PROJE RAPORU

1. PROJENİN ADI

Yeşil Kod

2. PROJENİN AMACI

Sunucu tabanlı web yazılımlarının (PHP, ASP, JSP vs) çalıştığı donanımın kaynaklarını verimli kullanarak, dinamik web sitelerinin web tarayıcısına yüklenme süresini azaltarak kullanıcılarına daha hızlı hizmet vermesini sağlayacak bir algoritma tasarlamak elektrik enerjisi tasarrufunda bulunmaktır.

Ayrıca bu proje ile yılda; 315 milyon dolar tasarruf ile aynı zamanda 5 milyon ağacın dikilmesine eş değer bir katma değer yaratmayı amaçladık.

3. GİRİŞ

Günümüzde internet dolayısı ile web siteleri hayatımızın ayrılmaz parçaları olmuştur. Her türlü işimizi web sayfaları aracılığı ile görür olduk. Yani web sayfaları olmadan (WWW) artık hayatımızı idame ettirmemiz çok zor gözüküyor. [1]

Diğer tarafta web sayfalarını kullanmanın da bir maliyeti var. Örneğin programlama giderleri, geliştirme giderleri, reklam ve çoğu kişinin pek farkında olmadığı enerji giderleri. Web siteleri de işlev görebilmeleri için elektrik enerjisine ihtiyaç duyar. Yani web sayfaları aslında elektrik ile çalışır. Bu elektriğin miktarı sayfaların kullanım sıklığı ve programlanması ile doğrudan ilişkilidir. İyi yazılmış kodlar daha az elektrik tüketecektir. Daha az elektrik tüketimi daha fazla tasarruf, daha az karbon salınımı ve daha az olumsuz klima etkisi getirecektir. Ama her ne olursa olsun programlama ve kod satırları gereklidir. Bunu sıfıra indirebilmemiz mümkün değildir. Ama tüm programlama iyileştirmelerinin ötesinde program kod satırlarının her zaman çalıştırılması gerekli midir, ne zaman çalıştırılsa en etkin kullanım

sağlanır soruları daha önemlidir. Biz bu proje ile bu sorulara cevap bularak büyük faydalar sağladık.

Genel olarak web sayfaları dolayısı ile siteler oluşturulurken günümüzde dinamik kodlar kullanılmaktadır. Dinamik kodlama Web 2.0 standardı ile hayatımıza girdi ve kullanıcı site etkileşimini en üst seviyelere taşıdı. [2]

Artan web sitesi kullanımı söz konusu olduğunda benim de yaşadığım en büyük sıkıntı birçok web kullanıcısının ortak sorunu haline geliyor. Yani ziyaretçi sayısı arttıkça sitede yavaşlamalar ve beklemeler olmaya başlıyor. Bunun nedeni sunucu kaynaklarının yetersiz kalmaya başlamasıdır.

Günümüzde PHP, ASP ve JSP en sık tercih edilen web dilleri olmuştur. Herkes bu dillerden birisini kullanarak sitelerine dinamik ve kullanıcı etkileşimli özellikler kazandırıyor.

Temel sorun her sayfa isteğinde bu dinamik dil kodlarının çalıştırılmasının gerekmesidir. Normalde siteler 1-10 gibi istekleri karşılayabilirken bu istek saniyede 100'lere ulaştığında sunucu bilgisayarın işlem gücü buna cevap veremez hale gelmektedir. Halbuki aynı sayfaya yapılan isteklerde dinamik kodların tekrar çalıştırılmasına gerek yok. Eğer bu kodların çalıştırılmış hali salt bir dosyada tutulabilirse hiç bekleme olmadan karşı taraftaki ziyaretçiye gönderilebilir. Biz bu projede çok basit donanım kaynakları ile çok düşük elektrik enerjisi tüketerek çok yüksek sayıda ziyaretçiye hizmet verebilecek çok yararlı bir sistem geliştirdik.

4. YÖNTEM

Projemizi gerçekleştirip sınayabilmek için bir geliştirme platformu seçmemiz gerektiğine karar verdik. Sonuçta algoritmamız temel olarak aynı yeteneklere sahip diğer geliştirme ortamlarına da sadece komut adı değişiklikleri ile uygulanabilecekti.

Bu amaçla dünyada Google, Facebook, Youtube gibi dev sitelerin de kullandığı en çok kullanılan web geliştirme dili olan PHP üzerinden geliştirme yapmaya karar verdik. [3]

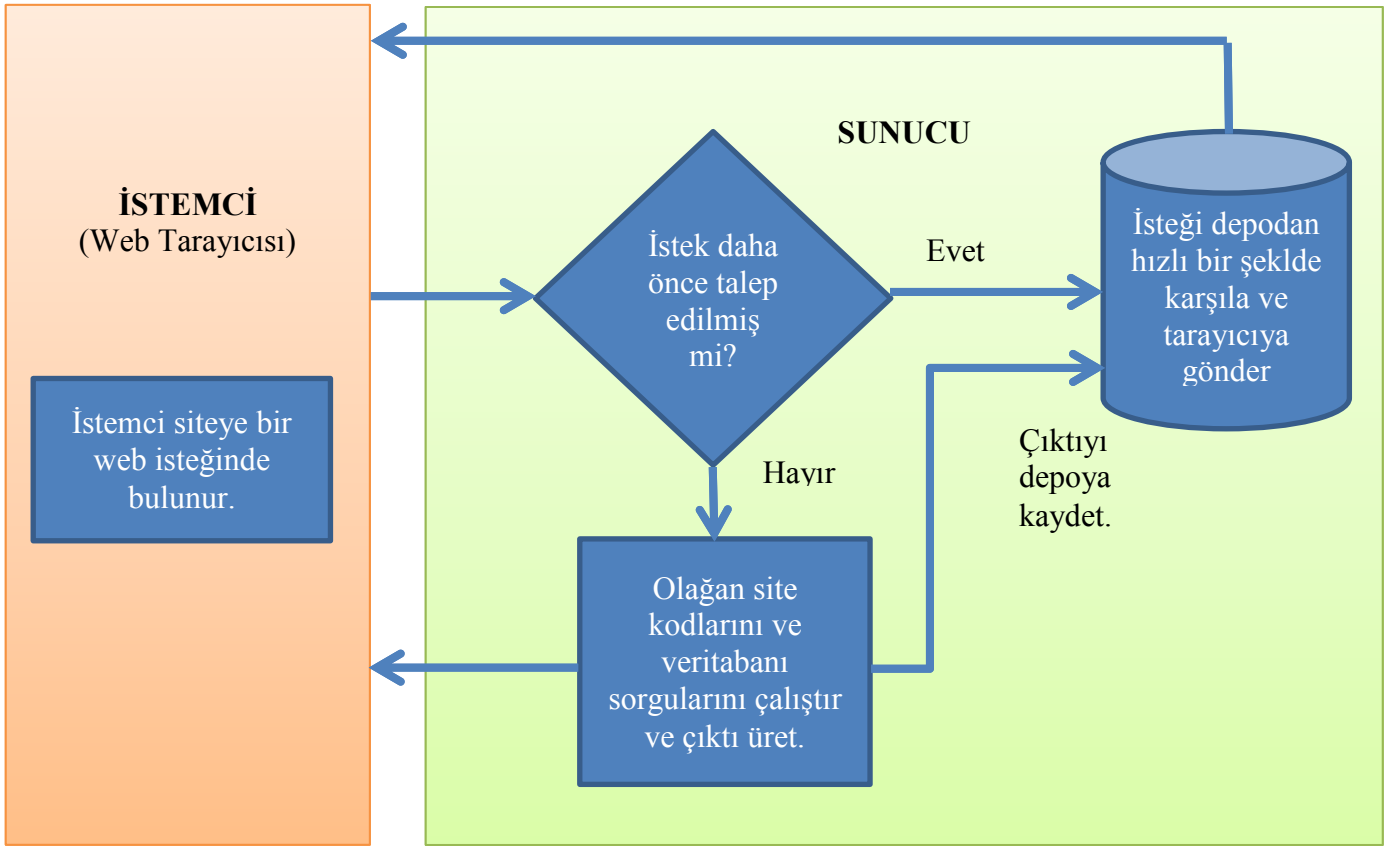
Geliştirdiğimiz algoritma kodunu sınayabileceğimiz gerçek bir web site çatısı arayışına girdik ve Wordpress'i keşfettik. Wordpress içerik yönetim sistemleri arasında en çok kullanılan açık kaynak kodlu bir site yapısıymış.[4] [5] [6]

Şuan dünyadaki web sitelerinin %21.6'sı Wordpress çatısı ile sunulmaktaymış. Toplam 15 milyar web sayfası arasının yaklaşık 3.24 milyarı Wordpress çatısının sayfalarından oluşmakta. [7]

Wordpress'in resmi sitesi olan Wordpress.org'daki duruma baktığımızda sadece 3.8 sürümünün 8.8 milyon kez geliştiriciler tarafından indirildiği görülebiliyor. [8]

1. **Kısım:** Sistemin çalışma şemasını çıkartılması

Çalışma şeması sistemin bel kemiğini oluşturuyor. Kodlamadan bağımsız olan bu çalışma şeması net ve sade olarak çıkartıldığında sadece programlama kodları değiştirilerek PHP dışında da herhangi bir sisteme uyarlanabilir.



Şekil-1: Yeşil Kod Sisteminin Çalışma Şeması

2. **Kısım:** Çalışma şemasına uygun olarak temel kodun hazırlanması ;

Çalışma şemasını karşılayacak şekilde PHP kodlarının yazılması bir hayli zaman aldı. PHP ve Apache ikilisinin işleyişinde çıktının tamponlanarak tarayıcıya gönderilmesi püf noktayı oluşturuyor. Apache PHP kodlarını, PHP yorumlayıcısına devrettikten sonra PHP tarayıcıya gönderilecek HTML çıktısının depolanacağı bir tampon oluşturuyor (output buffer) ve kodların çalışması bittikten sonra hepsi paketlere bölerek tarayıcıya gönderiyor. [9]

```php
<?php
	error_reporting(E_ALL ^ E_NOTICE);

	// istek yapılan URL'yi oluştur
	$istenen_url = 'http://'.$_SERVER[HTTP_HOST].$_SERVER[REQUEST_URI];

	// İstek yapılan URL için kullanılacak kaşe dosyası
	$kase_patika = dirname(__FILE__).'/depo/'.md5($istenen_url).'.yesil';

	// Kaşe dosyası mevcutsa oku, tarayıcıya gönder ve bitir
	if (file_exists($kase_patika))
		die(file_get_contents($kase_patika));


// Bu fonksiyon çıktı tamponunu (output_buffer)
// bir disk dosyasına kaydeder.
function dosyayi_kasele($cikti_tamponu)
{
   global $kase_patika;
   // Kaşe dosyası mevcut değilse oluştur
   if (!file_exists($kase_patika)){
      $sablon = "<!-- Bu sayfa %s tarihli, [%s] ön belleğinden getirilmiştir -->";
      $aciklama = sprintf($sablon, date('l jS \of F Y h:i:s A'),$_SERVER[REQUEST_URI]);
      file_put_contents($kase_patika, $aciklama.$cikti_tamponu);
   }
   return $cikti_tamponu;
}

	// Çıktı tamponu tarayıcıya boşaltılmadan önce
	// <dosyayi_kasele> alt yordamına gönder
	ob_start('dosyayi_kasele');
?>
```

Kod-1: yeşil_kod.php dosyamızın içeriği

Kodların çalışmasından kısaca bahsedersek. Kodu sitenin giriş dosyasının (*index.php*) en başına koymalıyız ki tüm site kodlarından önce bu kod çalışabilsin.

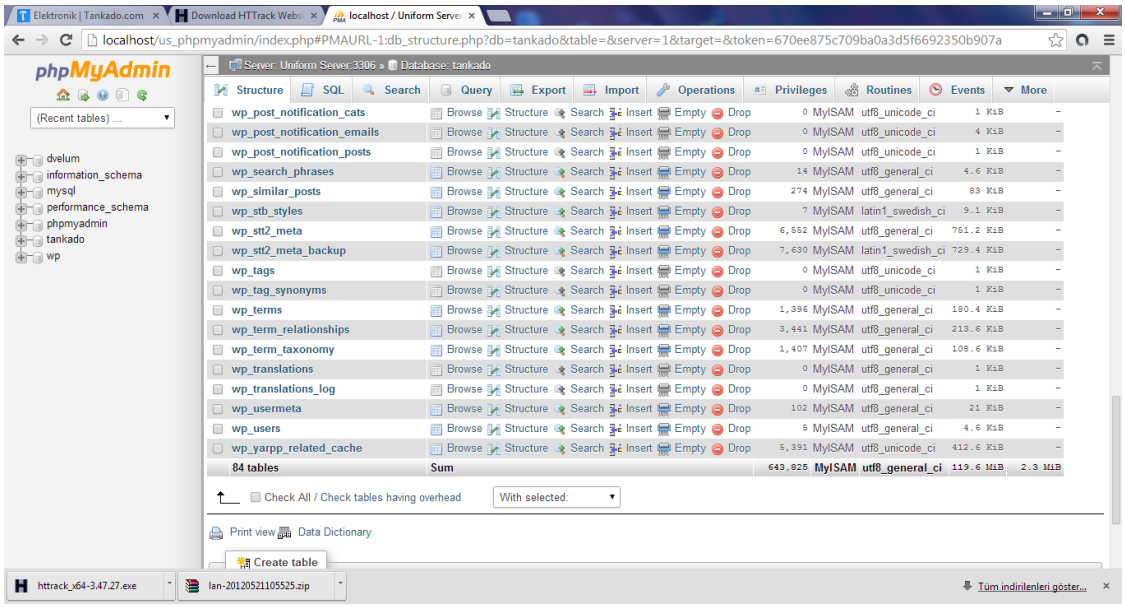
Öncelikle istek yapılan URL adresi tespit ediliyor ve bu adresin dosya sistemine kaydedilebileceği uygun ve tekil bir dosya adı oluşturmak için md5 ile özütü oluşturuluyor ve aynı dizindeki depo dizini gösteren bir patika tanımlanıyor.

Eğer istek yapılan URL'nin kaydı depo dizinin de yoksa kodun akışı bu bölümde duruyor. Sitenin geri kalanındaki kodlar normal çalışmalarına devam ediyor.

Burada kilit öneme sahip olan PHP'nin çıktı tamponunu kontrol eden ob fonksiyonlarından olan ob_start. Ob_start'a parametre olarak verdiğimiz *dosyayi_kasele* fonksiyonu PHP tüm çıktıyı tarayıcıya göndermeden önce PHP tarafından çalıştırılıyor. Biz de tam bu noktada PHP ürettiği bu çıktıyı depoya kaydediyoruz. Böylece daha sonra aynı URL tekrar istendiğinde depodan hızlıca karşılanabilecek. Daha sonra PHP'nin çıktısı tarayıcıya gönderiliyor. Kodların çalışması bitiyor.

3. **Kısım:** Kodun çalışmasının gerçek bir sistem üzerinde test edilmesi

Kodun çalışmasını test edebilmek için çok popüler ve büyük bir uygulama olan Wordpress'i sunucu bilgisayarımıza indirip kurduk ve testler yapabilmek için içerisinde yaklaşık 640 bin kayıt olan bir veritabanı ile ilişkilendirdik.



Resim-1: Sınama sistemimizin veritabanı tabloları

Wordpress'in giriş dosyası olan index.php dosyasının en başına yesil_kod.phpdosyamızı dahil ederek çalışmayı ilk olarak ele almasını sağladık.

```
<?php
/**
 * Front to the WordPress application. This file doesn't do anything, but loads
 * wp-blog-header.php which does and tells WordPress to load the theme.
 *
 * @package WordPress
 */

/**
 * Tells WordPress to load the WordPress theme and output it.
 *
 * @var bool
 */
require('yesil_kod.php');

define('WP_USE_THEMES', true);

/** Loads the WordPress Environment and Template */
require( dirname( __FILE__ ) . '/wp-blog-header.php' );
```

Kod-2: Wordpress'in giriş dosyasında yaptığımız değişiklik

Wordpress programlama ve kod açısından oldukça büyük bir sistem (sıkışmış hali 12MB) ve onbinlerce program satırından oluşuyor. Başta hedeflediğimiz gibi büyük bir sistem olan Wordpress'e sadece bir satır kod eklemesi yaparak entegrasyonu sağladık.

4. Kısım: Başarım testlerinin gerçekleştirilmesi

Başarım testlerini yapabilmek için sitemizi geçici bir alan adı atadık. Bu alan adını www.tubitak.com olarak belirledik ve istemci bilgisayara bu alan adına karşılık gelen IP adresini hosts dosyası ile tanımladık. Artık istemci bilgisayardan tubitak.com sitesi istendiğinde bizim test sistemimiz talebi karşılayacaktır.

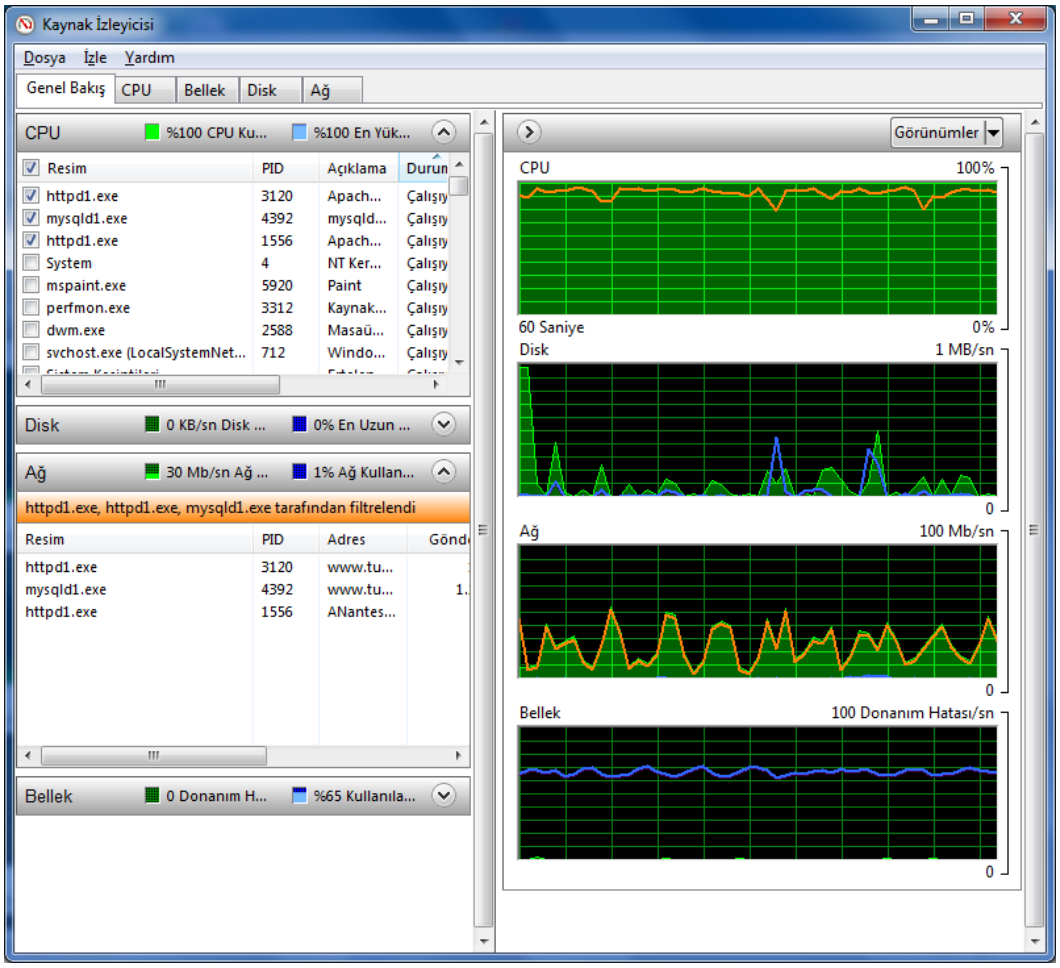
İstemci bilgisayarımız Pardus işletim sistemine sahip olduğu için birçok araca sahip. Wget aracı ise istek yapmak için kullanabileceğimiz çok yetenekli bir araçmış.

Wget'in yansılama özelliğini kullanarak aynı anda birden fazla çalıştırarak sunucumuzu zorladık. Aynı anda birden çok kopya çalıştırdığımızda sitemize aynı anda birden fazla ziyaretçi girişini simule etmiş olduk.

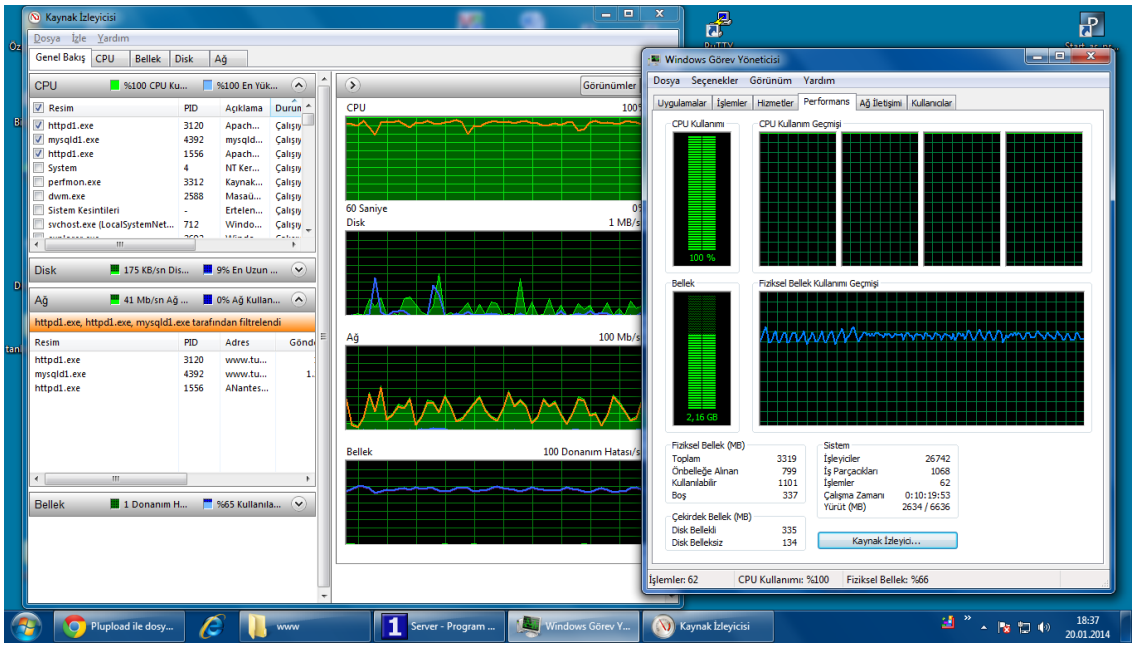
```
#!/bin/bash
for i in {1..20}
do
   wget –mk –domains tubitak.com http://www.tubitak.com/ &
done
```

Kod-3: Aynı anda 20 ziyaretçiyi taklit eden döngü kodumuz

Öncelikle sitemizi test ederken yesil_kod.phpolmadan başladık ve sunucunun yük durumu gözlemledik.



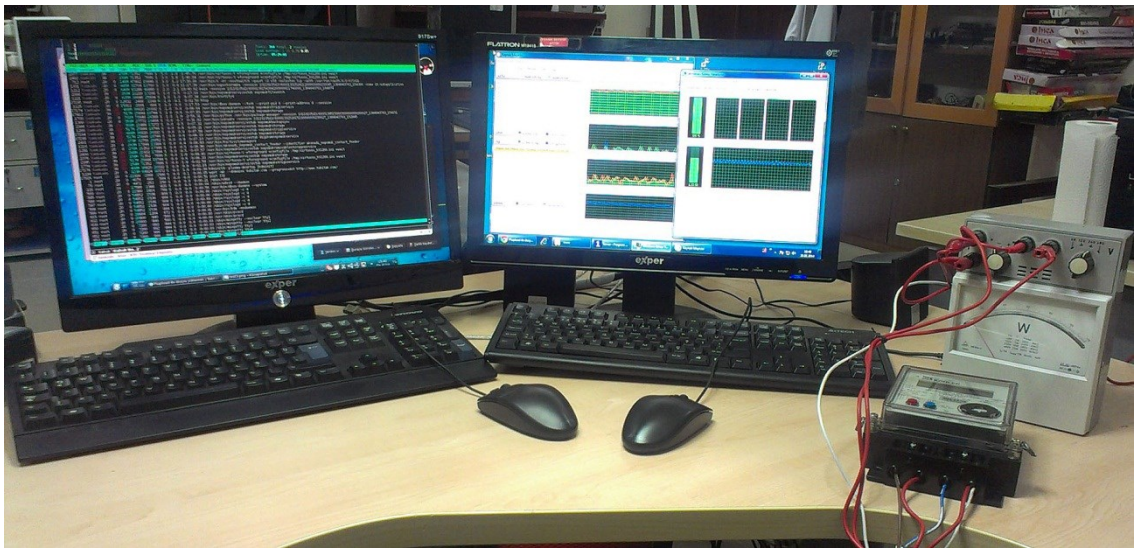
Resim-2: Yeşil Kod aktif değilken 20 ziyaretçi ile yapılan sunucu yük testi



Resim-3: Yeşil Kod aktif değilken 20 ziyaretçi ile yapılan sunucu yük testi

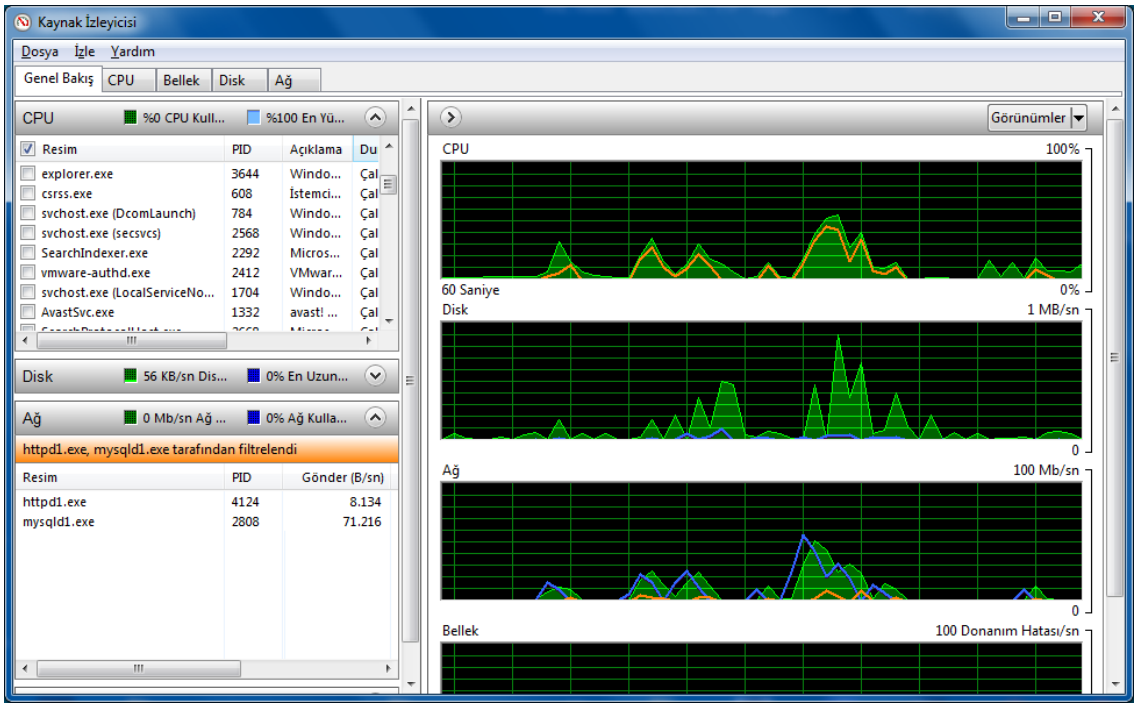
20 ziyaretçi ile yaptığımız testi sunucu karşılayabildi fakat bariz bir yavaşlama hissettik. İşletim sisteminin tepkileri oldukça azaldı ve CPU göstergesi hiçbir zaman %100'ün altına düşmedi. Bu test sırasında Wattmetremiz 95Watt civarını gösteriyordu. Watmetreye monitörü dahil etmedik. 64 bit'lik sistemimizde tek sabit disk, i5 3GHz işemci ve tek modül 4GB ram vardı.

10 ziyaretçi ile başladığımız sınamayı sırasıyla 20, 30 ve 40 lara çıkardık. 40 lı değerlerde sunucu neredeyse cevap veremez duruma geldi. Merakımıza yenik düşerek Pardus daki döngü değerini 400 yaptıktan kısa bir süre sonra sunucumuzun arabirimi ve faresi kilitlendi, uzunca bir beklemeden sonra bilgisayarımızı resetlemek zorunda kaldık. Bu arada elektriksel yük testlerini de yaptık, hangi yükler altında ne kadar elektrik tüketimi gerçekleştirdiğini de ölçtük.

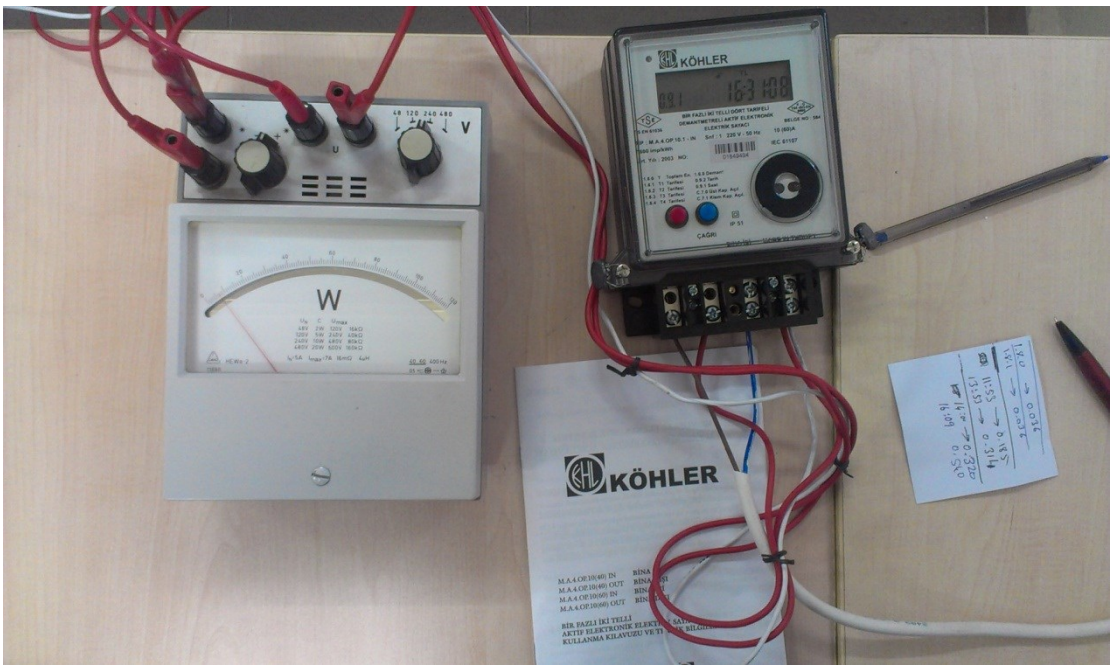


Resim-4: Sistemimizi Yeşil Kod akfif değilken 20 ziyaretçi ile test ederken

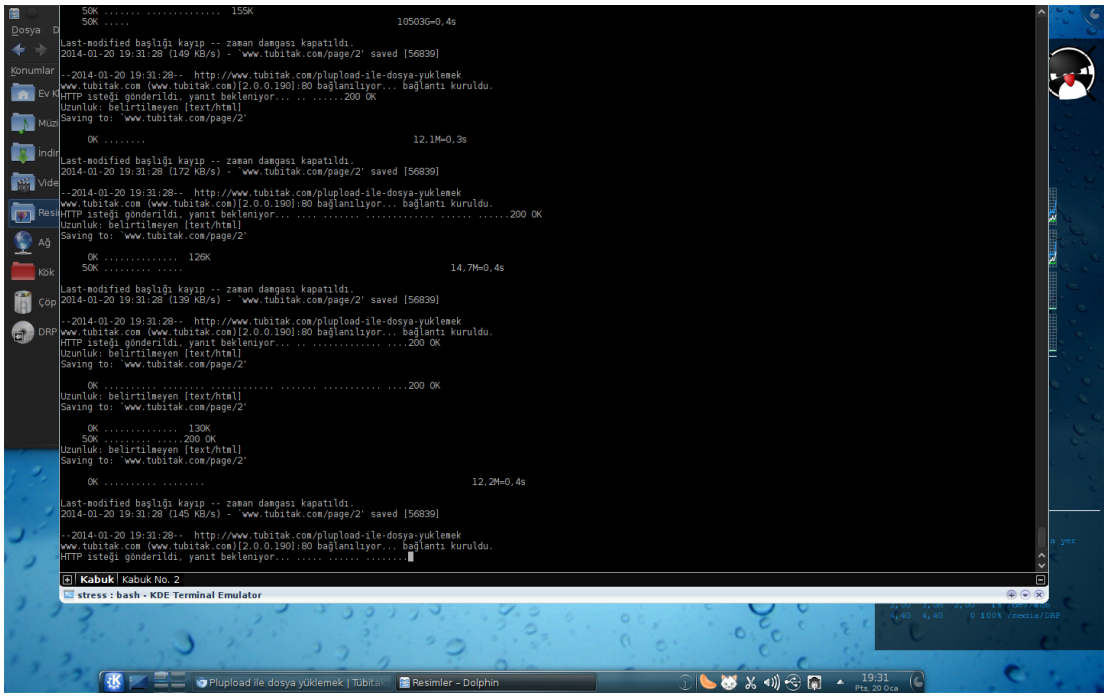
Evet Yeşil Kod'u devreye aldığımızda ilk olarak 10 ziyaretçi ile test yaptık ve sonuç mükemmeldi. CPU göstergemiz ortalama %10'lar civarında değişiyordu. Bundan sonra gönül rahatlığı ile ziyaretçi sayısını 20'ye çıkardığımızda CPU ancak %15'lere geldi ve Watt metremiz 55Watt civarında değer gösteriyordu.



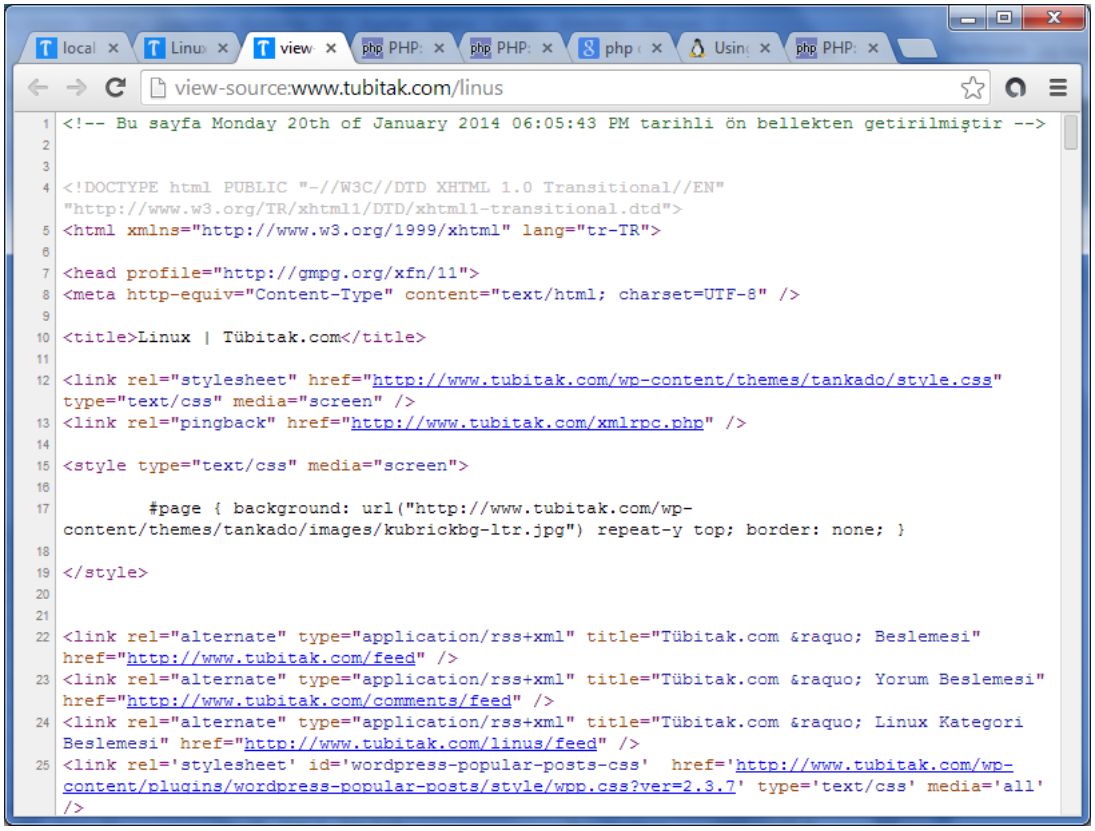
Resim-5: Yeşil Kod aktif ve 20 ziyaretçi ile test edilirken.



Resim-6: Enerji tüketimini ölçmek için kurduğumuz sistem



Resim-7: 20 farklı ziyaretçiyi simüle eden istemcimizin ekranı



Resim-8: İstemcimize gelen sayfanın kaynak kodu, üstte ön bellekten getirildiği bilgisi yer alıyor.

Kısım 5: Ölçümler

Uzun süreler testler yaptık. Sürekli taklit edilen ziyaretçi sayısını değiştirdik. Yeşil Kod aktifken veya pasifken yük testleri yaptık. Deneylerimiz arasında geçişler yaparken sürekli olarak güç (watt) ve iş (Kwatt/saat) değerlerini kayıt altına aldık. Aldığımız notlar arasından karşılaştırmalı sonuçlar aşağıdaki gibiydi.

Yeşil Kod	Ziyaretçi	Süre (dk)	Tüketim (kW/s)	Birim Tüketim	Değişim (%)
Pasif	50	129	0,220	1705	
Aktif	50	37	0,055	1486	12,838
Pasif	10	73	0,118	1616	
Aktif	10	62	0,077	1242	23,168
Pasif	20	59	0,101	1712	
Aktif	20	56	0,073	1304	23,851

Tablo-1: Yaptığımız testler sırasında ölçtüğümüz tüketim değerleri

Yaptığımız testlere göre ortalama %23,5 enerji tasarrufu sağlayabileceğimizi gördük. 50 paralel ziyaretçi ile testimizde aynı değişimi göremedik. Bunun nedenini sunucumuzun donanımının Yeşil Kod aktifken dahi sınır değerlerine yaklaşmış olması olarak yorumladık. Yine de %12'lik bir tüketim azalması ölçebildik. Saniyede 50 hit günlük 4.3 milyon hit'e denk gelmektedir.

Enerji tüketimi ile beraber sistemimizin asıl amacı olan hızlı tepki süresinde oldukça tatmin edici sonuçlar yakaladık. 20 paralel ziyaretçili testimiz sürerken bile çok seri sayfa açılma hızları yakalayabildik.

Gördük ki normal koşullarda böyle bir sistem ile en az %20 lik bir enerji tasarrufu ve yüksek sayfa hızı elde edebiliyoruz.

Elde ettiğimiz bu sonuçtan yola çıkarak dünya ölçeğinde orantısal bir tahmin yapmak gerekirse; ulaşılabilir kaynaklara göre dünyada;

- 15 milyar web sayfası [10]
- 148 milyon aktif web sitesi vardır [13]
- 150-300 watt aralığından güce sahip sunucuların ortalama elektriksel gücü 250Watt olarak tahmin edilmiştir [13]

- Dünyadaki en fazla sunucuya sahip şirketlerin (Google, Facebook, Akamai, Rackspace, Intel, Ebay, Yahoo, GoDady) toplam sunucu sayısı 4 milyon civarındadır. [14][15][16] (Bu şirketlerin dışındaki sunucu sayıları hesaba katılmamıştır)

Elimizdeki bu verilere göre dünyadaki tüm sunucuların yıllık elektrik sarfiyatı aşağıdaki formül ile yaklaşık olarak hesapladık.

*Toplam Sarfiyat = 365 gün * 24 saat * 0,25 kW/saat * 4 milyon sunucu*

Bu formül bize dünyadaki ölçülebilen sunucuların yılda en az (*diğer şirketlerin sunucuları hesaba katılmamıştır*)

8760 milyon kW/saat enerji tükettiğini göstermektedir.

Wikipedia verilerine göre elektriğin dünyadaki ortalama kilowatt saati 18 amerikan sentidir. [18] Buna göre araştırmamızdan elde ettiğimiz tasarruf oranına göre bizim sistemimiz ile dünyadaki tüm sunucular ölçeğinde yapabileceğimiz tasarruf;

Yeşil Kod ile sağlanabilecek tasarruf ; 8760 * %20 * 18 cent = **315 milyon dolardır**

Ayrıca Carbonfund'a göre kW/saat başına 0.0005925 ton CO_2 emisyonu oluşmakta [19] ve tek bir ağaç ile yılda 21 kg CO_2 emisyonu yok edebilmektedir. [20]

Bu da projemiz ile 1,038 milyon ton CO_2 emisyonunun doğaya salınmasını önleyerek 5 milyon ağaç gücünde bir fayda yaratmaktadır.

4. SONUÇ VE ÖNERİLER

Yaptığımız bu çalışma web sitelerinin yüklenme hızını üst seviyede artırarak sunucu kaynaklarının asgari seviyede kullanılmasını sağlamıştır.

Bu çalışma sayesinde sunucuların elektrik tüketimi azaltılarak aynı sunucu donanımı üzerinde daha fazla web sayfasının sunumunun yapılabilmesini sağlamıştır.

Tahmini hesaplamalarımıza göre yılda 315 milyon dolar tasarruf sağlamış, 5 milyon ağacın da korunmasına hizmet etme potansiyeli sunmuştur.

Projemiz sayesinde web siteleri statik biçime dönüştürülerek dinamik kodlar istemciden yalıtılmış böylece olası programlama zafiyetlerine karşı güvenlik de sağlanmıştır.

Projemizin gerçekleştirdiğimiz hali oldukça temeldir. Geliştirilmeye açıtır.

Projemizi daha da geliştirerek özel ihtiyaçlara göre konfigüre edilebilir hale getirmek mümkündür. Örneği bir web uygulamasını seçimlik olarak belirli kısımlarına uygulanabilir hale getirmeyi, belli zaman aralıklarında çalışabilir hale getirmeyi, çeşitli kullanım istatistiklerini de sunabilir hale getirip yaygınlaştırmayı amaçlıyoruz.

Projemizin sonuna geldiğinizde gördüklerimiz bizi çok heyecanlandırdı ve bu sistemi daha da geliştirmeyi düşünüyoruz.

5. KAYNAKLAR

[1] http://www.westhost.com/blog/2010/09/14/why-is-a-website-important-5-reasons-to-be-online/

[2] http://www.business2community.com/seo/the-benefits-of-web-2-0-sites-0412871

[3] http://w3techs.com/technologies/overview/programming_language/all

[4] http://yoast.com/wordpress-stats/

[5] http://www.google.com/trends/explore#q=WordPress,Joomla,Drupal

[6] http://w3techs.com/technologies/history_overview/content_management/all

[7] http://www.worldwidewebsize.com/

[8] http://wordpress.org/download/counter/

[9] http://www.creativedev.in/2012/05/php-output-buffering-explained/

[10] http://www.worldwidewebsize.com/

[11] http://www.zooknic.com/Domains/counts.html

[12] http://www.dnexpert.com/2007/11/27/world-domain-count-keeps-on-climbing/

[13] http://perspectives.mvdirona.com/2013/07/17/CountingServersIsHard.aspx

[14] http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/

[15] http://www.netcraft.com/internet-data-mining/hosting-provider-server-count/

[16] http://storageservers.wordpress.com/2013/07/17/facts-and-stats-of-worlds-largest-data-centers/

[17] http://www.mmo.org.tr/resimler/dosya_ekler/dd924b618b4d692_ek.pdf

[18] http://en.wikipedia.org/wiki/Electricity_pricing

[19] http://www.carbonfund.org/how-we-calculate

[20] http://www.arborenvironmentalalliance.com/carbon-tree-facts.asp