarında olduğu söylenebilir. MYRI'den farklı olarak, ASMA'da verimlilik kaybına neden olan önemli bir etken ağ altyapısının yetersiz performansı olmuştur.



Şekil-9: ASMA için verimlilik yüzdeleri

Ölçeklenebilirlik

Şekil-10'da ASMA'nın işlemci başına 250 bin ve 500 bin sayfa için 20 işlemciye kadar ölçeklenebilirliği gösterilmiştir. Şekilde görülen iki farklı seviye daha önce açıklanan dizisel yürütme zamanlarındaki anormallikten kaynaklanmaktadır. Buna rağmen iki farklı seviyedeki verimlilik değerlerinin birbirlerine çok yakın olması paralel PageRank gerçeklemesinin ASMA'daki ölçeklenebilirliği hakkında olumlu bir fikir vermektedir.

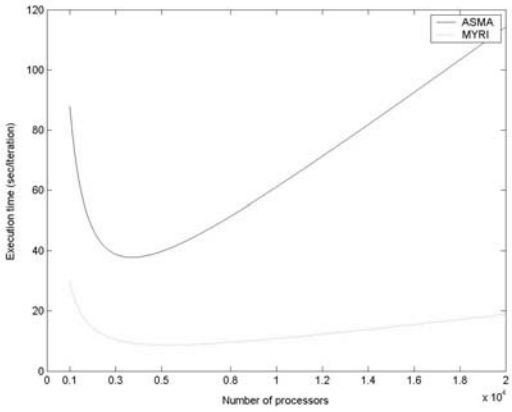


Şekil-10: Paralel PageRank'in ASMA üzerinde ölçeklenebilirliği

En İyi İşlemci Sayısı ve Zaman Tahmini

MYRI ve ASMA'da gözlenen yürütme sonuçları gözönüne alınarak ve en az kareler yakınsaması tekniği kullanılarak 2 milyar sayfanın PageRank hesaplaması için en iyi işlemci sayısı ve döngü başına zaman tahmini yapılmıştır [7]. Şekil-12'de bu tahminler gösterilmektedir. Buna göre MYRI için en kısa yürütme zamanı 5280 işlemciyle döngü başına 8.7 saniye olarak tahmin edilmektedir. Bu tahmin ASMA için 3720 işlemci ve 37.8 saniye olarak yapılmıştır.

Bu tahminlerde iki sistemin de deneylerdeki ölçeklenebilirliklerini koruyacakları varsayımı yapılmıştır. Bu varsayımların gerçeklikleri, kullanılan ağ altyapısı, algoritmalar ve veri yapıları ile yakından ilgili olduğundan tartışmalıdır.



Şekil-12: MYRI ve ASMA için en iyi zaman tahmini

Diğer Gözlemler

Döngü Sayısı

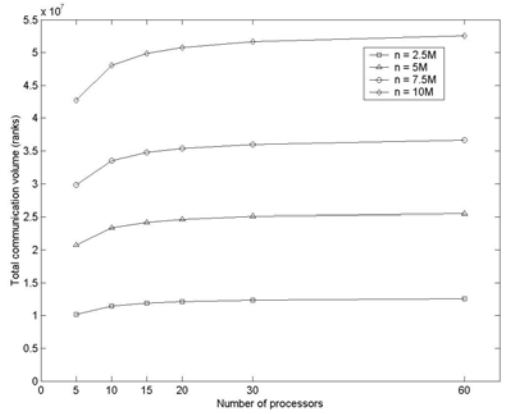
Deneylerde gözlenen PageRank döngülerinin sayısı Tablo-7'de verilmiştir. Burada N sayfa sayısı, C ise döngü sayısıdır. Dikkat edilirse döngü sayısı kullanılan işlemci sayısından bağımsızdır.

Tablo-7: PageRank döngü sayıları

N	250K	500K	1.25M	2.5M	3.75M	5M	7.5M	10M
C	12	11	11	10	9	10	9	9

İletişim Hacmi

Bir PageRank döngüsü sırasında işlemcilerin birbirlerine gönderdikleri PageRank değerlerinin sayısı Şekil-11'de gösterilmiştir.



Şekil-11: Bir paralel PageRank döngüsündeki iletişim hacimleri

İletişim Örüntüsü

Deneylerin hepsinde her işlemcinin kalan diğer işlemcilerle iletişim içine girdiği gözlenmiştir. Bunun sebebi işlemci sayısının sayfa sayısına nazaran çok az olması ve sayfaların bağlantı yapılarına göre oluşturdukları gruplar gözönüne alınmadan işlemcilere dağıtılmasıdır.

Okuma/Yazma Performansı

ASMA'da, PageRank yürütmesinin sonucunda elde edilen PageRank vektörlerinin, işlemciler tarafından dosyalara yazılması toplam yürütme zamanının %0.5'ini alırken bu oran MYRI'de %3'tür. Bu farkın nedeni MYRI'de G/Ç işlemlerinin ağ üzerinden merkezi bir dosya sistemine NFS ile yapılmasıdır.

Yürütme zamanını olumsuz yönde etkilediğinden, okuma performansı ile detaylı zamanlama bilgisi alınamamıştır. Ancak yoğun okuma gereksinimi yüzünden, NFS üzerinden diskten okuma yapmanın bu uygulama için ne kadar uygunsuz olduğu açıktır. Bu bakımdan lokal disk ulaşımı olması halinde MYRI'deki sonuçların çok daha iyi olacağı umulmaktadır.

Sonuç

Bu çalışmada arama sonuçlarında sağladığı kalite ispatlanmış PageRank algoritması, kişiselleştirilmiş arama servisleri verebilmek için MPI kullanarak paralel gerçeklenmiştir. İki Linux PC öbeği üzerinde yapılan deneylerin sonuçlarından görülebileceği gibi geliştirilen kodun ölçeklenebilirliği ve performansı umut verici olmakla beraber, uygun bir G/Ç alt yapısı, Web çizgeleri için tasarlanmış uygun bir çizge bölümleme tekniğinin kullanılması ve kodda yapılacak bazı iyileştirme çalışmalarıyla çok daha iyi sonuçların elde edileceği kesindir.

Kaynakça

[1] Google Arama Motoru, http://www.google.com

[2] **Brin, S. and Page L.**, 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine, Proceedings of the 7th International World Wide Web Conference.

[3] **Haveliwala, T. H.**, 1999. Efficient Computation of PageRank, Technical Report, Stanford University.

[4] **Page, L., Brin, S., Motwani R. and Winograd, T.**, 1998. The PageRank Citation Ranking: Bringing Order to the Web, Technical Report, Stanford University.

[5] **Page, L.**, 2001. Method for Node Ranking in a Linked Database, United States Patent 6285999, September.

[6] **Richardson, M. and Domingos, P.**, 2002. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank, to appear in Neural Information Processing Systems (NIPS).

[7] **Gürdağ, A. B.,** 2002. A Parallel Implementation of a Link-based Ranking Algorithm for Web Search Engines, *Yüksek Lisans Tezi*, Boğaziçi Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[8] **Kumar, V., Grama, A., Gupta, A. and Karypis, G.**, 1994. Introduction to Parallel Computing : Design and Analysis of Parallel Algorithms, Benjamin/Cummings Pub. Co., Redwood City, California.