

**T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ**

**PHP VE MYSQL TABANLI
UYGULAMALARDA PERFORMANS
YÖNETİMİ**

Yüksek Lisans Tezi

VURAL ÖKCÜ

İSTANBUL, 2016

**T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ**

**PHP VE MYSQL TABANLI
UYGULAMALARDA PERFORMANS
YÖNETİMİ**

Yüksek Lisans Tezi

VURAL ÖKCÜ

Tez Danışmanı: YRD. DOÇ. DR. YÜCEL BATU SALMAN

İSTANBUL, 2016

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ

Tezin Adı: PHP ve MySQL Tabanlı Uygulamalarda Performans Yönetimi
Öğrencinin Adı Soyadı: Vural ÖKCÜ
Tez Savunma Tarihi: 27.05.2016

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğu Fen Bilimleri Enstitüsü tarafından onaylanmıştır.

Doç. Dr. Nafiz ARICA
Enstitü Müdürü
İmza

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğunu onaylarım.

Doç. Dr. Mehmet Alper Tunga
Program Koordinatörü
İmza

Bu Tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Jüri Üyeleri

İmzalar

Tez Danışmanı
Yrd. Doç. Dr. Yücel Batu Salman

Üye
Doç. Dr. Mehmet Alper Tunga

Üye
Yrd. Doç. Dr. Sabri Serkan Güllüoğlu

TEŐEKKÖR

Bu alıŐmayı hazırlamamda manevi desteklerini hibir zaman eksik etmeyen eŐim Dilan ÖKCÖ'ye, aileme ve yardımları nedeniyle tez danıŐmanım Yrd. Do. Dr. Yücel Batu SALMAN'a teŐekkür ederim.

İstanbul, 2016

Vural ÖKCÖ



ÖZET

PHP VE MYSQL TABANLI UYGULAMALARDA PERFORMANS YÖNETİMİ

Vural ÖKCÜ

Yüksek Lisans, Bilgi Teknolojileri

Tez Danışmanı: Yrd. Doç. Dr. Yücel Batu Salman

Mayıs 2016, 79 sayfa

PHP ve MySQL; internet üzerinde hizmet veren çoğu web sitesinin temelini oluşturan sistemlerdir. Bu sistemlerin en verimli çalıştığı işletim sistemleri ise Linux dağıtımlarıdır. Bu altyapı üzerine kurulmuş uygulamalar; depolanan verilerin ve ziyaretçi sayılarının artmasına paralel olarak performans sorunlarıyla karşılaşmaktadır. Böyle durumlarda ya donanımsal olarak iyileştirme ya da yazılımsal optimizasyon ihtiyacı doğmaktadır. Donanımsal iyileştirmeler belli bir noktadan sonra büyük maliyetler yaratacağı için, yazılımın optimize edilmesi en uygun çözümdür.

Yazılım optimizasyonu konusunda yapılabilecek ilk işlem, uygulama kaynak kodlarının incelenmesi ve düşük performansa neden olan kodların düzenlenmesidir. Bu nedenle, bu çalışmada öncelikle PHP profil aracı ile PHP kod performansı incelenmiştir. Sonraki aşamada, önbellekleme mekanizmalarının PHP tabanlı bir uygulamada çalışma süresine etkisi JMeter ve ab test araçları ile araştırılmıştır.

İkinci bölümde; veri tabanı konusunda yapılan performans analizlerine yer verilmiştir. MySQL sorgu optimizasyonu mysqlslap ile test edilmişken, MySQL'e alternatif olabilecek durumlarda kullanılan NoSQL veri tabanı sistemleri ve özellikle veri tabanında arama işlemlerinin sık yapıldığı uygulamalarda üçüncü parti arama motoru sunucu yazılımları ab ve JMeter ile test edilmiştir.

Üçüncü bölümde; sunucu tabanlı yazılımsal işlemlerin uygulama performansına etkisi incelenmiştir. Bu konuda; farklı web sunucu yazılımları, PHP işleyiciler, PHP yorumlayıcılar ve web hızlandırıcıların uygulama üzerindeki etkileri araştırılmıştır. Çalışma kapsamında gerçekleştirilen testlerde, uygulamanın yanıt süresinde yüzde 99'a varan azalmalar kaydedildiği gözlenmiştir.

Anahtar Kelimeler: PHP, MySQL, Apache, NoSQL, performans testi

ABSTRACT

PERFORMANCE MANAGEMENT ON PHP AND MYSQL BASED APPLICATIONS

Vural ÖKCÜ

Graduate, Information Technologies

Thesis Supervisor: Yrd. Doç. Dr. Yücel Batu Salman

May 2016, 79 pages

PHP and MySQL are the systems that form the base of the most websites serving on the internet. The operating systems with which these systems work most efficiently are Linux distributions. Applications based on this infrastructure encounter performance problems in parallel with the increase in stored data and number of visitors. In such cases; system needs either improvement in hardware or optimization in software. Since hardware improvements will cause to great costs after a point, optimization of the software is the most appropriate solution.

The first action that can be taken about the issue of software optimization is analysis of application source codes and editing codes causing low performance. Therefore, PHP code performance was analysed through PHP profiler first in this research study. In the next step, the effect of caching mechanisms on response time of a PHP based application was investigated through JMeter and ab testing tools.

The performance analysis which was conducted about database took place in the second section. While MySQL query optimization was analysed through mysqlslap, NoSQL database systems in cases that could be an alternative of MySQL and third party search engine server software that was used in the applications in which searching process is often seen were tested through ab and JMeter testing tools.

In the third section; effect of server-based software processes on application performance was analysed. In this issue, the effects of different web server software, PHP handlers, PHP interpreters and web accelerators on application were searched. At the tests applied on scope of this study, up to 99 percent decreases on application response time were observed.

Keywords: PHP, MySQL, Apache, NoSQL, performance test

İÇİNDEKİLER

TABLolar	ix
ŞEKİLLER	xiii
KISALTMALAR	xiv
1. GİRİŞ	1
2. LİTERATÜR TARAMASI	3
2.1 PHP VE MYSQL	3
2.1.1 PHP'nin Çalışma Prensipleri.....	3
2.1.2 PHP Programlama Dilinin Avantajları.....	4
2.1.3 MySQL Veri Tabanı Yönetim Sisteminin Avantajları.....	5
2.1.4 PHP Kullanım İstatistikleri.....	6
2.1.5 MySQL Kullanım İstatistikleri	8
2.2 PHP VE MYSQL TABANLI UYGULAMANIN PERFORMANSI.....	9
2.2.1 PHP Performans Çözümleri.....	11
2.2.1.1 Kod performans incelemeleri	11
2.2.1.2 Önbellekleme mekanizmaları	12
2.2.1.2.1 İşlem kodu önbellekleme mekanizmaları.....	12
2.2.1.2.1.1 OPcache	13
2.2.1.2.1.2 XCache.....	14
2.2.1.2.2 Memory caching mekanizmaları	14
2.2.1.2.2.1 Memcached.....	15
2.2.2 Veri Tabanı Performans Çözümleri.....	17
2.2.2.1 MySQL sorgu performansı incelemeleri	17
2.2.2.2 NoSQL	18
2.2.2.2.1 Redis	20
2.2.2.2.2 MongoDB	21
2.2.2.3 Arama motoru sunucuları	22
2.2.2.3.1 Sphinx.....	23
2.2.2.3.2 Elasticsearch	24
2.2.3 Sunucu Performans Çözümleri.....	26

2.2.3.1 Web sunucu yazılımları.....	26
2.2.3.1.1 Apache HTTP sunucusu	26
2.2.3.1.2 Nginx.....	27
2.2.3.2 PHP işleyiciler (PHP handlers).....	29
2.2.3.3 PHP yorumlayıcılar (PHP interpreters)	30
2.2.3.3.1 HipHop Virtual Machine (HHVM)	30
2.2.3.4 Web sunucu hızlandırıcıları.....	31
2.2.3.4.1 Varnish Cache	31
3. VERİ VE YÖNTEM	33
3.1 PERFORMANS TEST ARAÇLARI	35
3.1.1 ab.....	35
3.1.2 Apache JMeter	36
3.1.3 XDebug.....	37
3.1.4 mysqlslap.....	38
4. BULGULAR	39
4.1 PHP PERFORMANS TESTLERİ	39
4.1.1 PHP Kod Performans Testleri	39
4.1.1.1 strstr ve strreplace fonksiyonları.....	39
4.1.1.2 “.” (nokta) ve “,” (virgül) string birleştirme operatörleri.....	40
4.1.1.3 " (çift tırnak) ve ' (tek tırnak) operatörleri.....	41
4.1.1.4 require, require_once, include, include_once fonksiyonları.....	42
4.1.1.5 for döngü uzunluğunun döngü içerisinde belirlenmesi.....	43
4.1.1.6 Dizi elemanlarına döngüler ile erişim (for, foreach, while).....	44
4.1.1.7 if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörü	45
4.1.1.8 Dizi elemanlarına anahtar ile erişim.....	47
4.1.1.9 isset ve strlen fonksiyonları.....	48
4.1.1.10 Nesne elemanına doğrudan veya metot yardımı (Getter) ile erişim.....	49

4.1.2 Önbellek Mekanizmaları Testleri.....	51
4.1.2.1 İşlem kodu önbellekleme (opcode caching)	
mekanizmaları.....	51
4.1.2.1.1 OPcache.....	51
4.1.2.1.2 XCache	53
4.1.2.2 Memory caching mekanizmaları	56
4.1.2.2.1 Memcached	56
4.2 VERİ TABANI PERFORMANS TESTLERİ	57
4.2.1 MySQL Sorgu Performans İncelemeleri.....	57
4.2.1.1 SELECT sorgularında sütun kısıtlaması.....	57
4.2.1.2 JOIN kullanımı	59
4.2.1.3 INNER JOIN – LEFT JOIN kullanımı	60
4.2.1.4 LIMIT kullanımı.....	61
4.2.2 NoSQL Performans Testleri.....	63
4.2.2.1 Redis	63
4.2.2.2 MongoDB.....	64
4.2.3 Arama Motoru Sunucuları.....	67
4.2.3.1 Sphinx	67
4.2.3.2 Elasticsearch.....	68
4.3 WEB SUNUCU PERFORMANSI	69
4.3.1 Web Sunucu Yazılımları Performansı	69
4.3.2 PHP İşleyici Performansı	70
4.3.3 PHP Yorumlayıcı Performansı	71
4.3.4 Web Sunucu Hızlandırıcıları Performansı	71
4.3.4.1 Varnish Cache	71
5. TARTIŞMA	74
6. SONUÇ.....	79
KAYNAKÇA	80
ÖZGEÇMİŞ.....	84

TABLULAR

Tablo 2.1: Sunucu taraflı programlama dillerinin web sitelerinde kullanım oranları.....	6
Tablo 2.2: TIOBE Index programlama dilleri kullanım oranları.....	7
Tablo 2.3: Veri tabanı yönetim sistemleri popülerlik sıralaması.....	9
Tablo 2.4: Elasticsearch yapısal terimlerinin SQL ve MongoDB'deki karşılıkları.....	25
Tablo 3.1: Test işlemlerinin yapılacağı bilgisayarın donanımsal özellikleri.....	33
Tablo 3.2: Hata ayıklama ve kod analiz işlemlerinin yapılacağı sistem bilgileri.....	33
Tablo 3.3: Test işlemlerinin yapıldığı sunucuda, PHP'nin phpinfo() fonksiyonu yardımıyla alınan sistem bilgileri.....	34
Tablo 4.1: strtr ve strreplace fonksiyonlarının Xdebug Profiler sonuçları.....	39
Tablo 4.2: strtr ve strreplace fonksiyonlarının JMeter sonuç ortalaması.....	39
Tablo 4.3: strtr ve strreplace fonksiyonlarının ab sonuç ortalaması.....	40
Tablo 4.4: "." (nokta) ve ";" (virgül) string birleştirme operatörlerinin Xdebug Profiler sonuçları.....	40
Tablo 4.5: "." ve ";" string birleştirme operatörlerinin JMeter sonuç ortalaması.....	40
Tablo 4.6: "." ve ";" string birleştirme operatörlerinin ab sonuç ortalaması.....	40
Tablo 4.7: " (çift tırnak) ve ' (tek tırnak) operatörlerinin Xdebug Profiler sonuçları.....	41
Tablo 4.8: " (çift tırnak) ve ' (tek tırnak) operatörlerinin JMeter sonuç ortalaması.....	42
Tablo 4.9: " (çift tırnak) ve ' (tek tırnak) operatörlerinin ab sonuç ortalaması.....	42
Tablo 4.10: include, include_once, require ve require_once fonksiyonlarının JMeter yanıt süresi ortalaması.....	42
Tablo 4.11: include, include_once, require ve require_once fonksiyonlarının ab yanıt süresi ortalaması.....	43
Tablo 4.12: for döngüsü uzunluğunun döngü içerisinde ve döngü dışarısında hesaplandığı dosyaların Xdebug Profiler sonuçları.....	44
Tablo 4.13: for döngüsü uzunluğunun döngü içerisinde ve döngü dışarısında hesaplandığı dosyaların JMeter yanıt süresi ortalaması.....	44
Tablo 4.14: for döngüsü uzunluğunun döngü içerisinde ve döngü dışarısında hesaplandığı dosyaların ab yanıt süresi.....	44
Tablo 4.15: for, while ve foreach fonksiyonlarının Xdebug Profiler sonuçları.....	45
Tablo 4.16: for, while ve foreach fonksiyonlarının JMeter yanıt süresi ortalaması.....	45

Tablo 4.17: for, while ve foreach fonksiyonlarının ab yanıt süresi ortalaması.....	45
Tablo 4.18: if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörlerinin Xdebug Profiler sonuçları	46
Tablo 4.19: if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörlerinin JMeter yanıt süresi ortalaması	46
Tablo 4.20: if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörlerinin ab yanıt süresi ortalaması	46
Tablo 4.21: Dizi elemanlarına anahtar ile erişimde Xdebug Profiler sonuçları.....	47
Tablo 4.22: Dizi elemanlarına anahtar ile erişimde JMeter yanıt süresi ortalaması	47
Tablo 4.23: Dizi elemanlarına anahtar ile erişimde ab yanıt süresi ortalaması	48
Tablo 4.24: isset ve strlen fonksiyonlarının Xdebug Profiler sonuçları.....	48
Tablo 4.25: isset ve strlen fonksiyonlarının JMeter yanıt süresi ortalaması	49
Tablo 4.26: isset ve strlen fonksiyonlarının ab yanıt süresi ortalaması	49
Tablo 4.27: Nesne elemanına doğrudan veya metot yardımı ile erişimde Xdebug Profiler sonuçları	50
Tablo 4.28: Nesne elemanına doğrudan veya metot yardımı ile erişimde JMeter yanıt süresi ortalaması.....	50
Tablo 4.29: Nesne elemanına doğrudan veya metot yardımı ile erişimde ab yanıt süresi ortalaması	50
Tablo 4.30: Opcache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin JMeter yanıt süresi ortalaması	52
Tablo 4.31: Opcache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalaması	52
Tablo 4.32: Opcache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin ab yanıt süresi ortalaması	53
Tablo 4.33: Opcache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin ab yanıt süresi ortalaması	53
Tablo 4.34: XCache ayarlarının phpinfo() çıktısı	54
Tablo 4.35: XCache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin JMeter yanıt süresi ortalaması.....	54
Tablo 4.36: XCache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalaması	55

Tablo 4.37: XCache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin ab yanıt süresi ortalaması	55
Tablo 4.38: XCache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin ab yanıt süresi ortalaması	55
Tablo 4.39: Sunucu üzerindeki Memcached uygulamasının özellikleri	56
Tablo 4.40: MySQL ve Memcached ile veri çekilen sayfaların JMeter yanıt süresi ortalaması	56
Tablo 4.41: MySQL ve Memcached ile veri çekilen sayfaların ab yanıt süresi ortalaması	57
Tablo 4.42: Tablodaki tüm sütunlardan veri çekilen SELECT sorgusunun mysqlslap çalışma süresi	58
Tablo 4.43: Tablodaki tek sütundan veri çekilen SELECT sorgusunun mysqlslap çalışma süresi	58
Tablo 4.44: INNER JOIN kullanılarak ve kullanılmadan veri çekilen sayfaların ab yanıt süresi ortalaması	60
Tablo 4.45: INNER JOIN ile tabloları birleştiren SELECT sorgusunun mysqlslap çalışma süresi	61
Tablo 4.46: LEFT JOIN ile tabloları birleştiren SELECT sorgusunun mysqlslap çalışma süresi	61
Tablo 4.47: LIMIT kullanılmayan SELECT sorgusunun mysqlslap çalışma süresi	62
Tablo 4.48: LIMIT kullanılan SELECT sorgusunun mysqlslap çalışma süresi	62
Tablo 4.49: Sunucu üzerindeki Redis uygulamasının özellikleri.....	63
Tablo 4.50: MySQL ve Redis ile veri çekilen sayfaların JMeter yanıt süresi ortalaması	63
Tablo 4.51: MySQL ve Redis ile veri çekilen sayfaların ab yanıt süresi ortalaması	64
Tablo 4.52: articles tablosu yapısı.....	64
Tablo 4.53: MySQL ve MongoDB’de INSERT işlemlerinin ab yanıt süresi ortalaması	65
Tablo 4.54: MySQL ve MongoDB’de SELECT işlemlerinin ab yanıt süresi ortalaması	65
Tablo 4.55: MySQL ve MongoDB’de UPDATE işlemlerinin ab yanıt süresi ortalaması	65

Tablo 4.56: MySQL ve MongoDB’de büyük boyutlu verinin INSERT işlemlerinin ab yanıt süresi ortalaması	66
Tablo 4.57: MySQL ve MongoDB’de büyük boyutlu verilerin SELECT işlemlerinin ab yanıt süresi ortalaması	66
Tablo 4.58: MySQL ve MongoDB’de büyük boyutlu verilerin UPDATE işlemlerinin ab yanıt süresi ortalaması	66
Tablo 4.59: Sunucu üzerindeki Sphinx modülünün özellikleri.....	67
Tablo 4.60: MySQL ve Sphinx’te veri çekme işlemlerinin JMeter yanıt süresi ortalaması	67
Tablo 4.61: MySQL ve Sphinx’te veri çekme işlemlerinin ab yanıt süresi ortalaması	68
Tablo 4.62: Sunucu üzerindeki Elasticsearch modülünün özellikleri.....	68
Tablo 4.63: MySQL ve Elasticsearch’te veri çekme işlemlerinin JMeter yanıt süresi ortalaması.....	69
Tablo 4.64: MySQL ve Elasticsearch’te veri çekme işlemlerinin ab yanıt süresi ortalaması	69
Tablo 4.65: Apache + PHP-FPM ve Nginx + PHP-FPM tabanlı sunucuda sayfaya gönderilen isteğin ab yanıt süresi ortalaması	70
Tablo 4.66: Apache + DSO ve Apache + PHP-FPM tabanlı sunucuda sayfaya gönderilen isteğin ab yanıt süresi ortalaması	70
Tablo 4.67: Apache + Zend Engine + PHP-FPM ve Apache + FCGI + HHVM tabanlı sunucuda sayfaya gönderilen isteğin ab yanıt süresi ortalaması	71
Tablo 4.68: Varnish Cache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin JMeter yanıt süresi ortalamaları	72
Tablo 4.69: Varnish Cache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalamaları	73
Tablo 4.70: Varnish Cache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin ab yanıt süresi ortalamaları.....	73
Tablo 4.71: Varnish Cache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalamaları	73
Tablo 5.1: Tez çalışmasında incelenen durumların, PHP ve MySQL tabanlı uygulamalarda performansa etkisi	78

ŞEKİLLER

Şekil 2.1: PHP tabanlı bir uygulamanın altyapısı ve işleyiş şeması	4
Şekil 2.2: Memcached kullanan bir sunucuya gelen istek sonrası verinin çekilme aşamaları	16
Şekil 2.3: Verinin Memcached'e kayıt aşamaları	16



KISALTMALAR

ab	:	ApacheBench
ACID	:	Atomicity, Consistency, Isolation, Durability
API	:	Application Programming Interface
BSON	:	Binary JSON
CGI	:	Common Gateway Interface
CPU	:	Central Processing Unit
DDOS	:	Distributed Denial of Service attack
DSO	:	Dynamic Shared Object
FCGI	:	FastCGI
FTP	:	File Transfer Protocol
GB	:	Gigabayt
GPL	:	GNU General Public License
HHVM	:	HipHop Virtual Machine
HTML	:	Hypertext Markup Language
HTTP	:	Hyper-Text Transfer Protocol
httpd	:	HTTP daemon
ICNC	:	International Conference on Natural Computation
IP	:	Internet Protocol
IT	:	Information Technology
JDBC	:	Java Database Connectivity

JSON	:	JavaScript Object Notation
MB	:	Megabayt
ms	:	Milisanıye
NFS	:	Network File System
P.K.	:	Primary Key
PHP-FPM	:	PHP FastCGI Process Manager
phpa	:	PHP Accelerator
RAM	:	Random Access Memory
RESTful	:	Representational State Transfer
SCGI	:	Simple Common Gateway Interface
SMTP	:	Simple Mail Transfer Protocol
SOAP	:	Simple Object Access Protocol
SQL	:	Structured Query Language
SSL	:	Secure Sockets Layer
SuPHP	:	Single User PHP
TB	:	Terabayt
TCP	:	Transmission Control Protocol
TLS	:	Transport Layer Security
VCL	:	Varnish Configuration Language
VMODs	:	Varnish Modules
VTYS	:	Veri Tabanı Yönetim Sistemleri
XML	:	Extensible Markup Language

1. GİRİŞ

Bilişim teknolojilerinin, hayatımızın her alanında yer aldığı günümüzde; özellikle web ve mobil teknolojileri hızla gelişmektedir. Akıllı telefon, tablet gibi mobil cihazların kullanımının artması; buna paralel olarak günlük yaşantımızın her anında internet kullanımını da artırmaktadır. Bu artış, internet sektörüne olan yatırımların artmasını ve sektörün her geçen gün yenilenen teknolojiyle daha hızlı bir ivme yakalamasını sağlamıştır.

Sosyal medya, haber, video, alışveriş ve bankacılık alanında hizmet veren web siteleri, internet kullanımının artmasıyla birlikte en çok kullanılan web siteleri haline gelmiştir. Bu web sitelerinin veri tabanı boyutları, ziyaretçi sayısı ve kullanıcının sitede geçirdiği sürenin artmasıyla birlikte kaynak tüketimleri artmış, aynı şekilde performans ve maliyet problemleri ortaya çıkmıştır. Firmalar; bu performansı en üst düzeye çıkarıp, maliyeti de minimum seviyelerde tutabilmek adına, sistemlerini sürekli olarak geliştirme çalışmalarına girmişlerdir. Bu duruma paralel olarak gerek yazılımsal, gerek donanımsal farklı teknolojik çözümler üretilmiş, üretilmeye de devam edilmektedir.

Bu tez çalışmasında, günümüzde popüler olarak kullanılan, sunucu taraflı web programlama dillerinden biri olan PHP ve birlikte en çok tercih edilen veri tabanı yönetim sistemi olan MySQL tabanlı web uygulamalarında performans incelemeleri yapılacaktır. Teze konu olan bu tip uygulamalar, ağırlıklı olarak Linux tabanlı sunucuda barınmakta olup, Apache web sunucusu üzerinde çalışmaktadır. Tamamen açık kaynak olan bu programlama dili, veri tabanı yönetim sistemi, işletim sistemi ve web sunucusunun dünyadaki kullanım oranı yüksek seviyelerde olduğu için sürekli olarak geliştirme ve sistemi iyileştirme çalışmaları devam etmektedir. Yüksek ziyaretçi alan ve sunucu kaynak tüketimi fazla olan bu tür web uygulamalarında kod tabanlı optimizasyon, PHP önbellekleme mekanizmaları, MySQL sorgu optimizasyonu, NoSQL uygulamaları, üçüncü parti arama motorları, web sunucu optimizasyonu ve web hızlandırıcılar ile ne düzeyde performans artışı sağlanacağı araştırılacaktır.

Bu çalışmaya konu olan web uygulamalarında, çoğu web uygulamasının temel sorunu olan MySQL sorguları ve önbellekleme metotlarının kullanılmaması kaynak tüketimini arttıran en önemli etkenler olarak değerlendirilecektir. Çalışmada bu sorunla ilgili olarak; MySQL'e alternatif olabilecek durumlarda NoSQL kullanımı ve kullanılması mümkün olan durumlarda oldukça başarılı sonuçlar verebilen önbellekleme mekanizmaları incelenecektir. Bunun yanı sıra sıkça kullanılan PHP fonksiyonlarının ve programlama esnasında yapılan hataların performansa etkisi araştırılacaktır.

Performans ölçüm işlemlerinde; istek yapılan sayfanın yanıt süresini ölçmek amacıyla Apache JMeter, ApacheBench (ab), PHP kodları ile dosyaların çalışma sürecini analiz etmek amacıyla Xdebug ve MySQL sorgularının yanıt süresini ölçmek amacıyla mysqlslap uygulamaları kullanılacaktır. JMeter ve ab; belirlenen anlık ve toplam istek sayısı simülasyonlarıyla belirli yük altında uygulamanın performansının ölçümü için oldukça kullanışlı uygulamalardır. Aynı şekilde mysqlslap uygulamasında da MySQL sorgusu; anlık ve toplam istek sayısı simülasyonu ile çalıştırılabilmektedir. Xdebug ise PHP tabanlı uygulamalarda kodların çalışma süresini ve analizini yapmak için kullanılan popüler bir uygulamadır.

Bu çalışma; ülkemizde, PHP tabanlı uygulamalarda performans çözümleri konusunda akademik kaynak eksiklikleri ve özel sektörde PHP tabanlı web uygulaması bulunan birçok firmanın karşılaştığı performans ve uygulama maliyet sorunları göz önünde bulundurularak gerçekleştirilmiştir. Çalışma sırasında uygulamalarda darboğaza neden olan durumlar için farklı çözüm yöntemleri test edilecek ve uygulanan yöntemlerin bu durumlarda performansa ne oranda katkıda bulunacağını tespit edilecektir.

2. LİTERATÜR TARAMASI

2.1 PHP VE MYSQL

PHP; sunucu taraflı çalışan, öncelikli amacı web geliştirme olan ve Hypertext Markup Language (HTML) içerisine gömülebilen programlama dilidir. İlk olarak “Personal Home Page” kelimesinden türetilmiş olan bu dilin günümüzdeki açılımı “PHP: Hypertext Preprocessor” olarak yapılmaktadır. 1994 yılında Rasmus Lerdorf tarafından geliştirilen PHP’nin, 3 Aralık 2015 tarihinde Zend 3.0 motoru ile yeni sürüm olan PHP 7.0.0 resmi olarak yayınlanmıştır.¹

MySQL; Oracle tarafından geliştirilen ve dağıtılan, açık kaynak, çoklu iş parçacıklı ve çok kullanıcı, ilişkisel Structured Query Language (SQL) veri tabanı yönetim sistemidir. Hurrison (2006), MySQL üzerine yazdığı kitabında ilk olarak bir İsveç firması olan MySQL AB firması tarafından oluşturulmuş olup, David Axmark, Allan Larsson ve Michael "Monty" Widenius tarafından bulunduğunu belirtmiştir. İlk sürümü 1994 yılında yayınlanan MySQL’in; Aralık 2015 itibariyle en son sürümü olan 5.7.10, Aralık 2015’te yayınlanmıştır.²

2.1.1 PHP’nin Çalışma Prensipleri

PHP sunucu tarafında çalışan bir programlama dilidir. İstemci tarafından PHP tabanlı bir web sayfasına yapılan istek aşağıdaki aşamalardan geçmektedir.

- i. İstemci tarafından web sunucusuna yapılan istek sonrasında, web sunucusu ilgili dosyayı algılar.
- ii. Dosya eğer statik bir HTML dosyasıysa, web sunucusu tarafından istemciye doğrudan dönüş yapılır fakat eğer dinamik bir PHP dosyasıysa söz konusu dosyanın çalıştırılması için PHP yorumlayıcısına iletilir.

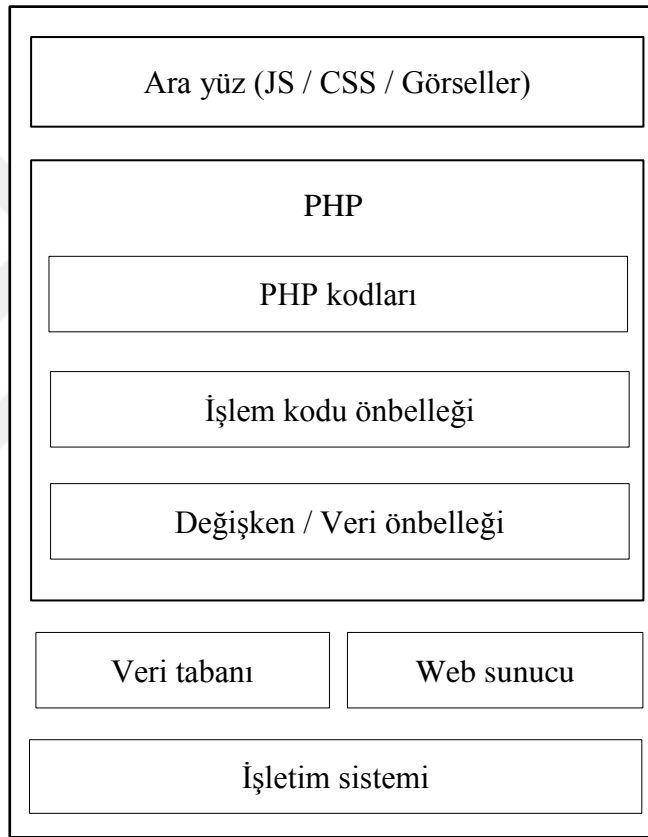
¹ PHP, History of PHP, <http://php.net/manual/tr/history.php.php> [erişim tarihi 22.12.2015].

² MySQL 5.7 Release Notes, <https://dev.mysql.com/doc/relnotes/mysql/5.7/en/> [erişim tarihi 25.12.2015].

- iii. PHP yorumlayıcısı, ilgili dosyada yer alan PHP kodlarını çalıştırarak web sunucusuna sonucu gönderir.
- iv. Web sunucusuna gönderilen bu sonuç, istemci tarafına iletilir.

PHP tabanlı bir web uygulamasının, altyapı katmanları ve sistemin işleyişi Şekil 2.1’de belirtilmiştir.

Şekil 2.1: PHP tabanlı bir uygulamanın altyapısı ve işleyiş şeması



Kaynak: Padilla, A. & Hawkins, T. (2010). Pro PHP Application Performance: Tuning PHP Web Projects for Maximum Performance.

2.1.2 PHP Programlama Dilinin Avantajları

PHP programlama dilini, diğer web yazılım dillerinden farklı kılan ve daha popüler olmasını sağlayan başlıca özellikleri mevcuttur. Valade (2010); PHP'nin avantajlarını bir sonraki sayfada yer alan maddeler ile açıklamıştır.

- i. Ücretsizdir.
- ii. Birçok işletim sistemi üzerinde çalışabilmektedir.
- iii. Çok geniş bir teknik destek topluluğuna ve dokümantasyona sahiptir.
- iv. Birçok veri tabanını desteklemektedir.
- v. Çoğu web barındırma paketlerinde hazır ve kullanılabilir durumda sunulmaktadır.
- vi. Açık kaynak olması nedeniyle; yazılımcıların ihtiyaçlarına göre düzenlemeler yapmasına ve yeni özellikler eklemesine izin vermektedir.

2.1.3 MySQL Veri Tabanı Yönetim Sisteminin Avantajları

PHP tabanlı uygulamalarda en çok tercih edilen veri tabanı sistemlerinden olan MySQL, sahip olduğu bazı özellikleri nedeniyle diğer veri tabanı yönetim sistemlerinden ayrılmaktadır. Valade (2010); MySQL veri tabanının avantajlarını aşağıdaki şekilde belirtmiştir.

- i. GNU General Public License (GPL) altında kullanımı ücretsizdir, ticari kullanımlarda ise diğer büyük veri tabanı sistemlerine göre makul ücretler istenmektedir.
- ii. Kullanımı kolaydır. İlişkisel veri tabanı yönetim sistemleriyle iletişim için standart dil olan SQL dilindeki birkaç basit ifadeyi kullanarak MySQL veri tabanını oluşturabilir ve etkileşime geçilebilir.
- iii. Birçok işletim sistemi üzerinde çalışabilmektedir.
- iv. Neredeyse bütün web barındırma paketlerinde hazır ve kullanılabilir durumdadır. Çoğu barındırma firmasından alınacak web barındırma hizmetinde, önceden ücretsiz olarak kurulmuş şekilde sunulmaktadır.
- v. Çok geniş bir teknik destek topluluğuna sahiptir.
- vi. Güvenlidir. Belirli bir kullanıcıya ya da gruba, istenilen ayrıcalıkların verilebilmesini sağlayan esnek bir yetkilendirme sistemine sahiptir. Kullanıcılara ait parolalar özel bir yöntemle şifrelenmektedir.
- vii. Büyük veri tabanlarını desteklemektedir. Her bir tablo için varsayılan dosya büyüklüğü 4 gigabayt (GB)'tır; fakat bu değer işletim sisteminin de

desteklemesi durumunda teorik limit olan 8 milyon terabayta (TB) yükseltilebilmektedir.

- viii. GPL açık kaynak lisansı sayesinde, yazılımcılar tarafından özelleştirilebilmektedir.

2.1.4 PHP Kullanım İstatistikleri

İngiltere merkezli internet hizmetleri şirketi Netcraft'ın 2013 yılının Ocak ayında yapmış olduğu araştırmada; 2,1 milyon Internet Protocol (IP) adresinde bulunan 244 milyon web sitesinde PHP altyapısı kullanıldığı belirtilmektedir.³ Bunun yanı sıra, web tabanlı hizmetler konusunda araştırma ve anket yapan w3techs.com web sitesinin 25 Ocak 2016 tarihli sunucu taraflı programlama dillerinin web sitelerinde kullanım oranı araştırması Tablo 2.1'de yer almaktadır. Tablonun hazırlanmasında, bir web sitesinin birden fazla programlama dili kullanabileceği durumu da göz önünde bulundurulmuştur.

Tablo 2.1: Sunucu taraflı programlama dillerinin web sitelerinde kullanım oranları

Programlama Dili	Kullanım Oranı
PHP	81.7%
ASP.NET	16.0%
Java	3.0%
Statik sayfalar	1.6%
ColdFusion	0.7%
Ruby	0.6%
Perl	0.5%
JavaScript	0.2%
Python	0.2%
Erlang	0.1%
Miva Script	0.1%

Kaynak: W3Techs, http://w3techs.com/technologies/overview/programming_language/all [erişim tarihi 25.01.2016].

³ PHP, Usage Stats for January 2013, <http://php.net/usage.php> [erişim tarihi 25.01.2016].

Hollanda merkezli, yazılım kalitesi alanında hizmet veren TIOBE Software BV şirketinin Ocak 2016 için hazırlanmış olduğu en popüler programlama dilleri listesinde, PHP; altıncı sırada yer almaktadır. Her ay düzenli olarak güncellenen bu listedeki derecelendirmeler, dünya çapındaki mühendislerin, kursların ve üçüncü parti sağlayıcıların sayısına dayanmaktadır.⁴ Kullanım oranlarının hesaplanması aşamasında Google, Bing, Yahoo, Wikipedia, Amazon gibi popüler sitelerden yararlanılan liste; Tablo 2.2’de yer almaktadır.

Tablo 2.2: TIOBE Index programlama dilleri kullanım oranları

Sıra	Programlama Dili	Kullanım Oranı
1	Java	% 21.465
2	C	% 16.036
3	C++	% 6.914
4	C#	% 4.707
5	Python	% 3.854
6	PHP	% 2.706
7	Visual Basic .NET	% 2.582
8	JavaScript	% 2.565
9	Assembly language	% 2.095
10	Ruby	% 2.047
11	Perl	% 1.841
12	Delphi/Object Pascal	% 1.786
13	Visual Basic	% 1.684
14	Swift	% 1.363
15	MATLAB	% 1.228
16	Pascal	% 1.194
17	Groovy	% 1.182
18	Objective-C	% 1.074
19	R	% 1.054
20	PL/SQL	% 1.016

Kaynak: TIOBE, TIOBE Index,
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html> [erişim tarihi 14.01.2016].

⁴ TIOBE, TIOBE Programming Community Index Definition,
http://www.tiobe.com/tiobe_index?page=programminglanguages_definition [erişim tarihi 14.04.2016].

2.1.5 MySQL Kullanım İstatistikleri

Dünyada en çok kullanılan veri tabanı yönetim sistemlerinden biri olan MySQL'in kullanım oranıyla ilgili detaylı çalışmalardan biri; Avusturyalı bilgi teknolojileri danışmanlık firması olan "solid IT" firmasına ait DB-Engines.com projesiyle gerçekleştirilmektedir. Aylık olarak güncellenen ve popülerlik sıralamasına göre oluşturulan listede, MySQL; 2016 yılı Ocak ayının en popüler ikinci veri tabanı yönetim sistemi olarak yer almaktadır. 20 farklı yapının yer aldığı liste Tablo 2.3'te yer almaktadır. Listedeki sıralama puanlarının hesaplanması; farklı parametreler baz alınarak yapılmaktadır. Bu popülerlik puanını hesaplama aşamasında aşağıda yer alan faktörler kullanılmaktadır.⁵

- i. Veri Tabanı Yönetim Sistemleri (VTYS) ile ilgili web sitelerinde yer alan içeriklerin sayısı ile Google ve Bing arama motoru sorgularındaki sonuçların sayısı,
- ii. Google Trend sorguları ile VTYS'nin aranma sıklığı,
- iii. VTYS hakkında popüler Information Technology (IT) soru-cevap platformlarındaki tartışma sayısı,
- iv. VTYS ile ilgili iş fırsatları,
- v. Twitter ve LinkedIn gibi sosyal ağlarda, söz konusu VTYS'ne olan ilgi.

⁵ solid IT, Method of calculating the scores of the DB-Engines Ranking, http://db-engines.com/en/ranking_definition [erişim tarihi 16.01.2016].

Tablo 2.3: Veri tabanı yönetim sistemleri popülerlik sıralaması

Sıra	VTYS	Veri Tabanı Modeli	Puan
1.	Oracle	İlişkisel VTYS	1496.08
2.	MySQL	İlişkisel VTYS	1299.26
3.	Microsoft SQL Server	İlişkisel VTYS	1144.06
4.	MongoDB	Doküman deposu	306.03
5.	PostgreSQL	İlişkisel VTYS	282.40
6.	DB2	İlişkisel VTYS	196.37
7.	Microsoft Access	İlişkisel VTYS	134.04
8.	Cassandra	Geniş sütun deposu	130.95
9.	SQLite	İlişkisel VTYS	103.74
10.	Redis	Anahtar-değer deposu	101.16
11.	SAP Adaptive Server	İlişkisel VTYS	83.18
12.	Elasticsearch	Arama motoru	77.21
13.	Solr	Arama motoru	75.39
14.	Teradata	İlişkisel VTYS	74.95
15.	Hive	İlişkisel VTYS	53.58
16.	HBase	Geniş sütun deposu	53.37
17.	FileMaker	İlişkisel VTYS	48.83
18.	Splunk	Arama motoru	43.12
19.	SAP HANA	İlişkisel VTYS	38.61
20.	Informix	İlişkisel VTYS	34.88

Kaynak: solid IT, DB-Engines Ranking, <http://db-engines.com/en/ranking> [erişim tarihi 16.01.2016].

2.2 PHP VE MYSQL TABANLI UYGULAMANIN PERFORMANSI

PHP tabanlı uygulamalarda, performansı etkileyecek birçok faktör mevcuttur. Eski sürüm kullanımı, düşük performanslı kodlama, yüksek trafik, çok büyük verilerle çalışmak bu konuda sıkça karşılaşılan durumlardandır.

Belirli aralıklarla güncellenen ve her yeni güncellemede performansı arttırılan PHP'nin en stabil sürümünün kullanılması performans açısından oldukça önemlidir. PHP'nin resmi web sitesinde yer alan açıklamada; Aralık 2015'te yayınlanan PHP 7 sürümünün, kullandığı Zend Engine 3 yorumlayıcısı ile bir önceki sürüm olan PHP 5.6'ya göre iki kat daha hızlı olduğu belirtilmiştir.⁶

⁶ PHP, News Archive – 2015, <http://php.net/archive/2015.php> [erişim tarihi 16.12.2015]

Uygulamanın kaynak kodlarının, uygun bir şekilde yazılmamasının da performans açısından olumsuz sonuçları olmaktadır. Fazla kaynak tüketimine neden olabilecek şekilde geliştirilen uygulamalarda performansı arttırmak için öncelikli olarak yapılması gereken işlemlerden biri kodların optimize edilmesidir. Kod optimizasyonu konusunda sıkça kullanılan fonksiyon ve operatörler, ilerleyen bölümlerde detaylı olarak incelenecektir.

Standart ayarlarla kurulumu yapılmış olan bir Apache web sunucusunda çalışmakta olan ve Apache'nin varsayılan PHP yorumlayıcısını kullanan uygulamalarda, istemci tarafından yapılan her istekte PHP kodları daha önce aynı istekte bulunmuş olsa bile tekrar tekrar yorumlanmaktadır. Derlenebilen programlama dillerinde, kod bir defa derlendikten sonra, gelen her istekte derlenen bu kod çalıştırılmaktadır. Fakat PHP, Python gibi yorumlanan dillerde ise, ilgili kodlar her istekte tekrar yorumlanmaktadır. Yoğun trafik alan bir web sitesinde, bu durum; uygulamanın performansını olumsuz bir şekilde etkilemektedir. Farklı bir yapıda çalışan PHP yorumlayıcıları, bu duruma çözüm olarak kullanılabilirlerdir.

Yoğun trafik alan PHP uygulamalarında, performansı artıracak bir diğer unsur da, PHP eklentisi olarak hazırlanan PHP hızlandırıcılarıdır. Çoğu PHP hızlandırıcı, gelen her istekte dosyaların çözümlenmesi ve yorumlanması işleminin her defasında yeniden yapılmaması amacıyla, yorumlanıp çalıştırılan PHP dosyalarını makine dili komutunun bir parçası olan işlem kodu (opcode) olarak önbelleğe alarak, bellekte tutmaktadır. Dosyaya gelecek olan bir sonraki istekte, PHP hızlandırıcı, PHP yorumlayıcıdan önce isteği karşılamakta ve dosyanın tekrar yorumlanmadan önce bellekte tutulan işlem kodu olarak çalıştırılmasını sağlamaktadır. Bu işlem sonrasında, hem kaynak tüketimi azaltılmakta hem de istek daha hızlı bir şekilde cevaplanarak uygulamanın performansı artırılmaktadır.

Veri tabanı yönetim sistemi olarak MySQL kullanan bir web sitesi, verilerin büyük boyutlara ulaşması ve yoğun trafik alması durumunda çok ciddi performans sorunlarıyla karşılaşabilmektedir. Böyle durumlarda, ilişkisel veri tabanı yönetim sistemlerine göre daha performanslı çalışabilen NoSQL kullanımına yönelmek sisteme performans

açısından önemli katkılarda bulunacaktır. Milisaniyelerin bile oldukça önem arz ettiği web sitelerinde verinin mümkün olduğu kadar hızlı bir şekilde kullanıcıya iletilmesi gerekmektedir. Bir saniyelik gecikmenin, büyük maddi kayıplara yol açtığı büyük sistemlerde; ağırlıklı olarak daha iyi performans verebilen NoSQL kullanılmaktadır.

Yüksek trafikli web sitelerinde performansa etki eden bir diğer etmen ise, site içerisindeki arama işlemleridir. Çok büyük boyutlu, ilişkisel bir veri tabanına sahip sistemde birden fazla kritere göre detaylı arama yapıldığı düşünüldüğünde; ilgili sonuçların istemciye geç gönderilmesi olağandır. Bu gibi durumlarda, metin indeksleme konusunda ilişkisel veri tabanı yönetim sistemlerine göre daha başarılı olan üçüncü parti arama motorları kullanılmakta ve sonuçlar çok daha hızlı bir şekilde istemciye iletilmektedir.

Tüm bunların yanı sıra, uygulamanın çalıştığı web sunucusunun performansı da, uygulamanın performansını ortaya koyan bir diğer etmendir. Apache web sunucusu üzerinde çalışan bir web uygulaması, yoğun trafik aldığı zamanlarda performans açısından yetersiz kalabilmektedir. Bu gibi durumlarda alternatif web sunucuları ya da web sunucusunun önünde hizmet veren web hızlandırıcılar (HTTP accelerators) kullanılmaktadır.

2.2.1 PHP Performans Çözümleri

2.2.1.1 Kod performans incelemeleri

PHP tabanlı uygulamalarda performans yönetimi konusunda öncelikli olarak yapılması gereken işlemlerin başında “kod optimizasyonu” yer almaktadır. Kod optimizasyonu, uygulamanın daha az kaynak kullanımına bağlı olarak Central Processing Unit (CPU) ve Random Access Memory (RAM) işlem sürelerinin kısılmasını, daha az disk alanı kullanımını sağlamaktadır. Buna paralel olarak, üzerinde çalıştığı sunucu donanımı ne kadar iyi olursa olsun, iyi bir şekilde yazılmamış bir PHP uygulaması düşük performansla çalışacaktır.

PHP uygulamalarının performansını ölçerken yapılması gereken en önemli işlemlerden biri, uygulamanın farklı parçaları için performans ölçümlemeye başlamaktır. Eğer uygulamanın hangi kısımlarının düşük performanslı çalıştığı bilinmezse, kod optimizasyonuna nereden başlanacağına karar vermek zor olacaktır. PHP tabanlı uygulamaların performansını ölçebilmek için birçok ücretli ve ücretsiz araç mevcuttur. Bu tez çalışmasının ilerleyen bölümlerinde performans ölçüm araçları ayrıca incelenecektir.

Farkında olmadan, gereksiz şekilde yazılan her kod performans düşüklüğüne sebep olabilmektedir. Örneğin, sonraki aşamada kullanılmayacak bir değişken oluşturulması ya da oluşturulan veri tabanı bağlantılarının işlemler bittikten sonra kapatılmaması düşük trafikli sitelerde çok büyük performans etkileri göstermese de, yüksek trafik alan sitelerde performansı ciddi bir şekilde düşürebilmektedir.

Padilla ve Hawkins (2010); PHP kodları üzerinde yaptıkları incelemede,

- i. “require” fonksiyonunun yerine “require_once” fonksiyonunun kullanımının,
- ii. 100.000 elemanlı bir dizinin, elemanlarının döngü içerisinde çalıştırılması işleminde, “for” ve “while” döngülerinin yerine “foreach” döngüsünün kullanımının,
- iii. Küçük dosyalar için “fread”, büyük dosyalar için “file_get_contents” fonksiyonunun kullanımının,
- iv. Nesne elemanlarına getter metodu ile erişim yerine, doğrudan erişimin

performans olarak etkili sonuçlar vereceğini belirtmişlerdir.

2.2.1.2 Önbellekleme mekanizmaları

2.2.1.2.1 İşlem kodu önbellekleme mekanizmaları

PHP işlem kodu önbellekleme (opcode caching) mekanizmaları, PHP tabanlı uygulamaların performansını arttırmak amacıyla geliştirilmiş olan araçlardır. Birçok

PHP hızlandırıcı, bir PHP dosyasına gelen her istekte, dosyanın kodlarının tekrar tekrar çözümlenip yorumlanmasının önüne geçmek amacıyla, yorumlanan ve işlem koduna dönüştürülen PHP dosyalarını önbelleğe alarak çalışır. Önbelleğe alınan kodlar paylaşımlı bellekte saklanır ve çalışma anında sabit disk üzerinde daha yavaş işleyen okuma işlemi minimize edilerek doğrudan buradan çalıştırılır.

Bu bölümde, PHP 5.5 sürümüyle birlikte sunulan OPcache mekanizması incelenecektir.

2.2.1.2.1.1 OPcache

OPcache, PHP 5.5 ve sonraki sürümlerle birlikte gelen, Zend Optimizer+ uygulamasından türetilerek geliştirilen bir caching mekanizmasıdır. PHP 5.2, 5.3 ve 5.4 sürümleri için de, PECL kütüphanesinde ZendOpcache adında eklenti paketi bulunmaktadır.⁷ Eklenti için ayrılan bellek, belleğe alınacak toplam dosya sayısı gibi ayarlar, PHP yapılandırma dosyası olan php.ini üzerinden yapılabilmektedir.

OPcache, web tabanlı bazı yönetim ve görüntüleme aracı desteğine de sahiptir. OpCacheGUI, opcache-status gibi araçlar; OPcache eklentisinin kaynak tüketimini inceleme, önbelleğe alınan dosyaları ve bu dosyalara gelen istekleri listeleme, grafiksel raporlamalar ve yapılandırma ayarlarını düzenleme imkânı sunmaktadır.

Uygulama analizi hizmeti sunan AppDynamics firmasının PHP ve MySQL altyapısını kullanan uygulaması üzerinde, Bolton (2014) üç saatlik inceleme gerçekleştirmiştir. Bu inceleme işlemi esnasında; OPcache kullanarak ve kullanılmadan; uygulamanın yanıt süreleri ölçülmüştür ve ölçüm sonuçlarına göre uygulamanın yanıt süresinde yüzde 14 oranında azalma olduğunu tespit etmiştir. Test edilen uygulamanın; giriş, ürün detayı, arama gibi farklı sayfalarında yapılan testler de ise yanıt süresinde yüzde 8 ile yüzde 74 aralığında değişen azalma görüntülenmiştir.⁸

⁷ PHP, OPcache, <http://php.net/manual/tr/book.opcache.php> [erişim tarihi 22.12.2015].

⁸ Bolton, R. 2014, Why Every PHP Application Should Use an OpCache [online], <https://blog.appdynamics.com/php/why-every-php-application-should-use-an-opcache/> [erişim tarihi 23.12.2015].

MySQL sunucusuna gönderilen sorgu yanıtı, üçüncü parti bir uygulamadan gelen yanıt geç geliyorsa ya da uygulama oldukça küçük işlemler içeriyorsa OPcache kullanımı beklenen performansı sağlayamayacaktır. Bu nedenle, OPcache kullanım yerinin doğru bir şekilde analiz edilmesi gerekmektedir.

2.2.1.2.1.2 XCache

XCache, açık kaynak işlem kodu önbellekleme mekanizmasıdır. Diğer işlem kodu önbellekleme mekanizmalarıyla aynı çalışma modeline sahip olan XCache, Lighttpd web sunucusunda çalışan bir geliştirici tarafından geliştirilmiştir. Geliştiricisi, XCache'in geliştirilmesinden önce PHP Accelerator (phpa), Truck MMCache ve APC kullandığını ve bu uygulamaların kaynak kodlarını inceleyerek bu konuda bilgi sahibi olduğunu belirtmektedir. APC ve eAccelerator'e ek modüller dahil etmenin zorluğu ve daha performanslı bir uygulama yapabileceğini düşünmesi nedeniyle bu uygulamayı geliştirmeye karar verdiğini söylemektedir.⁹

XCache'in en son versiyonu olan 3.2.x sürümü, PHP 5.6'ya tam olarak uyumlu olup, stabil bir şekilde çalışmaktadır. Bu mekanizmayı, benzer uygulamalardan ayıran en önemli özelliği ise geliştirilme çalışmalarına devam edilmesidir. Gerekli optimizasyonların yapılması halinde, XCache'in uygulamalarda kullanılmasıyla, sayfa yüklenme hızlarında 5 kata varan performans artışı sağlayabildiği belirtilmektedir. Bu konuda Padilla ve Hawkins (2010), yaptıkları testte, XCache kullanılarak uygulamanın yanıt süresinde yüzde 4 civarında bir azalma sağladığını analiz etmişlerdir.

2.2.1.2.2 Memory caching mekanizmaları

Memory caching, bellekte saklanan verilerin, mikroişlemci tarafından daha çabuk erişilen önbelleğe aktarılması işlemidir. Bu işlemin amacı en basit haliyle, uygulama tarafından sıkça kullanılan verilerin bellek yerine önbellekte saklanarak; uygulamanın daha hızlı çalışmasını sağlamaktır. Mikro işlemci, veriyi işlerken ilk olarak önbelleğe bakar ve eğer aradığını burada bulursa diğer bellek veya veri depolama aygıtlarını

⁹ XCache, Introduction to XCache, 2013, <https://xcache.lighttpd.net/wiki/Introduction> [erişim tarihi 26.12.2015].

okumaz. Bu nedenle, önbelleğe aktarılan veri, bellekteki diğer verilere göre çok daha hızlı okunup, işlenir. Bu durum, uygulamanın hızını önemli bir ölçüde artıran etkenlerden biri olarak kabul edilir.

Bu bölümde, PHP tabanlı uygulamalarda en popüler memory caching mekanizması olan Memcached incelenecektir.

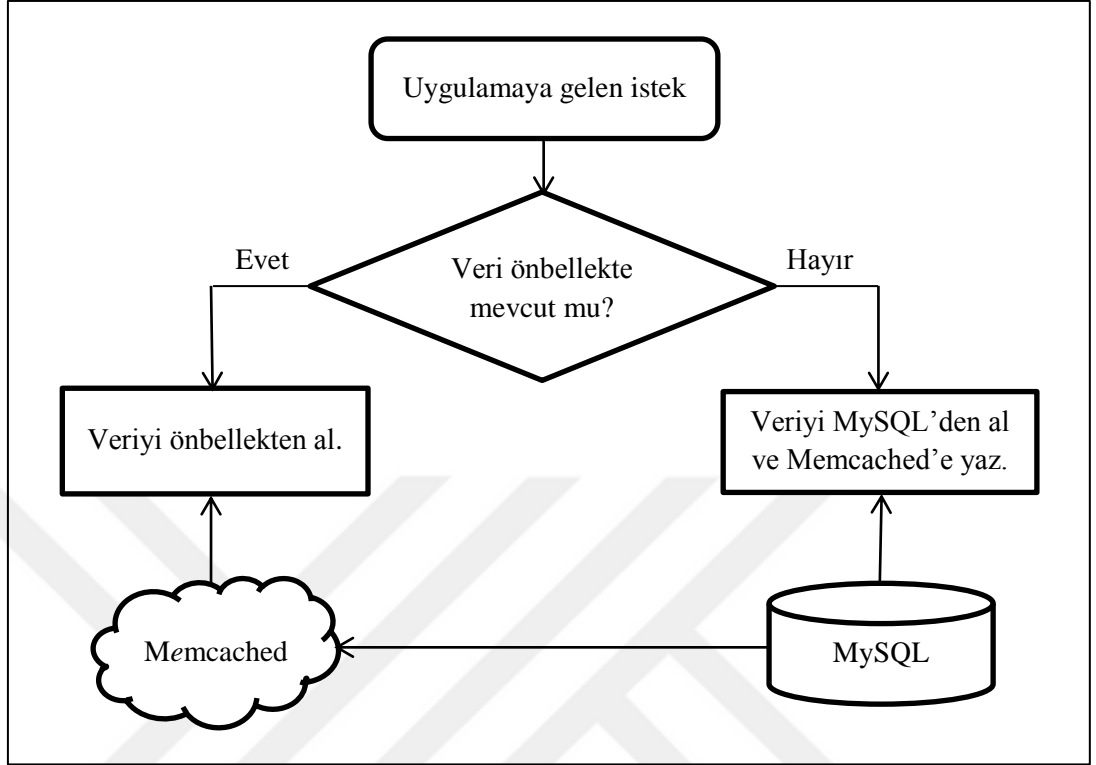
2.2.1.2.2.1 Memcached

Memcached, özellikle veri tabanı sunucusunun yükünü azaltarak web uygulamalarını hızlandırmak amacıyla kullanılan yüksek performanslı bir memory caching sistemidir. 2003 yılında livejournal.com web sitesindeki yükü azaltmak ve performansı arttırmak amacıyla Brad Fitzpatrick tarafından geliştirilen sistem, geliştirildikten bir süre sonra Wikipedia, Flickr gibi büyük web siteleri tarafından da kullanılmaya başlanmıştır.¹⁰

Memcached, sistemde kullanılmayan ve boş durumda olan belleği alarak; ihtiyaç duyulan kısımlarda kullanılmasını sağlamaktadır. Özellikle veri tabanı sunucusundaki yükü azaltmak amacıyla kullanılan Memcached, veriyi belirtilen süre için önbellekte tutar. Bu nedenle, aynı veriye tekrar ihtiyaç duyulduğu anda sonuç veri tabanı sunucusunda değil, doğrudan önbellekte aranır. Curioso ve diğerleri (2010), Memcached kullanan bir web sunucusuna gönderilen bir istek sonrası verinin çekilme ve kayıt aşamalarını Şekil 2.2 ve Şekil 2.3'teki gibi görselleştirmişlerdir.

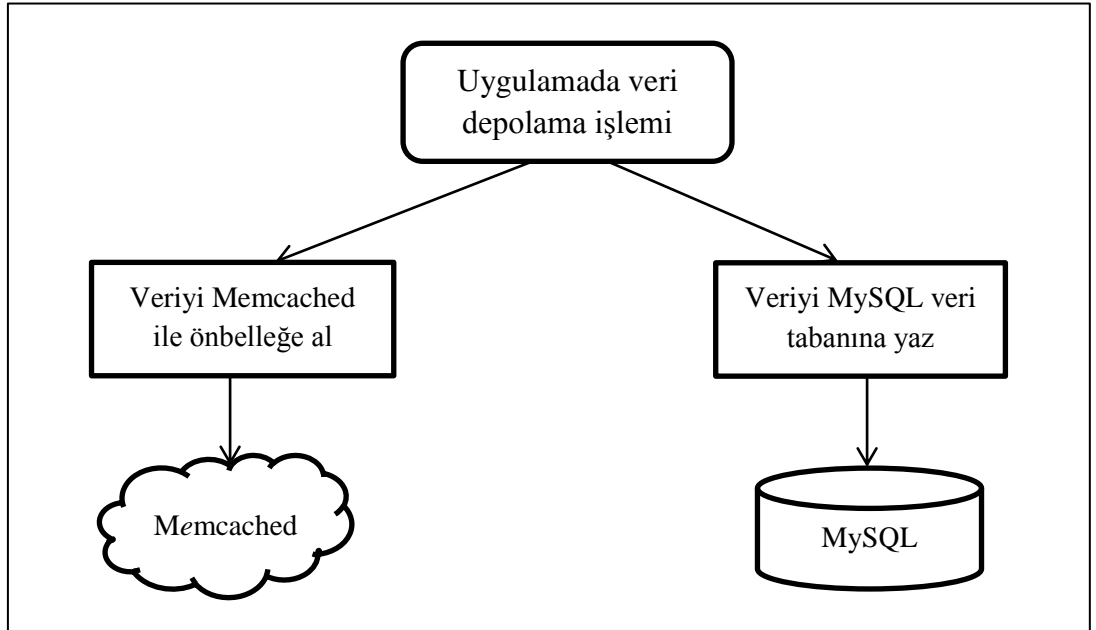
¹⁰ Memcached, About Memcached, <http://memcached.org/about> [erişim tarihi 20.01.2016].

Şekil 2.2: Memcached kullanan bir sunucuya gelen istek sonrası verinin çekilme aşamaları



Kaynak: Curioso, A., Bradford, R. ve Galbraith, P. (2010). *Expert PHP and MySQL*

Şekil 2.3: Verinin Memcached'e kayıt aşamaları



Kaynak: Curioso, A., Bradford, R. ve Galbraith, P. (2010). *Expert PHP and MySQL*

Memcached'in uygulamalarda performansa etkisi konusunda Abu Bakar ve diğeri (2010) yaptıkları çalışmada, sunucunun bir saniyede karşılayabileceği istek sayısındaki ve MySQL ile Apache sunucularındaki CPU kullanımındaki değişimleri incelemişlerdir. Yapılan testlerde; sunucunun bir saniyede karşılayabileceği istek sayısında yüzde 28 oranında artış, MySQL sunucusunun CPU kullanımında yüzde 94'e, Apache web sunucusunda CPU kullanımında yüzde 26'ya varan azalmalar tespit edilmiştir.

Memcached ile ilgili bir diğeri araştırma, Kroth (2014) tarafından PHP tabanlı uzaktan eğitim sistemi olan Moodle üzerinde gerçekleştirilmiştir. Kroth; Moodle'in cache sistemi üzerinde yaptığı çalışmada, ağ dosya sistemini, yerel dosya sistemini, MongoDB, Memcached ve MySQL veri tabanını kullanmıştır. Varsayılan olarak kullanılan Network File System (NFS) cache sistemine göre Memcached yüzde 74'e yaklaşan bir performans artışı sağlamıştır.

2.2.2 Veri Tabanı Performans Çözümleri

PHP ve MySQL tabanlı yüksek trafik alan uygulamalarda en büyük performans sorunları veri tabanı konusunda yaşanmaktadır. MySQL veri tabanı boyutunun büyümesinin, gelen istek sayısının ve buna bağlı olarak anlık sorgu sayısının artmasının sistemin performansı üzerinde oldukça büyük bir rolü bulunmaktadır. Bu bölümde uygulamadaki MySQL sorgularının optimizasyonu, NoSQL sistemleri incelenecektir.

2.2.2.1 MySQL sorgu performansı incelemeleri

MySQL sorgu optimizasyonu ile sunucunun, mevcut sorguyu en kısa sürede tamamlaması ve buna bağlı olarak birim zamanda maksimum sorgu çalıştırabilmesi amaçlanmaktadır. Bunun için ise cevabı en geç gelen sorguların tespit edilmesi gerekmektedir. Eğer bir sorgunun sonucu geç geliyorsa, bu sorgunun çalıştırılmasının öncesinin ve sonrasının değil başlangıç ve bitiş anının ölçümü daha sağlıklı olacaktır.

Büyük sistemlerde, MySQL sunucusunda yavaşlığa neden olacak sorguların manuel olarak tespiti için birkaç farklı çözüm bulunmaktadır. MySQL ayarlarından yavaş

sorguların kaydını tutarak ya da sorgunun kullanıldığı PHP dosyasının XDebug, Xhprof gibi analiz uygulamaları ile incelenmesi bu çözümlerden bazılarıdır. PHP profil uygulamaları; söz konusu SQL sorgusunu, hangi PHP dosyasının hangi satırında yer aldığını, işlem süresi gibi bilgileri ölçümler. Buradan alınacak sonuçlara göre, sistemde iyileştirme yapmak oldukça faydalı sonuçlar alınması sağlayacaktır.

MySQL sorgu analizi ve performans optimizasyonu konusunda Schwartz ve diğerleri (2012); kitaplarında yer verdiği araştırmalarında, bazı sorguların kaynak tüketimi ve yanıt sürelerini incelemişlerdir. NOT EXISTS alt sorgusuyla LEFT OUTER JOIN sorgusunda yapılan performans testinde; LEFT OUTER JOIN sorgusunun yüzde 15 oranında daha hızlı çalıştığını tespit etmişlerdir. Kitapta performans sonuçları incelenmeyen bazı sorgu örnekleri bu tez çalışmasında ele alınacaktır.

2.2.2.2 NoSQL

NoSQL (“Not onyl SQL” ya da “non SQL” olarak ifade edilebilmektedir), ilişkisel veri tabanlarında kullanılan tablo ilişkilerinden farklı bir şekilde modellenen, veri depolama ve okumayı sağlayan bir veri tabanı mekanizmasıdır. Bu yapıya benzer veri tabanları 1960’lı yıllarda da mevcuttu fakat verilerin oldukça büyüdüğü günümüzdeki kadar kullanımı yaygın değildi. Bu yaklaşımın oluşmasına neden olan etkenler; tasarımın basitliği, yatay olarak ölçeklenebilirliği, kolay kullanılabilirliğidir.

İlişkisel veri tabanları; büyük boyutlu verilerin bulunduğu, yoğun trafiği bulunan web sitelerinde, video ve ses yayını yapan uygulamalarda performans konusunda yetersiz kalmıştır. Standart bir ilişkisel veri tabanı yönetim sistemine göre okuma / yazma hızı çok daha iyi olan NoSQL mekanizmaları mevcuttur. Fakat NoSQL sistemlerinin bir çoğunda, ilişkisel veri tabanı yönetim sistemleri işlemlerin (transaction) stabil bir şekilde çalışması ve veri bütünlüğünü sağlamak için Atomicity, Consistency, Isolation, Durability (ACID) kurallarının tamamına uyulmamaktadır. ACID kuralları bir sonraki sayfada açıklanmıştır. Bu kuralların önem arz ettiği sistemlerde (bankacılık, finans vb.) NoSQL çok tercih edilen bir çözüm değildir.

- i. Atomicity (Bölünmezlik): Bir transaction içerisindeki bütün işlemlerin tamamlanması gerekir. Birinin tamamlanmaması durumunda yapılmış olan diğer işlemler geri alınır.
- ii. Consistency (Tutarlılık): Yapılan her işlem tutarlı veri üretilecek şekilde tamamlanmalıdır.
- iii. Isolation (İzolasyon): Her işlemlerin bağımsız bir şekilde işlenmesi ve dışarıdan etkiye kapalı olması durumudur.
- iv. Durability (Dayanıklılık): Tamamlanan bir işlem sonrası değişiklikler diske kalıcı olarak yansıtılır. Olası hatalara karşı esneklik söz konusudur.

NoSQL veri tabanları sınıflandırmak için farklı yaklaşımlar söz konusudur. Temel ve popüler NoSQL veri tabanlarını, veri modeline göre aşağıdaki gibi sınıflandırmak mümkündür.¹¹

- i. Kolon (Sütun) Tabanlı: İlişkisel veri tabanı sistemlerindeki şema yapısına benzer yapıda keyspace adı verilen nesnelerin en düşük düzeydeki halidir. Tuple adı verilen, temel olarak elementlerin sıralanmış bir liste tasarımı yapısına sahiptir. Cassandra, HBase, Druid, Vertica gibi sistemler bu yapıya örnek olarak gösterilebilir.
- ii. Doküman Tabanlı: Bu sistemlerde her kayıt bir doküman olarak isimlendirilmektedir. Dokümanlar genellikle JavaScript Object Notation (JSON) formatında olmaktadır. Apache CouchDB, MongoDB, DocumentDB, Couchbase, OrientDB gibi sistemler bu yapıyı kullanmaktadır.
- iii. Anahtar – değer: Bu sistemlerde anahtara karşılık gelen bir değer mevcuttur. Kolon yapısı bulunmamaktadır. Aerospike, CouchDB, Dynamo, HyperDex, MemcacheDB, Oracle NoSQL Database, Redis, OrientDB bu yapıyı kullanan örnek sistemlerdir.
- iv. Grafik: Bu sistemlerde veriler genellikle ilişkiler üzerinden yönetilir. Node ve relation şeklinde iki temel kavram üzerinden çalışmaktadır. Allegro,

¹¹ NoSQL Databases, <http://nosql-database.org/> [erişim tarihi 28.01.2016].

InfiniteGraph, MarkLogic, Neo4J, OrientDB, Virtuoso gibi sistemler bu yapıya örnek olarak gösterilebilir.

Bu bölümde en çok kullanılan NoSQL yapılarından olan Redis ve MongoDB incelenecektir.

2.2.2.2.1 Redis

Redis, bellek kullanımlı açık kaynak bir anahtar - değer deposudur. Birçok veri yapısını destekleyen, veri tabanı ve önbellek aracı olarak kullanılabilen bir veri yapısı sunucusu olan Redis, ANSI C programlama diliyle yazılmış olup Linux, OSX ve *BSD sistemlerde çalışabilmektedir.¹² Redis, Remote Dictionary Server (uzak sözlük sunucusu) sözcüğünden türetilmiştir.

Redis diğer anahtar – değer veri tabanlarından farklı yapıdaki verileri de desteklemektedir. Metin listeleri (lists), metin kümeleri (sets), dizili metin kümeleri (sorted sets), hash tabloları (hash tables) Redis'in desteklediği diğer yapı tipleridir. Yapılan testlerde 250 milyon adet anahtarı aynı anda depoladığı görülmüş ve bu limitin 2^{32} 'ye kadar çıkabildiği belirtilmiştir.¹³

Redis ile ilgili olarak tecrübelerini bir kitapta toplayan Macedo ve Oliveira (2011); bellekte depolanan Redis verilerinin, ayrıca asenkron bir şekilde RDB dosya formatında diskte depolandığını belirtmişlerdir. Varsayılan ayarlarına göre, Redis bellekteki verileri en az iki saniyede bir senkronize etmektedir. Böylece yaşanacak herhangi bir sistemsel hatada sadece birkaç saniyelik veri kaybı yaşanmaktadır. Veri kalıcılığının bu derece başarılı olduğu Redis; Github, Digg, Blizzard gibi büyük markalar tarafından kullanılmaktadır.

64 bit işletim sisteminde, Redis servisinin kaynak tüketim değerleri aşağıdaki gibidir.

- i. Boş bir Redis örneği, yaklaşık 1 megabayt (MB) bellek,

¹² Redis, Introduction to Redis, <http://redis.io/topics/introduction> [erişim tarihi 29.01.2016].

¹³ Redis, FAQ, <http://redis.io/topics/faq> [erişim tarihi 29.01.2016].

- ii. 1 milyon küçük anahtar – harf dizini (string) değer çifti yaklaşık olarak 100 MB bellek,
- iii. 1 milyon anahtar – 5 alanlı bir nesneyi temsil eden hash değeri çifti yaklaşık 200 MB bellek tüketmektedir.

Lerner (2010); Linux Journal dergisindeki makalesinde, Memcached gibi yüksek hızlı önbellekleme ve depolama sistemine ihtiyaç duyulması durumunda; kolay kurulumu, yüksek performansı ve birçok programlama dili desteği bulunması sebebiyle Redis'i incelemeyi önermiştir.

Redis'in resmi web sitesi redis.io üzerinde yayınlanan performans testlerinde; saniyede 552.028 SET (değer atama), 707.463 GET (değer alma), 767.459 LPUSH (ekleme) ve 770.119 LPOP (silme) operasyonu yapılabildiği belirtilmiştir.¹⁴ Bunun yanı sıra farklı donanımsal, yazılımsal güncelleme ile farklı performans sonuçları elde edilebilmektedir.

2.2.2.2.2 MongoDB

MongoDB, birden çok platformda çalışabilen (cross platform), esnek, ölçeklenebilir, doküman tabanlı bir veri tabanıdır. MongoDB Inc firması tarafından geliştirilen MongoDB'nin; sunucusu ve araçları GNU Affero General Public Licence, sürücüler ise Apache Licence ile lisanslanmış olup ücretsiz ve açık kaynak olarak yayınlanmaktadır.¹⁵ JSON'a benzer veri modeli olan Binary JSON (BSON) kullanarak verileri depolayan MongoDB, adını “humongous” (devasa, kocaman) kelimesinden almaktadır.

Doküman tabanlı MongoDB'de, ilişkisel veri tabanlarındaki “satır” terimi, daha çok daha esnek bir model olan “doküman” olarak karşılık bulmaktadır. Verilerin yazıldığı dokümanlar; metin, sayı, tarih, nesne, dizi (array), ikili (binary) veri, alt-dokümanları ve daha birçok farklı veri tiplerini içeren bir veya daha fazla alan içerebilir. Alanlar, dokümandan dokümana değişiklik gösterebilir. MongoDB'nin bu şekilde sabit bir şema yapısına bağlı kalmadan, sunduğu dinamik şema esnekliği sayesinde; uygulama

¹⁴ Redis, How fast is Redis?, <http://redis.io/topics/benchmarks> [erişim tarihi 29.01.2016].

¹⁵ MongoDB, MongoDB Licensing, <https://www.mongodb.org/licensing> [erişim tarihi 04.02.2016].

geliştiriciler, uygulamalar için gereksinim duyulan değişiklikleri çok hızlı bir şekilde gerçekleştirebilmektedirler.

Büyük verilerin kullanıldığı sistemlerde, veri tabanlarının ölçeklenebilirliği oldukça önem arz etmektedir. MongoDB'nin sahip olduğu doküman tabanlı veri modeli ile veriyi birçok sunucu üzerinde paylaşmak daha kolaydır. MongoDB ayrıca, yük dengeleyici (load balancing) özelliği ile verinin bulunduğu sunucular üzerindeki yükler göre gelen istekleri dağıtmaktadır (Chodorow ve Dirolf 2010, ss. 3-4). Sahip olduğu özellikler ve yüksek performansı nedeniyle; Foursquare, Forbes, LinkedIn, Adobe ve eBay birçok şirket, sistemlerinde MongoDB'yi tercih etmektedir.¹⁶

MongoDB performans analizi konusunda Alabama Üniversitesi'nden Parker, Poe ve Vrbsky (2013) yaptıkları araştırmada; SQL Server üzerinde çalışan bir SQL veri tabanı ile MongoDB veri tabanı performanslarını çalışma süresini baz alarak karşılaştırmıştır. Performans karşılaştırması aşamasında, her birinde farklı miktarda veri bulunan dört farklı test senaryosu uygulanmıştır. Her senaryo için iki veri tabanında da INSERT, SELECT ve UPDATE işlemleri gerçekleştirilmiş; çoğu UPDATE işleminde SQL'in, çoğu SELECT işleminde MongoDB'nin daha hızlı çalıştığı, INSERT işleminde ise her iki sistemin birbirine yakın sonuçlar verdiği tespit edilmiştir.

MongoDB'nin performansı ile ilgili bir diğer araştırma ise Abramova ve Bernardino (2013) tarafından Cassandra ve MongoDB'yi kıyaslayarak gerçekleştirilmiştir. Üç farklı kayıt sayısına ve farklı okuma, güncelleme, yazma işlemi oranlarına sahip senaryolarla gerçekleştirilen testlerde; 100 bin kayıt bulunan veri tabanlarında okuma işlemlerinde MongoDB'nin, Cassandra'ya göre daha hızlı yanıt verdiği belirtilmiştir.

2.2.2.3 Arama motoru sunucuları

Veri tabanlarında depolanan verilerin boyutu ve miktarı artması, yapılacak sorgulama işlemlerinde hız ve performans problemleri ile karşılaşmaktadır. Özellikle metin

¹⁶ MongoDB, Who's using MongoDB?, <https://www.mongodb.org/community/deployments?group=innovation-awards>. [erişim tarihi 04.02.2016].

tabanlı büyük verilerin veri tabanları tarafından indekslenmesi veri tabanı sunucusu için ekstra yük oluşturacağından, bu sorunu üçüncü parti yazılımlar ile çözüme yoluna gidilmiştir. Arama motoru sunucuları, büyük boyutlu metin bulunduran veri tabanlarında veri analizi ve sorgulamalarda kullanılan yazılımlardır. Çok büyük veri tabanlarıyla işlem yapılan sistemlerde ciddi bir performans artışı sağlamakta ve daha iyi arama sonuçları vermektedirler. Bu bölümde Sphinx ve Elasticsearch uygulamaları incelenecektir.

2.2.2.3.1 *Sphinx*

Sphinx, veri tabanlarındaki verileri belirli aralıklarla dizine ekleyerek (endeksleyerek), tam metin arama işlemlerini hızlandıran bir arama motorudur. SQL veri tabanlarıyla entegre bir şekilde tasarlanmış olan Sphinx, GPL lisansı altında dağıtılmaktadır.

Uygulamalarda Sphinx arama sonuçlarına üç farklı şekilde erişilebilmektedir:

- i. SphinxQL (Sphinx'in MySQL için hazırladığı uygulama) adı verilen küçük SQL alt kümeleri ile,
- ii. SphinxAPI arama Application Programming Interface (API) ile,
- iii. SphinxSE depolama motoru ile MySQL sunucu üzerinden.

Sphinx'in resmi web sitesinden yer alan bilgilere göre, uygulamanın bazı özellikleri aşağıda yer almaktadır.¹⁷

- i. Saniyede 10-15 MB arasında değişen yüksek indeksleme performansı,
- ii. 1 milyon dokümanın bulunduğu arşivde saniyede 150 - 250 sorgu ile yüksek arama performansı,
- iii. Gelişmiş indeksleme ve sorgulama araçları,
- iv. Saniyede binlerce sorguya, terabaytlarca veriyi işleyebilecek ölçeklenebilirlik,

¹⁷ Sphinx, Sphinx features, <http://sphinxsearch.com/docs/latest/features.html> [erişim tarihi 08.02.2016].

- v. SQL ve Extensible Markup Language (XML) veri kaynaklarıyla kolay entegrasyon,
- vi. Metin haricinde farklı veri tipi desteği (JSON, tarih, sayı),
- vii. SphinxQL, SphinxAPI ve SphinxSE arama ara yüzleri.

2.2.2.3.2 Elasticsearch

Elasticsearch yüksek ölçeklenebilirliğe sahip, açık kaynak tam metin arama ve analiz motorudur. Representational State Transfer (RESTful) API desteği bulunan Elasticsearch uygulamasının resmi web sitesinde yer alan bilgiye göre, Elasticsearch; Java, .NET, PHP, Python, Ruby gibi birçok programlama dili ile entegre bir şekilde çalışabilir yapıya sahiptir.¹⁸ Apache Lucene tabanlı olarak geliştirilen tam metin arama konusunda oldukça iyi performansa sahip Elasticsearch, çok dilli aramayı destekleyen sorgu API'yi, bunu mu demek istediniz önerisi, otomatik tamamlama gibi özelliklere sahiptir. Java programlama diliyle yazılmış olup, 2010 yılında ilk sürümü yayınlanan Elasticsearch yazılımını, Apache Lisansı altında açık kaynak olarak dağıtılmaktadır.

Elasticsearch verileri "index" adı verilen yapılarda depolamaktadır. JSON veri tipli doküman tabanlı bir yapıya sahip olan Elasticsearch, bağımsız bir şema yapısı imkânı sunmaktadır. Yeni bir kayıt eklendiği zaman Elasticsearch, veri yapısını kendi tespit edip buna göre endekslemektedir. Elasticsearch'te, ilişkisel veri tabanlarındaki tablo yapısına benzer yapıda tipler (types) bulunmaktadır. Endekslenen bir dokümanda bir veya daha fazla tip bulunabilmektedir. Yeni doküman eklemek için JSON tipindeki nesne ile Hyper-Text Transfer Protocol (HTTP) POST metodu, veri sorgulamak ve arama yapmak için ise http GET metodu kullanılmaktadır. Elasticsearch'ün yapısını oluşturan terimlerin, SQL ve MongoDB'de ki karşılıkları Tablo 2.4'te belirtilmektedir.

¹⁸ Elasticsearch, Community Contributed Clients, <https://www.elastic.co/guide/en/elasticsearch/client/community/current/index.html> [erişim tarihi 10.02.2016].

Tablo 2.4: Elasticsearch yapısal terimlerinin SQL ve MongoDB'deki karşılıkları

Elasticsearch	SQL	MongoDB
Endeks (Index)	Veri tabanı	Veri tabanı
Tip (Type)	Tablo	Koleksiyon (Collection)
Alan (Field)	Alan	Alan
JSON nesnelere	Kayıtlar	BSON nesnelere

Kaynak: Paro, A. (2015). Elasticsearch Cookbook

Kuó ve Rogoziński (2013), Elasticsearch ile ilgili arařtırmalarına yer verdikleri kitapta; ölçeklenebilir ve dağıtılabılır olma konusuna değinmiřtir. Kitapta yer alan bilgiye göre Elasticsearch, binlerce sunucu üzerinde petabaytlarca veri depolayabilecek řekilde tasarlanmıřtır. Bağımsız, tekil bir sunucu olarak çalışabilen Elasticsearch, çok büyük boyutlu verileri işleyebilmek ve hata toleransını azaltmak adına, cluster adı verilen birbiriyle bağlantılı birçok sunucuda dağıtık olarak çalışabilmektedir. Veriler, cluster içerisinde bulunan ve her birine node adı verilen sunucular üzerinde daha küçük parçalara (shard) bölünerek dağıtılabilmektedir. Bu tür bir yapıda yapılan sorgulamalarda Elasticsearch; sorguyu ilişkili tüm parçalara (shard) iletir ve sonuçları birleřtirerek yanıt döndürür.

Tüm bu özellikleri nedeniyle, Wikipedia, Stack Overflow, Github, Xing, Tango, Docker gibi projelerde farklı kullanım senaryoları ile kullanılmaktadır. Wikipedia, tam metin arama ve “bunu mu demek istemiřtiniz?” modülleri için; Github ise 130 milyar satırlık kod sorgusu için Elasticsearch kullanılmaktadır.¹⁹

Bai (2013) tarafından 9. International Conference on Natural Computation (ICNC)'da; HBase ve Elasticsearch tabanlı, büyük boyutlu günlük (log) verilerinin içerisinde gerçek zamanlı arama analizi arařtırması yayınlanmıřtır. 7 GB büyüklüğünde ve 148.928.992 olayı içeren veri içerisinde “bigdata” kelimesi aranmıř ve farklı miktarda sonuçlara göre yanıt süresi ölçülmüřtür. Yapılan ölçümlere göre, belli bir sayıya kadar arama sonucu

¹⁹ Elasticsearch, Use Cases, <https://www.elastic.co/use-cases> [eriřim tarihi 10.02.2016].

miktarı arttıkça yanıt süresi de artmış fakat bir noktadan sonra yanıt süresi azalmaya başlamıştır.

2.2.3 Sunucu Performans Çözümleri

Uygulamaların üzerinde çalıştığı sunucunun performansının yüksek olması, uygulamanın da doğrudan yüksek performans ile çalışmasını sağlamaktadır. Sunucuya iyi donanımlar entegre edilerek uygulamanın performansı yükseltilebilir, fakat çok yoğun trafik alan uygulamalarda en iyi donanımın bulunan sunucular da yetersiz kalabilmektedir. Bunun yanı sıra, yüksek donanım maliyetleri de internet sektöründe hizmet veren firmaların en büyük problemlerinden biridir. Maliyetleri azaltmak adına, uygulamaların ve sunucunun optimizasyonu gerekmektedir. Bu bölümde sunucu tarafında yapılabilecek optimizasyon çalışmaları ve üçüncü parti uygulamalar yer alacaktır.

2.2.3.1 Web sunucu yazılımları

İnternet kullanıcılarının, web sunucusu üzerinde bulunan sayfalara, dokümanlara ve erişimini sağlayan yazılımlardır. Statik içeriklerin servisi, Secure Sockets Layer (SSL) / Transport Layer Security (TLS) desteği, ters vekil (reverse proxy) özelliği, yük dengeleme, içerik sıkıştırma ve önbellekleme, yetki kontrolü, gelişmiş günlük ve daha birçok işlevi gerçekleştirebilmektedirler. Bu bölümde en çok tercih edilen web sunucu yazılımları olan Apache ve Nginx incelenecektir.

2.2.3.1.1 Apache HTTP sunucusu

Apache HTTP sunucusu, Apache Yazılım Vakfı (The Apache Software Foundation) tarafından geliştirilen, güçlü, ücretsiz ve açık kaynak bir web sunucu yazılımıdır. 1995 yılında çalışmalarına başlanan Apache HTTP sunucusu, Unix ve Windows işletim sistemlerine uyumluluğu ile 2016 Şubat ayı itibariyle günümüzde en çok kullanılan web

sunucusu yazılımıdır.²⁰ Apache sunucusu çalıştırılmaya başlandığında, çalışan işlemler listesindeki işlem ismi "HTTP daemon" kelimesinin kısaltılışı olan httpd olarak görünmektedir.

Apache web sunucusunun bazı özellikleri aşağıda belirtilmiştir.

- i. Tamamen açık kaynaktır.
- ii. Yüksek bir ölçeklenebilirliğe sahiptir.
- iii. mod_proxy modülü ile ters vekil (reverse proxy) olarak önbellekleme, yük dengeleyici görevlerinde çalışabilmektedir.
- iv. OpenSSL ile SSL ve TLS desteğine sahiptir.
- v. Sürekli olarak geliştirilmesine devam edilmektedir.
- vi. Üçüncü parti yazılımlarla genişletilebilir bir yapıya sahiptir.
- vii. Gzip sıkıştırma ve açma özelliğine sahiptir.
- viii. Apache modül API kullanılarak yazılacak modüllerle isteğe göre özelleştirilebilir.
- ix. Hata raporlama, yama yazılımlar, destek konusunda gelişmiş topluluklara sahiptir.
- x. Gelişmiş kimlik doğrulama ve yetki kontrolüne sahiptir.
- xi. .htaccess dosyası ile web sunucu yapılandırmasından bağımsız olarak izin seviyesinde yapılandırma yapılabilmektedir.
- xii. mod_cgi modülü ile Common Gateway Interface (CGI) ile yazılmış dinamik sayfaları çalıştırabilmektedir.
- xiii. mod_status modülü ile gerçek zamanlı performans izlemesi yapılabilmektedir.

2.2.3.1.2 Nginx

Nginx, minimum sistem kaynağı tüketme amacıyla tasarlanmış, eş zamanlı yüksek sayıda isteğe cevap verebilecek, yüksek performanslı web sunucusudur. Web sunucusu

²⁰ Netcraft, February 2016 Web Server Survey, <http://news.netcraft.com/archives/2016/02/22/february-2016-web-server-survey.html> [erişim tarihi 17.02.2016]

olmasının yanı sıra, ters vekil sunucu, e-posta vekil sunucusu ve Transmission Control Protocol (TCP) vekil sunucusu olarak ta çalışabilmektedir. Yapılan testlerle Linux, Mac OS, Windows, FreeBSD, Solaris, AIX ve HP-UX işletim sistemleriyle uyumlu olduğu belirtilmektedir.²¹

Netcraft firmasının yaptığı araştırmalara göre, en çok kullanılan üç web sunucusundan biri olan Nginx; Wordpress, HP, Wix gibi markalar tarafından tercih edilmektedir.²² Nginx'in çalışma yapısının incelenmesi durumunda, bir adet ana işlem (master process) ve buna bağlı olan alt işlemler olduğu görülmektedir. Bu alt işlemler (child process) önbellek yöneticisi (cache manager), önbellek yükleyicisi (cache loader) ve birkaç tane de işçi işleminden (worker process) oluşmaktadır. Ana işlem, belirtilen konfigürasyonu okuyarak işlemleri başlatır ve alt işlemleri oluşturur. Önbellek yükleyici işlemi, başlangıçta diskte bulunan önbelleğe alınmış veriyi belleğe aktarır. Önbellek yönetici işlemi, periyodik olarak çalışır ve diske önbellek olarak kaydedilecek verileri konfigürasyonda belirtilen limiti aşmaması amacıyla küçültür. İşçi işlemler ise ağ bağlantılarını işler, diskteki veriyi okur ve diske veri yazar, veri akışı olan sunucular arası iletişimi sağlayarak neredeyse işin tamamını yapar.

Nginx web sitesinde yer alan bilgilere göre, Nginx HTTP sunucusunun bazı özellikleri aşağıda belirtilmiştir.²³

- i. Statik dosyaların servisini sağlar. Statik dosyalar için otomatik endeksleme özelliği mevcuttur.
- ii. Ters vekil sunucusu olarak çalışabilme, önbellekleme ve yük dengeleme özelliğine sahiptir.
- iii. FastCGI, uwsgi, Simple Common Gateway Interface (SCGI) desteğine ve önbellekleme özelliğine sahiptir.
- iv. OpenSSL ile SSL ve TLS desteğine sahiptir.
- v. Gzip sıkıştırma ve açma özelliğine sahiptir.

²¹ Nginx, Tested OS and platforms, http://nginx.org/en/#tested_os_and_platforms [erişim tarihi 22.02.2016]

²² Nginx, <https://www.nginx.com/customers> [erişim tarihi 22.02.2016]

²³ Nginx, Basic HTTP server features, <http://nginx.org/en/> [erişim tarihi 22.02.2016]

- vi. IP veya şifre bazlı yetki doğrulaması yapabilmektedir.
- vii. Eş zamanlı ya da bir adresten anlık gelen bağlantı sayısı kısıtlanabilmektedir.
- viii. Esnek bir şekilde yapılandırılabilir bir yapıya sahiptir.
- ix. Erişim, hata, sistem kayıtları gibi gelişmiş günlük tutma özelliğine sahiptir.

2.2.3.2 PHP işleyiciler (PHP handlers)

PHP işleyiciler; web sunucusu üzerinde bulunan PHP dosyalarının yorumlanması aşamasında kullanılan uygulamalardır. Söz konusu işleyiciler, PHP yorumlayıcı ile web sunucu arasında bir iletişim protokolü olarak görev alır. PHP sürümünün seçimi gibi, PHP işleyicinin seçimi de web uygulamasının performansı açısından önem arz etmektedir. Dynamic Shared Object (DSO), CGI, Single User PHP (suPHP) ve FastCGI (FCGI); sık kullanılan PHP işleyicilerdir. CPU ve bellek tüketimi, uygulama güvenliği gibi konularda farklılıklar yaratan bu uygulamalardan herhangi birinin seçimini; sistemin ihtiyaçları doğrultusunda gerçekleştirmek gerekmektedir.

- i. **DSO:** mod_php olarak bilinen bu uygulama; Apache modülü gibi çalıştığı için, PHP betiği “nobody” (Ubuntu’da www-data) kullanıcı adıyla çalıştırılır. Düşük CPU ve bellek tüketimi ve yüksek performans avantajlarının yanı sıra, “nobody” kullanıcısı tarafından çalıştırılması sonucu oluşan güvenlik zafiyeti de bu uygulamanın dezavantajıdır.
- ii. **CGI:** PHP; bir Apache modülü olarak değil CGI modülü olarak çalışır. PHP betiği DSO’da olduğu gibi “nobody” kullanıcı adıyla çalıştırılır fakat sunucu da “Apache suEXEC” modülü aktif ise, isteğin hangi kullanıcıdan geldiği öğrenilebilmektedir. Düşük bellek tüketimi ile çalışan bu uygulama; güvenlik ve hız konusunda ise düşük performanslıdır.
- iii. **SuPHP:** PHP, CGI işleyicisinde olduğu gibi bir CGI modülü olarak çalışır. Fakat CGI işleyicisinden farklı olarak; PHP betiği “nobody” olarak değil, mevcut kullanıcı adıyla çalıştırılır. CPanel tarafından varsayılan olarak kullanılan bu işleyici; düşük bellek tüketimi ve güvenlik konusunda avantajlıyken, yüksek CPU kullanımı nedeniyle yoğun trafik alan sunucularda kapasiteyi zorlamaktadır.

- iv. **FCGI:** mod_fcgid olarak ta bilinen FCGI, CGI işleyicisinin daha yüksek performanslı alternatifidir. PHP FastCGI Process Manager (PHP-FPM) ile kullanılarak PHP betiği mevcut kullanıcı adıyla çalıştırılır ve düşük CPU kullanımı ile yüksek performanslı olarak çalışır. FastCGI oluşturduğu işlem havuzu ile yüksek trafikli sunucularda performans artışı sağlar.

Web sunucusu yazılımı ve PHP işleyicilerle ilgili, IBM araştırmacılarından Suzumura ve diğerleri (2008), PHP ve JSP dillerinin performans ölçümleri sırasında yaptıkları araştırmada; PHP tabanlı uygulama performansı için Apache + DSO, Apache + FCGI ve Litespeed + FCGI olmak üzere üç farklı sistem kullanmıştır. PHP tabanlı bankacılık, e-ticaret ve destek sistemlerinde uygulanan test işlemleri sonrasında, en yüksek performansı Litespeed + FCGI altyapısı; en düşük performansı Apache + FCGI altyapısı sergilemiştir.

2.2.3.3 PHP yorumlayıcılar (PHP interpreters)

2.2.3.3.1 HipHop Virtual Machine (HHVM)

HHVM; PHP ve Facebook tarafından geliştirilen Hack programlama diliyle geliştirilmiş uygulamaları çalıştırmak için tasarlanmış açık kaynak bir sanal makine uygulamasıdır. PHP 5 desteğinin yanı sıra, PHP 7'nin birçok özelliğini de desteklemektedir. HHVM PHP kodlarını bytecode tipine çevirir ve çevrilen bu işlem kodu just-in-time derleyicisiyle makine diline dönüştürülür.²⁴

Facebook mühendislerinin (Keith Adams vd., 2014) yapmış olduğu testlerde, HHVM'in standart PHP yorumlayıcısına göre Drupal altyapılı uygulamada yaklaşık 1,5 ve Wordpress altyapılı uygulamada yaklaşık 1,6 kat performans artışı sağladığı belirlenmiştir. Bunun yanı sıra facebook.com'da HHVM'e geçiş sonrası uygulama hızınının 1,8 kata varan düzeyde arttığı belirtilmiştir.

²⁴ HHVM, <http://hhvm.com/> [erişim tarihi 27.02.2016]

2.2.3.4 Web sunucu hızlandırıcıları

Web sunucu hızlandırıcıları; ters vekil sunucuları gibi web sunucusunun önünde çalışan ve gelen isteği ilk olarak karşılayan yazılımlardır. Önbellekleme ve sıkıştırma gibi teknikleri kullanarak, içeriğin istemciye daha hızlı şekilde iletilmesini sağlamaktadırlar. Özellikle yoğun trafik alan sistemlerde, kaynak tüketimini ve sistemin yanıt süresini önemli derecede azaltmaktadır. Varnish Cache; web sunucu yazılımı olarak kullanılan Nginx'in yanı sıra bu amaçla kullanılan en popüler yazılımdır.

2.2.3.4.1 Varnish Cache

Varnish Cache, dosyaları ya da dosyaların belirli kısımlarını bellekte tutarak, aynı istekler için yanıt süresini ve sunucunun bant genişliği (bandwidth) tüketimi düşüren yüksek performanslı, açık kaynak bir web hızlandırıcısıdır. Varnish; Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) ve diğer network protokollerini destekleyen vekil sunucularından farklı olarak sadece HTTP üzerine yoğunlaşmıştır. Linux ve FreeBSD işletim sistemlerinde geliştirilen ve test edilen Varnish'i sadece bir önbellekleme yapan ters HTTP vekili olarak görmemek gerekir. Kurulumu bağı olarak; uygulama güvenlik duvarı, Distributed Denial of Service attack (DDOS) savunması, yük dengeleyici görevlerini de yerine getirebilmektedir.

Varnish, gelen istekleri ayrı ayrı çalışan iş parçacıkları olarak işler. Yapılandırma ayarlarındaki aktif iş parçacığı sayısı limitine ulaşıldığında, yeni gelen bağlantılar kuyruğa alınır. Kuyruğa alınan bu bağlantılar da ayarlarda belirlenen limite ulaşırsa, yeni gelecek istekler doğrudan reddedilir.

Varnish Configuration Language (VCL), Varnish'in temel yapılandırma mekanizmasıdır. VCL kodları kullanılarak yazılacak eklenti ile sunucuya gelen istekler ve doküman önbellekleme ilkeleri farklı bir şekilde tanımlanabilmektedir. Yüklenen VCL eklenti, C diline çevrilerek derlenir. Varnish, bu esnek ve uyarlanabilir özelliğiyle diğer web hızlandırıcılara göre öne çıkmaktadır. Varnish ayrıca Varnish Modules

(VMODs) özelliđi ile birçok VCL eklenti desteđi sađlayarak daha geliřtirilebilir bir yapıya sahiptir.

Graziano (2013), Linux Journal dergisinde yayınlanan bir yazısında, Varnish Cache'in performansı ile ilgili arařtırmasına yer vermiřtir. Graziano'nun yaptıđı testler sonucunda Apache sunucu üzerinde alıřan bir dosyaya, Varnish portundan (6081) eriřim yapmıř ve yanıt süresinde yüzde 89 oranında bir azalma sađlamıřtır.



3. VERİ VE YÖNTEM

Bir uygulamanın performansı çok farklı boyutlarda değerlendirilebilir. Uygulamanın CPU kullanımı, ölçeklenebilirlik birer performans kıstası olarak gösterilebilirken; bu çalışmada performans, uygulamanın yanıt süresi ve sistem kaynak tüketimi olarak değerlendirilecektir. Bu değerlendirmeyi yapabilmek için, birçok performans test araçları bulunmaktadır. Bu tez çalışmasında PHP tabanlı uygulamaların performansını değerlendirmek için Xdebug, ab ve Apache JMeter; MySQL sorgularının performansını analiz etmek için ise mysqlslap uygulaması kullanılacaktır.

Performans test işlemleri, “Windows 7 Home Premium 64 bit” işletim sistemi üzerinde kurulacak olan uygulamalar ile gerçekleştirilecektir. Test işlemi uygulanacak sistemin yazılım geliştirme çalışmaları da bu bilgisayar da gerçekleşmektedir. Test işlemlerinin yapılacağı bilgisayarın donanımsal özellikleri Tablo 3.1’de belirtildiği gibi olacaktır.

Tablo 3.1: Test işlemlerinin yapılacağı bilgisayarın donanımsal özellikleri

Donanım	Özellik
İşlemci	Intel(R) Core(TM) i7-3610QM CPU @ 2.30 GHz
RAM	8,00 GB

Yukarıda bilgileri yer alan sistem üzerinde gerçekleştirilen yazılım geliştirme çalışmalarının hata ayıklama ve analiz işlemleri, uygulamanın çalışacağı sistemden önce XAMPP v1.8.3 programı vasıtasıyla Windows üzerine Apache, MySQL ve PHP kurularak oluşturulan ortam üzerinde gerçekleştirilmiştir. PHP’nin phpinfo() fonksiyonu yardımıyla alınan sistem bilgileri Tablo 3.2’de yer aldığı gibidir.

Tablo 3.2: Hata ayıklama ve kod analiz işlemlerinin yapılacağı sistem bilgileri

PHP Version	5.5.9
Compiler	MSVC11 (Visual C++ 2012)
Server API	Apache 2.0 Handler
Apache Version	Apache/2.4.7 (Win32) OpenSSL/1.0.1e PHP/5.5.9

Teste tabi tutulacak uygulamalar, aynı bilgisayar üzerinde VMware Workstation programı aracılığı ile “Ubuntu Server 15.10 64 bit” işletim sistemi kurularak sanallaştırılan sistem üzerinde çalışacaktır. VMware Workstation, bir bilgisayar üzerinde Windows, Linux, FreeBSD, Solaris gibi birden fazla işletim sistemi çalıştırılmasını sağlayan yazılımdır. VMware Workstation kullanarak; 2.048 MB RAM ve 100 GB disk alanı kapasitesine sahip sanal sunucu oluşturulmuştur. Söz konusu sanal sunucuya, sırasıyla aşağıdaki komutlar yardımıyla Apache web sunucusu, PHP5 ve MySQL veri tabanı sunucusu kurulumu yapılmıştır.

```
sudo apt-get update
sudo apt-get install apache2
sudo apt-get install mysql-server
sudo apt-get install php5
sudo apt-get install libapache2-mod-php5
sudo apt-get install php5-cgi
sudo apt-get install php5-cli
sudo apt-get install php5-mysql
sudo service apache2 restart
```

Yapılan kurulumlar sonrası, sanal sunucu üzerinde çalışan Apache web sunucusuna erişim IP’si sistem tarafından 192.168.78.128 olarak oluşturulmuştur. Bu web sunucusundaki performans analizleri, söz konusu IP adresine yapılacak isteklerle gerçekleştirilecektir.

Test sunucusundaki PHP’nin phpinfo() fonksiyonu yardımıyla alınan sistem bilgileri Tablo 3.3’te yer almaktadır.

Tablo 3.3: Test işlemlerinin yapıldığı sunucuda, PHP’nin phpinfo() fonksiyonu yardımıyla alınan sistem bilgileri.

PHP Version:	5.6.11-1ubuntu3.1
Server API:	Apache 2.0 Handler
Apache Version:	Apache/2.4.12 (Ubuntu)
MySQL Version:	5.6.28

3.1 PERFORMANS TEST ARAÇLARI

Performans testleri, çeşitli koşullarda uygulanan yük altında; uygulama, veri tabanı, işletim sistemi ya da ağın göstereceği tepkiyi incelemek amacıyla gerçekleştirilmektedir. Sistemde bir açık bulmak bu testlerdeki esas amaç olmayıp, sistemin belirlenen koşullarda nasıl sonuçlar verdiğini ölçmek öncelikli amaçtır. Performans testlerindeki sonuçların güvenilirliğinin sağlanması için, testin yapıldığı ve uygulandığı ortamın test uygulandığı sürece sabit ve istikrarlı bir yapıda olması gerekir. Koşulların sürekli değiştiği ortamda, buna bağlı olarak sonuçlar da düzensizlik gösterecektir.

3.1.1 ab

ab, Apache web sunusunun performansını ölçmek amacıyla, Apache tarafından geliştirilen test aracıdır. Apache Lisansı altında, ücretsiz ve açık kaynak bir şekilde dağıtılmaktadır. Herhangi bir kullanıcı ara yüzüne sahip olmayan bu uygulama, komut satırı üzerinden çalıştırılmaktadır.

Bu tez çalışmasında ab uygulaması, sisteme belirli miktarda eşzamanlı olarak gönderilen istekleri, sistemin yanıtlama süresini tespit etmek amacıyla kullanılacaktır.

Versiyon numarası 2.3 olan ab ile performans testi uygulanırken aşağıda belirtilen adımlar izlenmektedir:

- i. Windows komut satırında doğrudan test işlemleri yapabilmek için ab uygulamasının bulunduğu klasöre “cd C:\xampp\apache\bin” komutu ile geçiş yapılır.
- ii. Test işlemi uygulamak için; “ab -n SAYI1 -c SAYI 2 localhost/” formatında komut yazılır. Ardından test sonuçları ekrana gelmeye başlar. Bu komutta yer alan SAYI1 değişkeni, uygulanacak toplam istek sayısını, SAYI2 değişkeni ise anlık istek sayısını belirtmektedir. Ayrıca “localhost/” yerine istek gönderilecek sayfa adresi yazılmaktadır. Bu parametrelerinin yanı sıra

farklı parametreler de bulunmaktadır, fakat test işlemleri için bu iki parametre yeterli olacaktır.

- iii. Test sonuç çıktısında; yanıt süresi, tamamlanan istek sayısı, sunucunun bir saniyede karşılayabileceği istek sayısı gibi veriler yer almaktadır.

3.1.2 Apache JMeter

Apache JMeter, Java ile yazılmış performans test aracıdır. Web uygulamalarının yanı sıra, Simple Object Access Protocol (SOAP) web servisleri, mail, FTP ve Java Database Connectivity (JDBC) kullanarak veri tabanı sunucularının performans analizi ve yük testi için kullanılabilmesi ab'den ayıran özelliklerinin başında gelmektedir. Birden fazla eşzamanlı kullanıcı simülasyonu ile gerçekleşen testlere ait sonuçlar tablo veya grafik şeklinde alabilme olanağı sunmaktadır.

JMeter (versiyon 2.13) uygulaması, sisteme belirli miktarda eşzamanlı olarak istek gönderilerek, sistemin yanıtlama süresini tespit etmek amacıyla kullanılacaktır. Beklenen performansın elde edilemediği durumlarda sorunun tespiti için, hata ayıklama ve durum analizi işlevi bulunan XDebug eklentisi kullanılacaktır.

JMeter ile yük testi aşağıdaki aşamalar izlenerek gerçekleştirilmiştir.

- i. Program çalıştırıldığında sol menüde bir “Test Plan” adında örnek bir test çalışması yer almaktadır. Bu çalışmanın adı, üzerine bir kere tıklanarak değiştirilebilmektedir.
- ii. Söz konusu çalışmanın üzerine sağ tıklayıp, açılan menüde “Add” => “Threads (Users)” => “Thread Group” seçeneği seçilir.
- iii. Thread Group, uygulanacak yük testindeki kullanıcı sayısı, isteğin tekrar sayısı ve zamanının ayarlandığı adımdır.
- iv. Gerekli ayarlar yapıldıktan sonra, sol menüdeki “Thread Group” seçeneğine sağ tıklayıp, açılan menüde “Add” => “Sampler” => “HTTP Request” seçeneği seçilir.

- v. HTTP Request adımı, yük testinin uygulanacağı web ya da IP adresi, port numarası, proxy, istek yöntemi gibi ayarların yapıldığı adımdır.
- vi. Bu aşamadan sonra, sol menüde yer alan çalışmanın üzerine sağ tıklanarak, açılan menüde “Add” => “Listener” => “View Results in Table” seçeneği seçilir.
- vii. “View Results in Table” adımı test sonuçlarının tablo olarak alındığı adımdır.

3.1.3 XDebug

XDebug, PHP tabanlı uygulamalarda hata ayıklama ve uygulama içerisindeki metodların ne kadar sürede işlevlerini yerine getirdiğinin (profil) incelenmesini sağlayan bir PHP aracıdır. Uygulamayı analiz etmek için sistem kaynak tüketimi fazla olduğu için, genellikle uygulama geliştirme sırasında kullanılmaktadır. XDebug eklentisinin analiz sonuçları doğrudan okunabilecek formatta olmadığı (cachegrind.out uzantılı) için, sonuçları çözümleyebilecek ve gösterebilecek ek bir uygulama gerekmektedir. Bu işlevi yapabilecek birçok uygulama mevcut olup, bu çalışmada analiz sonuçları sadece Windows işletim sistemi üzerinde çalışan WinCacheGrind uygulamasıyla incelenecektir.

XDebug eklentisinin etkinleştirilmesi için PHP'nin konfigürasyonlarının yer aldığı php.ini dosyasında, **xdebug.profiler_enable** değeri 1 olarak, analiz sonuçlarının çıktısı alınacak klasörün yolu ise **xdebug.profiler_output_dir** değeri ise "C:\xampp\tmp" olarak güncellenmiştir. Bu düzenleme ile birlikte, localhost üzerinde istek gönderilen PHP dosyalarının cachegrind.out formatlı profil sonuçları “C:\xampp\tmp” dizininde oluşur.

WinCacheGrind programının, “File” menüsü altındaki “Open” seçeneği ile “C:\xampp\tmp” klasöründeki analiz sonuçlarından ilgili olanı seçilir ve gerekli inceleme yapılır.

3.1.4 mysqlslap

mysqlslap, MySQL sunucusu üzerine çeşitli simülasyonlara göre gönderilen yükün sonuçlarını analiz etmek için kullanılan bir araçtır. Bu tez çalışmasında, MySQL sorgu analizleri bölümünde kullanılacak olan mysqlslap aracı, komut satırında aldığı ekstra birkaç parametre ile çalıştırılmaktadır. Aşağıda test işlemleri esnasında sıkça kullanılacak bir komut örneği mevcuttur.

```
mysqlslap --user=KULLANICI_ADI --password --host=localhost --  
concurrency=ANLIK_ISTEK_SAYISI --iterations=TEKRAR_SAYISI --create-  
schema=SEMA_CIKTISI_ADI --query="SQL_SORGUSU;"
```

Anlık istek sayısı, sorgunun tekrarlanma sayısı gibi farklı parametrelerle farklı yük testleri uygulayarak sorgunun yüksek trafik altındaki performansı da incelenmiş olmaktadır. mysqlslap komutu çalıştırdıktan sonra ekranda isteklerin ortalama, minimum ve maksimum çalışma süreleri listelenmektedir. Analizler bu değerler göz önünde bulundurularak gerçekleştirilecektir.

4. BULGULAR

4.1 PHP PERFORMANS TESTLERİ

4.1.1 PHP Kod Performans Testleri

4.1.1.1 strtr ve strreplace fonksiyonları

10.000 kere çalışacak bir for döngüsü içerisinde; “Selam dünya!” cümlesindeki “Selam” kelimesinin “Merhaba” kelimesi ile değiştirilmesi işlemi strtr ve strreplace fonksiyonları kullanılmıştır.

Fonksiyonların bulunduğu dosyalara üç farklı istek gönderilmiş ve dosyaların çalışma sürelerinin milisaniye (ms) türünden ölçümleri Tablo 4.1’de belirtilmiştir.

Tablo 4.1: strtr ve strreplace fonksiyonlarının Xdebug Profiler sonuçları

	strtr (ms)	strreplace (ms)
1. İstek	95	81
2. İstek	75	51
3. İstek	82	52
Ortalama	84	61

Dosyalara gönderilen saniyede 1 istek ile toplam 20 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.2’de yer almaktadır.

Tablo 4.2: strtr ve strreplace fonksiyonlarının JMeter sonuç ortalaması

	strtr (ms)	strreplace (ms)
Ortalama	93	36

Dosyalara gönderilen anlık 1 istek ile toplam 20 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.3’te yer almaktadır.

Tablo 4.3: strtr ve strreplace fonksiyonlarının ab sonuç ortalaması

	strtr (ms)	strreplace (ms)
Ortalama	80,77	17,82

4.1.1.2 “.” (nokta) ve “,” (virgül) string birleştirme operatörleri

10.000 kere çalışacak bir for döngüsü içerisinde; “Selam dünya! Bu bir testtir.” cümlesindeki “Selam”, “dünya!”, “Bu bir testtir.” ifadelerinin birleştirilmesinde işleminde “.” (nokta) ve “,” (virgül) operatörleri kullanılmıştır.

Operatörlerin kullanıldığı dosyalara üç farklı istek gönderilmiş ve dosyaların çalışma sürelerinin ms türünden ölçümleri Tablo 4.4’te belirtilmiştir.

Tablo 4.4: “.” (nokta) ve “,” (virgül) string birleştirme operatörlerinin Xdebug Profiler sonuçları

	“,” (virgül) operatörü (ms)	“.” (nokta) operatörü (ms)
1. İstek	16	16
2. İstek	18	16
3. İstek	15	20
Ortalama	16,3	17,3

Dosyalara gönderilen saniyede 1 istek ile toplam 20 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.5’te yer almaktadır.

Tablo 4.5: “.” ve “,” string birleştirme operatörlerinin JMeter sonuç ortalaması

	“,” (virgül) operatörü (ms)	“.” (nokta) operatörü (ms)
Ortalama	28	33

Dosyalara gönderilen anlık 1 istek ile toplam 20 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.6’da yer almaktadır.

Tablo 4.6: “.” ve “,” string birleştirme operatörlerinin ab sonuç ortalaması

	“,” (virgül) operatörü (ms)	“.” (nokta) operatörü (ms)
Ortalama	13,3	13,9

4.1.1.3 " (çift tırnak) ve ' (tek tırnak) operatörleri

100.000 kere çalışacak bir for döngüsü içerisinde; “Selam dünya” metnini içeren bir değişken “Bu bir testtir.” cümlesiyle iki farklı şekilde birleştirilerek ekrana yazdırılmıştır. Tek tırnak içerisinde metnin, tırnak dışına yazılan değişkeni nokta (.) operatörü ile birleştirilerek ve çift tırnak içerisinde hem değişkenin hem de metnin yazıldığı iki farklı dosya oluşturulmuştur.

Çift tırnak operatörünün kullanıldığı dosyanın kod yapısı aşağıda yer almaktadır.

```
<?php
$merhaba = "Merhaba dünya!";
echo "$merhaba Bu bir testtir";
?>
```

Tek tırnak operatörünün kullanıldığı dosyanın kod yapısı aşağıda yer almaktadır.

```
<?php
$merhaba = "Merhaba dünya!";
echo $merhaba . ' Bu bir testtir.';
?>
```

Operatörlerin kullanıldığı dosyalara üç farklı istek gönderilmiş ve dosyaların çalışma sürelerinin ms türünden ölçümleri Tablo 4.7’de belirtilmiştir.

Tablo 4.7: " (çift tırnak) ve ' (tek tırnak) operatörlerinin Xdebug Profiler sonuçları

	" (çift tırnak) operatörü (ms)	' (tek tırnak) operatörü (ms)
1. İstek	269	404
2. İstek	369	310
3. İstek	428	321
Ortalama	355,33	345

Dosyalara gönderilen saniyede 1 istek ile toplam 20 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.8’de yer almaktadır.

Tablo 4.8: " (çift tırnak) ve ' (tek tırnak) operatörlerinin JMeter sonuç ortalaması

	" (çift tırnak) operatörü (ms)	' (tek tırnak) operatörü (ms)
Ortalama	69	69

Dosyalara gönderilen anlık 1 istek ile toplam 20 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.9'da yer almaktadır.

Tablo 4.9: " (çift tırnak) ve ' (tek tırnak) operatörlerinin ab sonuç ortalaması

	" (çift tırnak) operatörü (ms)	' (tek tırnak) operatörü (ms)
Ortalama	75,7	71,8

4.1.1.4 require, require_once, include, include_once fonksiyonları

require, require_once, include, include_once fonksiyonlarının performans ölçümü için ilgili dosyalara 5 farklı denemede, saniyede 100 olmak üzere, 5 saniye boyunca toplam 500 istek gönderilmiştir. İlgili dosyaların hepsi, her birinde boş ve yeni bir sınıfın tanımlandığı, dört farklı dosyayı içermektedir. JMeter ve ab test ölçüm sonuçları Tablo 4.10 ve 4.11'de yer almaktadır.

Tablo 4.10: include, include_once, require ve require_once fonksiyonlarının JMeter yanıt süresi ortalaması

	include (ms)	include_once (ms)	require (ms)	require_once (ms)
1. test	388	286	305	286
2. test	380	259	395	201
3. test	203	199	283	214
4. test	357	314	250	229
5.test	245	310	221	243
Ortalama	314,6	274,2	290,8	234,6

Tablo 4.11: include, include_once, require ve require_once fonksiyonlarının ab yanıt süresi ortalaması

	include (ms)	include_once (ms)	require (ms)	require_once (ms)
1. test	0,811	0,686	0,842	0,687
2. test	0,811	0,700	0,850	0,690
3. test	0,810	0,655	0,842	0,695
4. test	0,830	0,710	0,825	0,680
5.test	0,859	0,695	0,819	0,710
Ortalama	0,824	0,689	0,836	0,692

4.1.1.5 for döngü uzunluğunun döngü içerisinde belirlenmesi

1.000 kere çalışacak bir for döngüsü içerisinde; 10 elemanlı bir dizinin elemanları for döngüsü içerisinde, döngünün tekrar sayısı count fonksiyonu ile dinamik olarak ve döngü dışında doğrudan hesaplanarak döndürülmüştür.

Döngü tekrar sayısının dinamik olarak hesaplandığı kod örneği aşağıdaki gibidir.

```
for ($i = 0; $i < 1000; $i++) {  
    $items = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);  
    for ($j = 0; $j < count($items); $j++) {}  
}
```

Döngü tekrar sayısının değişkene atanarak (sabit olarak) hesaplandığı kod örneği aşağıdaki gibidir.

```
for ($i = 0; $i < 1000; $i++) {  
    $items = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);  
    $counter = count($items);  
    for ($j = 0; $j < $counter; $j++) {}  
}
```

Dosyalara üç farklı istek gönderilmiş ve dosyaların çalışma sürelerinin ms türünden ölçümleri Tablo 4.12’de belirtilmiştir.

Tablo 4.12: for döngüsü uzunluğunun döngü içerisinde ve döngü dışarısında hesaplandığı dosyaların Xdebug Profiler sonuçları

	döngü tekrar sayısının dinamik olarak hesaplandığı (ms)	döngü tekrar sayısının değişkene atanarak hesaplandığı (ms)
1. İstek	235	54
2. İstek	242	36
3. İstek	234	45
Ortalama	237	45

Dosyalara gönderilen saniyede 10 istek ile toplam 100 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.13'te yer almaktadır.

Tablo 4.13: for döngüsü uzunluğunun döngü içerisinde ve döngü dışarısında hesaplandığı dosyaların JMeter yanıt süresi ortalaması

	döngü tekrar sayısının dinamik olarak hesaplandığı (ms)	döngü tekrar sayısının değişkene atanarak hesaplandığı (ms)
Ortalama	8	6

Dosyalara gönderilen anlık 10 istek ile toplam 100 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.14'te yer almaktadır.

Tablo 4.14: for döngüsü uzunluğunun döngü içerisinde ve döngü dışarısında hesaplandığı dosyaların ab yanıt süresi

	döngü tekrar sayısının dinamik olarak hesaplandığı (ms)	döngü tekrar sayısının değişkene atanarak (sabit olarak) hesaplandığı (ms)
Ortalama	4,12	2,32

4.1.1.6 Dizi elemanlarına döngüler ile erişim (for, foreach, while)

10.000 kere çalışacak bir for döngüsü içerisinde, 10 elemanlı bir dizinin elemanları for, foreach ve while döngüsü ile ekrana yazdırılmıştır.

Dosyalara üç farklı istek gönderilmiş ve dosyaların çalışma sürelerinin ms türünden ölçümleri Tablo 4.15'te belirtilmiştir.

Tablo 4.15: for, while ve foreach fonksiyonlarının Xdebug Profiler sonuçları

	for (ms)	while (ms)	foreach (ms)
1. İstek	292	282	56
2. İstek	310	281	39
3. İstek	280	297	58
Ortalama	294,00	286,67	51

Dosyalara gönderilen saniyede 10 istek ile toplam 100 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.16’te yer almaktadır.

Tablo 4.16: for, while ve foreach fonksiyonlarının JMeter yanıt süresi ortalaması

	for (ms)	while (ms)	foreach (ms)
Ortalama	37	36	33

Dosyalara gönderilen anlık 10 istek ile toplam 100 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.17’de yer almaktadır.

Tablo 4.17: for, while ve foreach fonksiyonlarının ab yanıt süresi ortalaması

	for (ms)	while (ms)	foreach (ms)
Ortalama	33,504	35,358	32,788

4.1.1.7 if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörü

1.000.000 kere çalışacak bir for döngüsü içerisinde, integer tipinde bir değişken oluşturulmuş ve if ifadesi içerisinde eşitlik ile denklik operatörleri kullanılmıştır.

Eşitlik kontrolü yapılan kod örneği aşağıdaki gibidir.

```
$x = 1;
for ($i = 0; $i < 1000000; $i++) {
    if ($x == "1") {}
}
```

Denklik kontrolü yapılan kod örneği aşağıdaki gibidir.

```
$x = 1;
for ($i = 0; $i < 1000000; $i++) {
    if ($x === 1) {}
}
```

Dosyalara üç farklı istek gönderilmiş ve dosyaların çalışma sürelerinin ölçümleri gerçekleştirilmiş ve test sonuçları Tablo 4.18’de belirtilmiştir.

Tablo 4.18: if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörlerinin Xdebug Profiler sonuçları

	== (eşitlik) (ms)	=== (denklik) (ms)
1. İstek	326	248
2. İstek	333	226
3. İstek	351	288
Ortalama	336,67	254

Dosyalara gönderilen saniyede 10 istek ile toplam 100 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.19’da yer almaktadır.

Tablo 4.19: if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörlerinin JMeter yanıt süresi ortalaması

	== (eşitlik) (ms)	=== (denklik) (ms)
Ortalama	63	37

Dosyalara gönderilen anlık 10 istek ile toplam 100 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.20’de yer almaktadır.

Tablo 4.20: if koşul ifadesi içerisinde eşitlik (==) ve denklik (===) operatörlerinin ab yanıt süresi ortalaması

	== (eşitlik) (ms)	=== (denklik) (ms)
Ortalama	61,99	29,08

4.1.1.8 Dizi elemanlarına anahtar ile erişim

100.000 kere çalışacak bir for döngüsü içerisinde, üç elemanı bulunan ve anahtarları string yapıda bulunan bir dizinin elemanına erişim; tek tırnak kullanarak (`$dizi['anahtar']`) ve kullanılmadan (`$dizi[anahtar]` formatında) denenmiştir.

Not: Dizi elemanına, `$dizi[anahtar]` şeklinde erişim denemesinde, PHP yorumlayıcısı anahtarı bir PHP sabiti olarak değerlendirecek ve “Notice” tipinde bir uyarı verecektir. Hata ve uyarı mesajları gizlenmiş bir uygulamada, genellikle bu uyarı fark edilmeden işlemlere devam edilmektedir.

Dosyalara üç farklı istek gönderilmiş, dosyaların çalışma sürelerinin ölçümleri gerçekleştirilmiş ve test sonuçları Tablo 4.21’de belirtilmiştir.

Tablo 4.21: Dizi elemanlarına anahtar ile erişimde Xdebug Profiler sonuçları

	<code>\$dizi['anahtar']</code> (ms)	<code>\$dizi[anahtar]</code> (ms)
1. İstek	40	165
2. İstek	60	151
3. İstek	57	124
Ortalama	52,33	146,67

Dosyalara gönderilen saniyede 10 istek ile toplam 100 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.22’de yer almaktadır.

Tablo 4.22: Dizi elemanlarına anahtar ile erişimde JMeter yanıt süresi ortalaması

	<code>\$dizi['anahtar']</code> (ms)	<code>\$dizi[anahtar]</code> (ms)
Ortalama	23	69

Dosyalara gönderilen anlık 10 istek ile toplam 100 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.23’te yer almaktadır.

Tablo 4.23: Dizi elemanlarına anahtar ile erişimde ab yanıt süresi ortalaması

	\$dizi['anahtar'] (ms)	\$dizi[anahtar] (ms))
Ortalama	15,236	53,022

4.1.1.9 isset ve strlen fonksiyonları

100.000 kere çalışacak bir for döngüsü içerisinde, döngü dışında tanımlanmış bir değişkenin karakter sayısına göre koşul sorgulaması yapılmıştır. Bunun için isset ve strlen fonksiyonları kullanılmıştır.

isset fonksiyonu ve karakter sayısı koşul sorgusu yapılan kod örneği aşağıda yer aldığı gibidir.

```
$string = "Merhaba Dünya";  
for ($i = 0; $i < 100000; $i++) {  
    if (! isset($foo{10})) {}  
}
```

strlen fonksiyonu ile karakter sayısı koşul sorgulaması yapılan kod örneği aşağıda yer aldığı gibidir.

```
$string = "Merhaba Dünya";  
for ($i = 0; $i < 100000; $i++) {  
    if (strlen($string) < 10) {}  
}
```

Dosyalara üç farklı istek gönderilmiş, dosyaların çalışma sürelerinin ölçümleri gerçekleştirilmiş ve test sonuçları Tablo 4.24'te belirtilmiştir.

Tablo 4.24: isset ve strlen fonksiyonlarının Xdebug Profiler sonuçları

	isset (ms)	strlen (ms)
1. İstek	69	1.987
2. İstek	49	1.985
3. İstek	41	2.018
Ortalama	53	1.996,67

Xdebug Profiler sonuçlarına göre, 100.000 adet strlen metodu çalışma süresi yaklaşık olarak 350 ms, dosyanın çalışma süresi ise 1.630 ms olmuştur.

Dosyalara gönderilen saniyede 10 istek ile toplam 100 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.25'te yer almaktadır.

Tablo 4.25: isset ve strlen fonksiyonlarının JMeter yanıt süresi ortalaması

	isset (ms)	strlen (ms)
Ortalama	6	13

Dosyalara gönderilen anlık 10 istek ile toplam 100 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.26'da yer almaktadır.

Tablo 4.26: isset ve strlen fonksiyonlarının ab yanıt süresi ortalaması

	isset (ms)	strlen (ms)
Ortalama	4,07	10,53

4.1.1.10 Nesne elemanına doğrudan veya metot yardımı (Getter) ile erişim

100.000 kere çalışacak bir for döngüsü içerisinde, döngü dışında tanımlanmış sınıfın ve bu sınıfın bir örneği olan nesnenin elemanına iki farklı şekilde erişim sağlanmıştır. İlk olarak nesne elemanı public görünümlü tanımlanmış ve doğrudan erişim sağlanmıştır. İkinci olarak ise nesne elemanı private görünümlü olarak tanımlanmış ve public tanımlanan metot ile elemanın değeri döndürülmüştür.

Nesne elemanına doğrudan erişim yapılan kod örneği aşağıda yer aldığı gibidir.

```
for ($i = 0; $i < 100000; $i++) {  
    echo $test->text;  
}
```

Nesne elemanına metot yardımıyla erişim yapılan kod örneği aşağıda yer aldığı gibidir.

```
for ($i = 0; $i < 100000; $i++) {  
    echo $test->getText();  
}
```

Dosyalara üç farklı istek gönderilmiş, dosyaların çalışma sürelerinin ölçümleri gerçekleştirilmiş ve test sonuçları Tablo 4.27’de belirtilmiştir.

Tablo 4.27: Nesne elemanına doğrudan veya metot yardımı ile erişimde Xdebug Profiler sonuçları

	Doğrudan erişim (ms)	Metot ile erişim (ms)
1. İstek	76	3.724
2. İstek	76	3.463
3. İstek	120	3.484
Ortalama	90,67	3.557

Xdebug Profiler sonuçlarına göre, 100.000 adet getText() metodu çalışma süresi yaklaşık olarak 450 ms, dosyanın çalışma süresi ise 3.050 ms olmuştur.

Dosyalara gönderilen saniyede 10 istek ile toplam 100 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.28’de yer almaktadır.

Tablo 4.28: Nesne elemanına doğrudan veya metot yardımı ile erişimde JMeter yanıt süresi ortalaması

	Doğrudan erişim (ms)	Metot ile erişim (ms)
Ortalama	53	62

Dosyalara gönderilen anlık 10 istek ile toplam 100 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.29’da yer almaktadır.

Tablo 4.29: Nesne elemanına doğrudan veya metot yardımı ile erişimde ab yanıt süresi ortalaması

	Doğrudan erişim (ms)	Metot ile erişim (ms)
Ortalama	37,67	46,31

4.1.2 Önbellek Mekanizmaları Testleri

4.1.2.1 İşlem kodu önbellekleme (opcode caching) mekanizmaları

Ubuntu sanal sunucusunda; Wordpress'in 4.4.2 numaralı versiyonu²⁵ indirilmiş ve “/var/www/html/wordpress/” klasörü içerisine kurulumu yapılmıştır. Wordpress yönetici panelinden 10 adet “Test” başlıklı ve 300 karakterden oluşan içerik eklenmiştir. Bunun dışında Wordpress'in sistemsal ayarları dahil, hiçbir değişiklik yapılmamıştır. Zend OPcache ve XCache performans testleri bu Wordpress bloğu üzerinde uygulanmıştır.

4.1.2.1.1 OPcache

OPcache modülü; PHP 5.5 ve sonraki sürümlerde otomatik yüklü fakat pasif bir şekilde gelmektedir. OPcache modülü pasif durumda iken ve aşağıda yer alan konfigürasyon ayarları ile aktifleştirildikten sonra, Wordpress ile oluşturulan bloğun ana sayfasına ve oluşturulan test içerik sayfalarından birine istekler gönderilmiştir.

php.ini dosyası opcache yapılandırma ayarları aşağıda yer aldığı gibi düzenlenmiştir.

```
opcache.enable_cli=0
opcache.memory_consumption=128
opcache.interned_strings_buffer=4
opcache.max_accelerated_files=4000
opcache.max_wasted_percentage=5
opcache.use_cwd=1
opcache.validate_timestamps=1
opcache.revalidate_freq=240
opcache.revalidate_path=0
opcache.save_comments=1
opcache.load_comments=1
opcache.fast_shutdown=0
opcache.enable_file_override=0
```

²⁵ <http://wordpress.org/latest.tar.gz> (erişim tarihi 12.03.2016)

Ana sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.30’da yer almaktadır.

Tablo 4.30: OPcache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin JMeter yanıt süresi ortalaması

	OPcache pasif (ms)	OPcache aktif (ms)
1. istek	181	75
2. istek	180	82
3. istek	173	80
Ortalama	178	79

OPcache pasif durumdayken istek sayısı saniyede 10, toplamda 100’e çıkarıldığında yanıt süresi ortalaması 6.000 ms seviyelerine çıkmaktadır.

OPcache pasif durumdayken, ana sayfaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 81 seviyelerine kadar çıkmıştır. OPcache aktif durumdayken ise CPU kullanım miktarı maksimum yüzde 36 seviyesinde olmuştur.

İçerik detay sayfasına gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.31’de yer almaktadır.

Tablo 4.31: OPcache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalaması

	OPcache pasif (ms)	OPcache aktif (ms)
1. istek	154	63
2. istek	151	62
3. istek	150	60
Ortalama	151,67	61,67

OPcache pasif durumdayken, içerik detay sayfaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 78 seviyelerine kadar çıkmıştır. OPcache aktif durumdayken ise CPU kullanım miktarı maksimum yüzde 27 seviyesinde olmuştur.

Ana sayfaya gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.32’de yer almaktadır.

Tablo 4.32: Opcache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin ab yanıt süresi ortalaması

	OPcache pasif (ms)	OPcache aktif (ms)
1. istek	178,27	64,08
2. istek	177,51	65,66
3. istek	174,99	66,74
Ortalama	176,92	65,49

İçerik detay sayfasına gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.33’te yer almaktadır.

Tablo 4.33: Opcache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin ab yanıt süresi ortalaması

	OPcache pasif (ms)	OPcache aktif (ms)
1. istek	148,89	39,98
2. istek	149,35	42,05
3. istek	150,15	41,22
Ortalama	149,46	41,08

4.1.2.1.2 XCache

SSH ile Ubuntu sanal sunucusuna “sudo apt-get install php5-xcache” komutuyla XCache kurulumu yapılmıştır.

phpinfo() fonksiyonun çıktısına göre bazı XCache ayarları Tablo 4.34’te yer aldığı gibidir.

Tablo 4.34: XCache ayarlarının phpinfo() çıktısı

XCache Version	3.2.0
Modules Built	acher optimizer coverager assembler encoder decoder
xcache.acher	On
xcache.shm_scheme	mmap
xcache.size	60M
xcache.slots	8K
xcache.ttl	0
xcache.optimizer	Off

XCache modülü pasif durumda iken ve aşağıda yer alan yapılandırma ayarları ile aktifleştirildikten sonra, Wordpress ile oluşturulan bloğun ana sayfasına ve oluşturulan test içerik sayfalarından birine istekler gönderilmiştir.

Ana sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.35’te yer almaktadır.

Tablo 4.35: XCache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin JMeter yanıt süresi ortalaması

	XCache pasif (ms)	XCache aktif (ms)
1. istek	181	92
2. istek	180	84
3. istek	173	87
Ortalama	178	87,67

XCache pasif durumdayken, ana sayfaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 81 seviyelerine kadar çıkmıştır. XCache aktif durumdayken ise CPU kullanım miktarı maksimum yüzde 37 seviyesinde olmuştur.

İçerik detay sayfasına gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.36’da yer almaktadır.

Tablo 4.36: XCache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalaması

	XCache pasif (ms)	XCache aktif (ms)
1. istek	154	66
2. istek	151	68
3. istek	150	64
Ortalama	151,67	66

XCache pasif durumdayken, içerik detay sayfaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 78 seviyelerine kadar çıkmıştır. XCache aktif durumdayken ise CPU kullanım miktarı maksimum yüzde 28 seviyesinde olmuştur.

Ana sayfaya gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.37’de yer almaktadır.

Tablo 4.37: XCache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin ab yanıt süresi ortalaması

	XCache pasif (ms)	XCache aktif (ms)
1. istek	178,27	81,55
2. istek	177,51	77,84
3. istek	174,99	77,48
Ortalama	176,92	78,96

İçerik detay sayfasına gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.38’de yer almaktadır.

Tablo 4.38: XCache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin ab yanıt süresi ortalaması

	XCache pasif (ms)	XCache aktif (ms)
1. istek	148,89	50,98
2. istek	149,35	50,96
3. istek	150,15	50,52
Ortalama	149,46	50,82

4.1.2.2 Memory caching mekanizmaları

4.1.2.2.1 Memcached

launchpad.net²⁶ web sitesinden MySQL için hazırlanmış test “employee” veri tabanı indirilip, içeri aktarılmıştır. Memcached’in performansa etkisi; her istekte employees tablosundan "SELECT * FROM employees LIMIT 0, 10000" sorgusu ile 10.000 kaydın okunduğu dosya ve sorgunun bir kere çalıştırılıp sonuçlarının Memcached’e aktarıldığı, sonraki isteklerin Memcached tarafından karşılandığı dosyaya yapılacak isteklerle ölçülmüştür.

Ubuntu sanal sunucusuna kurulan Memcached’in özellikleri Tablo 4.39’da belirtilmiştir.

Tablo 4.39: Sunucu üzerindeki Memcached uygulamasının özellikleri

Version	2.2.0
Port	11211
Memory	128 MB
libmemcached version	1.0.18

MySQL sunucusuna bağlantı, mysql sınıfı yardımıyla gerçekleştirilmiş olup; veriler ilgili sınıfın “query” isimli metoduyla sorgulanıp, “fetch_object” metoduyla çekilmiştir. Verilerin MySQL ve Memcached’den çekildiği sayfalara gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.40’ta yer almaktadır.

Tablo 4.40: MySQL ve Memcached ile veri çekilen sayfaların JMeter yanıt süresi ortalaması

	MySQL (ms)	Memcached (ms)
1. istek	62	35
2. istek	59	35
3. istek	63	35
Ortalama	61,33	35

²⁶ <https://launchpad.net/test-db/> (erişim tarihi 16.03.2016)

Verilerin MySQL ve Memcached'den çekildiği sayfalara gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.41'de yer almaktadır.

Tablo 4.41: MySQL ve Memcached ile veri çekilen sayfaların ab yanıt süresi ortalaması

	MySQL (ms)	Memcached (ms)
1. istek	40,84	11,28
2. istek	41,50	10,54
3. istek	41,24	10,44
Ortalama	41,19	10,75

Test işlemleri esnasında; sunucudaki kaynak tüketimi “top” komutuyla incelenmiş olup, mysql işleminin CPU kullanımı maksimum yüzde 6,5, bellek kullanımı yüzde 13,5 seviyelerini görmüşken, memcached işleminin CPU kullanımı maksimum yüzde 0,7, bellek kullanımı ise yüzde 0,8 seviyelerinde olmuştur.

4.2 VERİ TABANI PERFORMANS TESTLERİ

4.2.1 MySQL Sorgu Performans İncelemeleri

MySQL sorgu performansı incelemelerinde, daha önceden launchpad.net web sitesi üzerinden indirilmiş olan “employee” veri tabanı kullanılmıştır. PHP ve MySQL tabanlı uygulamalarda en sık yapılan MySQL sorgu hatalarının ve alternatif çözümlerin “mysqlslap” aracılığıyla performans incelemeleri gerçekleştirilmiştir.

4.2.1.1 SELECT sorgularında sütun kısıtlaması

Bir tablo üzerinde yapılacak SELECT sorgusunda, tüm sütunlar yerine sadece ihtiyaç olan sütunun verisinin çekilmesinin performansa etkisi incelenmiştir. Performans testlerinin simülasyonu anlık 10 istek ile toplamda 5 defa tekrarlanacak şekilde gerçekleştirilmiştir.

Tablodaki tüm sütunların verilerinin çekildiği sorgu için uygulanan performans test komutu aşağıdaki gibidir.

```
mysqlslap --user=root --password --host=localhost --  
concurrency=10 --iterations=5 --create-schema=employees --  
query="SELECT * FROM employees;"
```

Tablodaki tüm sütunların çekildiği SELECT sorgusunun mysqlslap çalışma süreleri Tablo 4.42’de yer almaktadır.

Tablo 4.42: Tablodaki tüm sütunlardan veri çekilen SELECT sorgusunun mysqlslap çalışma süresi

	İşlem süresi (saniye)
Her bir tekrarda, sorguların ortalama çalışma süresi	2,299
Her bir tekrarda, sorguların minimum çalışma süresi	2,229
Her bir tekrarda, sorguların maksimum çalışma süresi	2,353

Sadece “emp_no” sütununa ait verilerin çekildiği sorgu için uygulanan performans test komutu aşağıda yer almaktadır.

```
mysqlslap --user=root --password --host=localhost --  
concurrency=10 --iterations=5 --create-schema=employees --  
query="SELECT emp_no FROM employees;"
```

Tablodaki “emp_no” sütununun çekildiği SELECT sorgusunun mysqlslap çalışma süreleri Tablo 4.43’te yer almaktadır.

Tablo 4.43: Tablodaki tek sütundan veri çekilen SELECT sorgusunun mysqlslap çalışma süresi

	İşlem süresi (saniye)
Her bir tekrarda, sorguların ortalama çalışma süresi	1,078
Her bir tekrarda, sorguların minimum çalışma süresi	1,043
Her bir tekrarda, sorguların maksimum çalışma süresi	1,149

4.2.1.2 JOIN kullanımı

“employee” veri tabanında, çalışanların şuan çalıştıkları departman isimleriyle birlikte, kendi isim ve soy isimlerinin çekilmesini gerektiren durumlarda JOIN terimi kullanılarak ve kullanılmadan yapılan işlemlerdeki performanslar ölçülmüştür.

INNER JOIN terimi yardımıyla yazılan tek MySQL sorgusu ile verilerin çekildiği dosyaya ve önce tüm çalışanların çekildiği, ardından bir döngü yardımıyla listelendiği ve çalışanın “emp_no” bilgisine göre ilgili departman bilgilerinin çekildiği dosyaya ab ile anlık 1 adet olmak üzere toplam 10 istek gönderilmiştir.

INNER JOIN ile yazılan SQL sorgusu aşağıdaki gibidir.

```
SELECT first_name, last_name, dept_name FROM employees  
AS e INNER JOIN dept_emp AS de ON de.emp_no = e.emp_no  
INNER JOIN departments AS d ON d.dept_no = de.dept_no  
WHERE de.to_date > NOW() LIMIT 1000;
```

Adım adım uygulanan SQL sorguları ise aşağıdaki gibidir.

```
SELECT emp_no, first_name, last_name FROM employees;
```

Yukarıda sorgunun çıktısı döngüye sokularak her sonuç için ayrı ayrı çalıştırılan aşağıdaki SQL sorgusu çalıştırılmıştır.

```
SELECT dept_no FROM dept_emp WHERE emp_no = '$emp_no'  
AND to_date > NOW();
```

Yukarıdaki sorgudan elde edilen “dept_no” verisi ile “department” tablosundan “dept_name” verisinin çekildiği SQL sorgusu da aşağıdaki gibidir.

```
SELECT dept_name FROM departments WHERE dept_no =  
'$dept_no';
```

Verilerin INNER JOIN kullanılarak ve kullanılmadan çekildiği sayfalara gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.44'te yer almaktadır.

Tablo 4.44: INNER JOIN kullanılarak ve kullanılmadan veri çekilen sayfaların ab yanıt süresi ortalaması

	INNER JOIN sorgusu kullanılan dosya (ms)	Çoklu sorgu kullanılan dosya (ms)
1. istek	1.252,17	60.265,45
2. istek	1.245,17	60.649,47
3. istek	1.198,07	65.709,76
Ortalama	1.231,80	62.208,23

4.2.1.3 INNER JOIN – LEFT JOIN kullanımı

“employee” veri tabanında, çalışanların şuan çalıştıkları departman isimleriyle birlikte, kendi isim ve soy isimlerinin çekildiği MySQL sorgusu INNER JOIN ve LEFT JOIN kullanılarak uygulanmıştır. Performans testleri söz konusu sorgulara, anlık 10 istek ile 5 kere tekrarlan simülasyon ile uygulanmıştır.

INNER JOIN ile tabloları birleştirerek uygulanan performans test komutu aşağıdaki gibidir.

```
mysqlslap --user=root --password --host=localhost --
concurrency=10 --iterations=5 --create-schema=employees --
query="SELECT first_name, last_name, dept_name FROM
employees AS e INNER JOIN dept_emp AS de ON de.emp_no =
e.emp_no INNER JOIN departments AS d ON d.dept_no =
de.dept_no WHERE de.to_date > NOW();"

```

INNER JOIN ile tabloları birleştiren SQL sorgusunun mysqlslap çalışma süreleri Tablo 4.45'te yer almaktadır.

Tablo 4.45: INNER JOIN ile tabloları birleştiren SELECT sorgusunun mysqlslap çalışma süresi

	İşlem süresi (saniye)
Her bir tekrarda, sorguların ortalama çalışma süresi	5,807
Her bir tekrarda, sorguların minimum çalışma süresi	5,687
Her bir tekrarda, sorguların maksimum çalışma süresi	6,024

LEFT JOIN ile tabloları birleştirerek uygulanan performans test komutu aşağıda yer almaktadır.

```
mysqlslap --user=root --password --host=localhost --  
concurrency=10 --iterations=5 --create-schema=employees --  
query="SELECT first_name, last_name, dept_name FROM  
employees AS e INNER JOIN dept_emp AS de ON de.emp_no =  
e.emp_no INNER JOIN departments AS d ON d.dept_no =  
de.dept no WHERE de.to date > NOW();" ;"
```

LEFT JOIN ile tabloları birleştiren SQL sorgusunun mysqlslap çalışma süreleri Tablo 4.46'da yer almaktadır.

Tablo 4.46: LEFT JOIN ile tabloları birleştiren SELECT sorgusunun mysqlslap çalışma süresi

	İşlem süresi (saniye)
Her bir tekrarda, sorguların ortalama çalışma süresi	8,670
Her bir tekrarda, sorguların minimum çalışma süresi	8,373
Her bir tekrarda, sorguların maksimum çalışma süresi	9,545

4.2.1.4 LIMIT kullanımı

MySQL sorgularında, LIMIT kullanarak sonucu kısıtlamanın performans üzerindeki etkisi; bir önceki test işleminde kullanılan, çalışanların şuan çalıştıkları departman isimleriyle birlikte, kendi isim ve soy isimlerinin INNER JOIN ile çekildiği sorgu ile birleştirilerek ölçülmüştür. Limitleme değeri 1000 olarak belirlenmiş ve performans testleri söz konusu sorgulara, anlık 10 istek ile 5 kere tekrarlan simülasyon ile uygulanmıştır.

LIMIT kullanılmadan uygulanan performans test komutu aşağıda yer almaktadır.

```
mysqlslap --user=root --password --host=localhost --
concurrency=10 --iterations=5 --create-schema=employees --
query="SELECT first_name, last_name, dept_name FROM
employees AS e INNER JOIN dept_emp AS de ON de.emp_no =
e.emp_no INNER JOIN departments AS d ON d.dept_no =
de.dept no WHERE de.to date > NOW();"

```

LIMIT kullanılmadan verilerin çekildiği SQL sorgusunun mysqlslap çalışma süreleri Tablo 4.47’de yer almaktadır.

Tablo 4.47: LIMIT kullanılmayan SELECT sorgusunun mysqlslap çalışma süresi

	İşlem süresi (saniye)
Her bir tekrarda, sorguların ortalama çalışma süresi	5,807
Her bir tekrarda, sorguların minimum çalışma süresi	5,687
Her bir tekrarda, sorguların maksimum çalışma süresi	6,024

LIMIT kullanarak uygulanan performans test komutu aşağıda belirtilmiştir.

```
mysqlslap --user=root --password --host=localhost --
concurrency=10 --iterations=5 --create-schema=employees --
query="SELECT first_name, last_name, dept_name FROM
employees AS e INNER JOIN dept_emp AS de ON de.emp_no =
e.emp_no INNER JOIN departments AS d ON d.dept_no =
de.dept no WHERE de.to date > NOW() LIMIT 1000;"

```

LIMIT kullanılarak verilerin çekildiği SQL sorgusunun mysqlslap çalışma süreleri Tablo 4.48’de yer almaktadır.

Tablo 4.48: LIMIT kullanılan SELECT sorgusunun mysqlslap çalışma süresi

	İşlem süresi (saniye)
Her bir tekrarda, sorguların ortalama çalışma süresi	0,031
Her bir tekrarda, sorguların minimum çalışma süresi	0,027
Her bir tekrarda, sorguların maksimum çalışma süresi	0,038

4.2.2 NoSQL Performans Testleri

4.2.2.1 Redis

“employee” veri tabanı Redis performans testlerinde de kullanılmıştır. Redis’in performansa etkisinin ölçüm yöntemi; Memcached performans testine benzer şekilde gerçekleştirilmiştir. Her istekte employees tablosundan "SELECT * FROM employees LIMIT 0, 10000" sorgusu ile 10.000 kaydın okunduğu dosya ve sorgunun bir kere çalıştırılıp sonuçlarının Redis’e aktarıldığı, sonraki isteklerin Redis tarafından karşılandığı dosyaya yapılacak isteklerle ölçülmüştür.

Ubuntu sanal sunucusuna kurulan Redis’in özellikleri Tablo 4.49’da belirtilmiştir.

Tablo 4.49: Sunucu üzerindeki Redis uygulamasının özellikleri

Redis version	3.0.7
PHP Redis version	2.2.7
Port	6379

MySQL sunucusuna bağlantı, mysqli sınıfı yardımıyla gerçekleşmiş olup; veriler ilgili sınıfın “query” isimli metoduyla sorgulanıp, “fetch_object” metoduyla çekilmiştir. Redis sunucusuna bağlantı için ise Github üzerindeki “predis” kütüphanesi²⁷ kullanılmıştır. Verilerin MySQL ve Redis’ten çekildiği sayfalara gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter yanıt sürelerinin ortalaması Tablo 4.50’de yer almaktadır.

Tablo 4.50: MySQL ve Redis ile veri çekilen sayfaların JMeter yanıt süresi ortalaması

	MySQL (ms)	Redis (ms)
1. istek	59	16
2. istek	65	15
3. istek	63	16
Ortalama	62,33	15,67

²⁷ <https://github.com/nrk/predis> (erişim tarihi: 22.03.2016)

Verilerin MySQL ve Redis'ten çekildiği sayfalara gönderilen anlık 5 istek ile toplam 50 istek sonrası, ab yanıt sürelerinin ortalaması Tablo 4.51'de yer almaktadır.

Tablo 4.51: MySQL ve Redis ile veri çekilen sayfaların ab yanıt süresi ortalaması

	MySQL (ms)	Redis (ms)
1. istek	41,04	4,99
2. istek	41,49	5,30
3. istek	40,95	5,35
Ortalama	41,16	5,21

Test işlemleri esnasında; sunucudaki kaynak tüketimi “top” komutuyla incelenmiş olup, mysql işleminin CPU kullanımı maksimum yüzde 7,3, bellek kullanımı yüzde 13,5 seviyelerini görmüşken, “redis-server” işleminin maksimum CPU ve bellek kullanımı ise yüzde 0,7 seviyelerinde olmuştur.

4.2.2.2 MongoDB

MongoDB ve MySQL performans testleri, “employee” veri tabanının “employees” tablosu ile sonradan oluşturulan “article” isimli veri tabanının, Tablo 4.52’de yapısı verilen “articles” isimli tablosu üzerinde gerçekleştirilmiştir. Bu tablolardaki veriler, aynı isim ve yapıdaki MongoDB koleksiyonlarına da eklenmiştir.

Tablo 4.52: articles tablosu yapısı

Sütun	Yapı
article_id	INT (11) – Primary Key (P.K.) – AUTO INCREMENT
title	VARCHAR (255)
text	TEXT

employees tablosu üzerinde 1.000 adet INSERT işleminin yapıldığı sayfaya gönderilen isteğin ab yanıt sürelerinin ortalaması Tablo 4.53’te yer almaktadır.

Tablo 4.53: MySQL ve MongoDB’de INSERT işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	MongoDB (ms)
1. istek	87,01	188,01
2. istek	74,04	112,06
3. istek	73,00	124,01
Ortalama	78,02	141,36

employees tablosu üzerinde tüm verinin SELECT işlemi ile çekildiği sayfaya gönderilen isteğin ab test sonuçları Tablo 4.54’te yer almaktadır.

Tablo 4.54: MySQL ve MongoDB’de SELECT işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	MongoDB (ms)
1. istek	1.497,09	1.811,10
2. istek	1.378,08	1.652,09
3. istek	1.272,07	1.614,09
Ortalama	1.382,41	1.692,43

employees tablosu üzerinde 1.000 adet UPDATE işleminin yapıldığı sayfaya gönderilen isteğin ab test sonuçları Tablo 4.55’te yer almaktadır.

Tablo 4.55: MySQL ve MongoDB’de UPDATE işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	MongoDB (ms)
1. istek	6.535,37	513,03
2. istek	6.076,35	454,03
3. istek	6.179,35	477,03
Ortalama	6.263,69	481,36

articles tablosuna, php for döngüsü yardımıyla; “title” alanı “Lorem Ipsum”, “text” alanı on iki paragraf ve 1.000 kelimedenden oluşan 1.000 adet veri eklenmiştir. Aşağıda yer alan performans işlemleri bu veriler kullanılarak gerçekleştirilmiştir.

articles tablosu üzerinde 1.000 adet INSERT işleminin yapıldığı sayfaya gönderilen isteğin ab test sonuçları Tablo 4.56’da belirtilmiştir.

Tablo 4.56: MySQL ve MongoDB’de büyük boyutlu verinin INSERT işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	MongoDB (ms)
1. istek	50.754,90	226,01
2. istek	54.226,10	201,01
3. istek	54.468,12	202,01
Ortalama	53.149,71	209,68

5.000 satır içeriği bulunan articles tablosu üzerinde tüm verinin SELECT işlemi ile çekildiği sayfaya gönderilen isteğin ab test sonuçları Tablo 4.57’de yer almaktadır.

Tablo 4.57: MySQL ve MongoDB’de büyük boyutlu verilerin SELECT işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	MongoDB (ms)
1. istek	3,00	67,00
2. istek	3,00	59,00
3. istek	2,00	61,00
Ortalama	2,67	62,33

articles tablosu üzerinde 1.000 adet UPDATE işleminin yapıldığı sayfaya gönderilen isteğin ab test sonuçları Tablo 4.58’de belirtilmiştir.

Tablo 4.58: MySQL ve MongoDB’de büyük boyutlu verilerin UPDATE işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	MongoDB (ms)
1. istek	2.118,12	424,03
2. istek	2.089,12	414,02
3. istek	2.113,12	386,02
Ortalama	2.106,79	408,02

4.2.3 Arama Motoru Sunucuları

4.2.3.1 Sphinx

“employee” veri tabanı Sphinx arama motoru performans testlerinde de kullanılmıştır. “employees” tablosunda “Mary” isimli çalışanların MySQL ve Sphinx sorgularıyla listelendiği dosyalara gönderilen istekler sonrası performans ölçülmüştür. Söz konusu sorgular PHP yardımıyla çalıştırılmıştır ve 224 adet sonuç; foreach döngüsü ile ekrana yazdırılmıştır.

Ubuntu sanal sunucusu üzerine kurulan Sphinx modülünün özellikleri Tablo 4.59’da yer aldığı gibidir.

Tablo 4.59: Sunucu üzerindeki Sphinx modülünün özellikleri

Sphinx version	2.0.4-id64-release
PHP sphinx version	1.3.3
Port	9312

Sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası JMeter test ortalaması Tablo 4.60’da belirtilmiştir.

Tablo 4.60: MySQL ve Sphinx’te veri çekme işlemlerinin JMeter yanıt süresi ortalaması

	MySQL (ms)	Sphinx (ms)
1. istek	107	18
2. istek	107	14
3. istek	105	17
Ortalama	106,33	16,33

Aranan verilerin MySQL sunucusundan çekildiği dosyaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 48 seviyelerine kadar çıkmıştır. Verilerin Sphinx ile çekildiği dosyaya gönderilen istekler esnasında ise CPU kullanım miktarı maksimum yüzde 1,7 seviyesinde olmuştur.

Sayfaya gönderilen anlık 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.61’de belirtilmiştir.

Tablo 4.61: MySQL ve Sphinx’te veri çekme işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	Sphinx (ms)
1. istek	95,35	4,24
2. istek	94,31	2,86
3. istek	94,85	3,58
Ortalama	94,84	3,56

4.2.3.2 Elasticsearch

“employee” veri tabanı Elasticsearch performans testlerinde de kullanılmıştır. “employees” tablosunda “Mary” isimli çalışanların MySQL ve Elasticsearch sorgularıyla listelendiği dosyalara gönderilen istekler sonrası performans ölçülmüştür. Söz konusu sorgular PHP yardımıyla çalıştırılmıştır ve 224 adet sonuç; foreach döngüsü ile ekrana yazdırılmıştır.

Ubuntu sanal sunucusu üzerine kurulan Elasticsearch modülünün özellikleri Tablo 4.62’de yer aldığı gibidir.

Tablo 4.62: Sunucu üzerindeki Elasticsearch modülünün özellikleri

Elasticsearch version	1.7.2
PHP Elasticsearch kütüphanesi	elasticsearch/elasticsearch ~ 2.0
Port	9200

Ana sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası JMeter test ortalaması Tablo 4.63’te belirtilmiştir.

Tablo 4.63: MySQL ve Elasticsearch'te veri çekme işlemlerinin JMeter yanıt süresi ortalaması

	MySQL (ms)	Elasticsearch (ms)
1. istek	114	49
2. istek	110	55
3. istek	114	49
Ortalama	112,67	51

Aranan verilerin MySQL sunucusunda çekildiği dosyaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 48 seviyelerine kadar çıkmıştır. Verilerin Elasticsearch ile çekildiği dosyaya gönderilen istekler esnasında ise CPU kullanım miktarı maksimum yüzde 18,5 seviyesinde olmuştur.

Ana sayfaya gönderilen anlık 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.64'te belirtilmiştir.

Tablo 4.64: MySQL ve Elasticsearch'te veri çekme işlemlerinin ab yanıt süresi ortalaması

	MySQL (ms)	Elasticsearch (ms)
1. istek	93,92	24,60
2. istek	94,19	25,65
3. istek	92,60	24,05
Ortalama	93,57	24,77

4.3 WEB SUNUCU PERFORMANSI

4.3.1 Web Sunucu Yazılımları Performansı

Apache ve Nginx web sunucu yazılımlarının performans testleri; her iki sunucuya da PHP-FPM işleyicisi kurularak, daha önce Wordpress ile oluşturulan bloğun ana sayfasına gönderilen isteklerle gerçekleştirilmiştir. Sayfaya gönderilen anlık 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.65'te verilmiştir.

Tablo 4.65: Apache + PHP-FPM ve Nginx + PHP-FPM tabanlı sunucuda sayfaya gönderilen isteğin ab yanıt süresi ortalaması

	Apache + PHP-FPM (ms)	Nginx + PHP-FPM (ms)
1. istek	57,44	57,27
2. istek	57,91	55,45
3. istek	55,66	56,38
Ortalama	57,00	56,36

Yukarıdaki istekler gerçekleştirilirken, Apache web sunucusundaki maksimum CPU kullanım oranı yüzde 73, bellek kullanım oranı yüzde 25 seviyelerinde olmuşken; Nginx web sunucusundaki maksimum CPU kullanım oranı yüzde 86, bellek kullanım oranı yüzde 15 seviyelerinde olmuştur.

4.3.2 PHP İşleyici Performansı

Apache web sunucusunun varsayılan PHP işleyicisi DSO ile PHP-FPM performans testleri, Wordpress ile önceden oluşturulmuş olan bloğun ana sayfasına gönderilen isteklerle gerçekleştirilmiştir. Sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.66’da verilmiştir.

Tablo 4.66: Apache + DSO ve Apache + PHP-FPM tabanlı sunucuda sayfaya gönderilen isteğin ab yanıt süresi ortalaması

	Apache + DSO (ms)	Apache + PHP-FPM (ms)
1. istek	179,55	57,44
2. istek	172,64	57,91
3. istek	172,69	55,66
Ortalama	174,96	57,00

Yukarıdaki istekler gerçekleştirilirken, DSO işleyicisinin kullanıldığı sunucudaki maksimum CPU kullanım oranı yüzde 89, bellek kullanım oranı yüzde 18 seviyelerinde olmuşken; PHP-FPM işleyicisinin kullanıldığı sunucudaki maksimum CPU kullanım oranı yüzde 73, bellek kullanım oranı yüzde 25 seviyelerinde olmuştur.

4.3.3 PHP Yorumlayıcı Performansı

Sunucuda varsayılan olarak kurulu PHP yorumlayıcısı Zend Engine (versiyon 2.6.0) ile HHVM (versiyon 3.12) performans testleri, daha önce Wordpress ile oluşturulmuş olan bloğun ana sayfasına gönderilen isteklerle gerçekleştirilmiştir.

Sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.67’de verilmiştir.

Tablo 4.67: Apache + Zend Engine + PHP-FPM ve Apache + FCGI + HHVM tabanlı sunucuda sayfaya gönderilen isteğin ab yanıt süresi ortalaması

	Apache + Zend Engine + PHP-FPM (ms)	Apache + FCGI + HHVM
1. istek	57,44	0,73
2. istek	57,91	0,62
3. istek	55,66	0,33
Ortalama	57,00	0,56

Yukarıdaki istekler gerçekleştirilirken, Zend Engine yorumlayıcısının kullanıldığı sunucudaki maksimum CPU kullanım oranı yüzde 73, bellek kullanım oranı yüzde 25 seviyelerinde olmuşken; PHP-FPM işleyicisinin kullanıldığı sunucudaki maksimum CPU kullanım oranı yüzde 0,3 ve bellek kullanım oranı yüzde 1,7 seviyelerinde olmuştur.

4.3.4 Web Sunucu Hızlandırıcıları Performansı

4.3.4.1 Varnish Cache

Varnish pasif durumda iken ve aktifleştirildikten sonra, Wordpress ile oluşturulmuş olan bloğun ana sayfasına ve oluşturulan test sayfalarından birine istekler gönderilmiştir. Çalışmalar esnasında Varnish 80 numaralı portu kullanırken, Apache 8080 numaralı portu kullanmıştır.

/etc/default/varnish dosyasındaki yapılandırma ayarları aşağıdaki gibi düzenlenmiştir.

```
DAEMON_OPTS="-a :80 \  
-T localhost:6082 \  
-f /etc/varnish/default.vcl \  
-S /etc/varnish/secret \  
-s malloc,256m"
```

/etc/varnish/default.vcl dosyası içerisindeki yapılandırma ayarları aşağıda yer aldığı gibi düzenlenmiştir.

```
backend default {  
    .host = "127.0.0.1";  
    .port = "8080";  
}
```

Ana sayfaya gönderilen saniyede 5 istek ile toplam 50 istek sonrası, JMeter test ortalaması Tablo 4.68’de verilmiştir.

Tablo 4.68: Varnish Cache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin JMeter yanıt süresi ortalamaları

	Varnish pasif (ms)	Varnish aktif (ms)
1. istek	182	13
2. istek	184	12
3. istek	187	14
Ortalama	184,33	13

Varnish pasif durumdayken, ana sayfaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 81 seviyelerine kadar çıkmıştır. Varnish aktif durumdayken ise CPU kullanım miktarı maksimum yüzde 5,4 seviyesinde olmuştur.

İçerik detay sayfasına gönderilen saniyede 5 istek ile toplam 50 istek sonrası JMeter test ortalaması Tablo 4.69’da verilmiştir.

Tablo 4.69: Varnish Cache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalamaları

	Varnish pasif (ms)	Varnish aktif (ms)
1. istek	153	13
2. istek	154	14
3. istek	156	13
Ortalama	154,33	13,66

Varnish pasif durumdayken, içerik detay sayfaya gönderilen istekler esnasında, “top” komutu ile sunucunun kaynak tüketimi incelenmiş ve CPU kullanım miktarı yüzde 78 seviyelerine kadar çıkmıştır. Varnish aktif durumdayken ise CPU kullanım miktarı maksimum yüzde 4,7 seviyesinde olmuştur.

Ana sayfaya gönderilen anlık 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.70’te verilmiştir.

Tablo 4.70: Varnish Cache aktif ve pasif durumdayken ana sayfaya gönderilen isteklerin ab yanıt süresi ortalamaları

	Varnish pasif (ms)	Varnish aktif (ms)
1. istek	184,03	2,46
2. istek	183,01	2,18
3. istek	182,11	2,72
Ortalama	183,05	2,45

İçerik detay sayfasına gönderilen anlık 5 istek ile toplam 50 istek sonrası ab test ortalaması Tablo 4.71’de verilmiştir.

Tablo 4.71: Varnish Cache aktif ve pasif durumdayken detay sayfasına gönderilen isteklerin JMeter yanıt süresi ortalamaları

	Varnish pasif (ms)	Varnish aktif (ms)
1. istek	155,85	2,34
2. istek	152,41	2,38
3. istek	154,51	2,48
Ortalama	154,26	2,40

5. TARTIŞMA

Bu tez çalışmasında, günümüzde en sık kullanılan web yazılım dili PHP ve veri tabanı yönetim sistemlerinden biri olan MySQL tabanlı web uygulamalarında oluşabilecek performans düşüklüğünün sebeplerinin tespiti ve bu konuda alınabilecek bazı aksiyonlar araştırılmıştır. Uygulamanın performans artışının sadece donanımsal iyileştirme ile gerçekleştirilemeyeceği, yazılımsal yöntemlerin; performansa ne derece etki edeceği incelenmiştir. Bu çalışmada incelenen konuların çoğunun bir arada bulunduğu bir çalışmanın olmaması ve ülkemizde bu konularla ilgili yeterince kaynak bulunmaması; bu çalışmanın gerçekleştirilmesindeki büyük etmendir. Uygulama performansını etkileyen belirli bir konuda yapılmış çalışmaların eski tarihli olmaları; yeni sistemlerde uygulandığında farklı sonuçların ortaya çıkmasına neden olabilmektedir. Bu çalışma, performansa etki eden birçok farklı yöntemin bir arada bulunması ve test işlemlerinin desteklenen güncel sürümler üzerinde gerçekleştirilmiş olması nedeniyle önem taşımaktadır. Test ortamı, çoğu web uygulamanın üzerinde çalıştığı ortama mümkün olduğunca benzer düzeyde tutulmuş ve bu durum da uygulanan yöntemlerin mevcut sistemlerde uygulanabilirliği de sağlanmıştır. Çalışmada incelenen yöntemlerin benzeri donanımsal ve yazılımsal özelliklere sahip uygulamalarda uygulanması durumunda elde edilen performans artışlarına yakın düzeyde sonuç alınması beklenmektedir.

Bu çalışmada, Linux tabanlı işletim sistemi üzerinde ve Apache web sunucusu bünyesinde çalışan, PHP yazılım dili ve MySQL veri tabanı ile geliştirilen web uygulamalarının performansını etkileyen birçok faktör incelenmiştir. İnceleme ve test işlemleri, iki farklı yapıda çalışan uygulamalarla gerçekleştirilmiştir. Bu uygulamalar; PHP kodları ile oluşturulmuş sayfa içerisindeki kodların çalışma sürelerinin ölçümü ile sayfalara belirlenebilir sayıda anlık ve toplam istek gönderilerek yanıt sürelerinin ölçümü esasıyla çalışmaktadır. Kodların çalışma süresi ve sayfanın yanıt süresinin yanı sıra uygulamanın sistem kaynaklarını tüketim miktarı da uygulama performansının değerlendirilmesi açısından önemli bir ölçüt olmuştur.

Bu tez çalışmasında performans test işlemlerinde yük testi amacıyla kullanılan, JMeter ve ab uygulamalarının yanıt süreleri arasında bazı testlerde farklılıklar gözlemlenmiştir. JMeter, arka planda daha fazla veri kaydı ve mantıksal işlemler yaptığı için bir isteğin yanıt süresi ab'ye göre daha fazla olmaktadır. ab ise terminalden çalışması ve JMeter kadar fonksiyonel özelliğe sahip olmaması nedeniyle isteklerin yanıt sürelerini daha hızlı analiz edebilmektedir. Bazı test senaryolarında, ab ile JMeter yanıt süreleri arasında oluşan farklılıkların temel nedeni; söz konusu test senaryosu esnasında JMeter'in çalıştığı bilgisayar üzerinde çok fazla kaynak tüketmesi ve buna bağlı olarak gönderilen isteklerin yanıtlarının geç analiz edilmesidir. Bu gibi senaryoların yaşandığı durumlarda, ab test sonuçlarının kullanılmasının daha sağlıklı olacağı düşünülmektedir.

Uygulama performansını etkileyen faktörlerden ilki olarak araştırılan, geliştiriciler tarafından sıkça yapılabilen kod hataları incelemelerinde, bazı durumlar için üç kata varan performans sağlayıcı çözümler değerlendirilmiştir. Padilla, A. ve Hawkins, T.'in (2010) require ve require_once fonksiyonlarının performansı için uyguladıkları test işleminde, require fonksiyonunun yüzde 5 daha performanslı çalıştığı belirtilmişken; bu çalışmada yapılan test işlemlerinde ise require_once fonksiyonu yüzde 17 daha performanslı çalışmıştır. Bu durumun olası nedeninin, uygulamanın çalıştığı sunucunun işletim sistemi, web sunucusu veya PHP sürüm farklılıklarının olabileceği düşünülmektedir.

Önbellekleme mekanizmaları kullanımı uygulamanın yanıt süresini yaklaşık olarak üç kata varan düzeyde artırırken, CPU kullanımını da üçte iki oranında azaltmıştır. Çok sık olarak kullanılan MySQL sorgularının üzerinde yapılan testlerde, bazı durumlarda altmış kata varan yanıt süresi performans artışları gözlemlenmiştir. İşlem kodu önbellekleme mekanizmalarından olan OPcache'in performansı konusunda, AppDynamics firmasının yaptığı testlerde; yanıt süresinde yüzde 14 azalma sağlanmış ve bu değer kullanım yerine göre yüzde 74 seviyelerine kadar gelebildiği belirtilmiştir. Bu çalışmada, bu konuda yapılan testlerde ise yanıt süresinde yüzde 56 ile yüzde 72 aralığında değişen azalmalar sağlanmıştır. Bir diğer işlem kodu önbellekleme mekanizması olan XCache konusunda, Padilla, A. ve Hawkins, T.'in (2010) yaptıkları testte, yanıt süresinde yüzde 4 oranında düşüş sağlarken; bu çalışmada yapılan testlerde

yanıt süresindeki düşüş; yüzde 66 seviyelerine ulaşmıştır. Memory caching mekanizmalarından biri olan Memcached konusunda Abu Bakar ve diğerleri yaptıkları testte yanıt süresinde yüzde 28 oranında düşüş sağlarken, bu çalışmada yapılan testlerde yanıt süresinde yüzde 43 ile yüzde 74 aralığında düşüş sağlanmıştır.

MySQL'e alternatif olabilecek durumlarda kullanılan NoSQL veri tabanı uygulamaları da bu çalışmada değerlendirilmiştir. Anahtar-değer tipli NoSQL veri tabanlarından biri olan Redis; yapılan test işlemlerinde; yanıt süresinde yüzde 75 ile yüzde 87 aralığında, CPU tüketiminde yüzde 90, bellek tüketiminde ise yüzde 95 oranında düşüşler sağlanarak oldukça başarılı sonuçlar sağlamıştır. Doküman tabanlı NoSQL veri tabanı sistemi olan MongoDB ise, büyük metin tipli verilerin kullanıldığı INSERT ve UPDATE işlemlerinde MySQL'e göre yüzde 92 ile yüzde 99'a varan performans artışı sağlarken, SELECT işlemlerinde ise aynı performansı sağlayamadığı gözlemlenmiştir. Özellikle, log tutma gibi INSERT ve UPDATE işlemlerinin çok sık olduğu durumlarda MongoDB kullanımının, yapılan testlere göre ciddi oranda performans artışı sağlaması beklenmektedir.

MySQL sunucusunun veri boyutu arttıkça arama işlemlerindeki performans sorununu gidermek adına, arama işlemlerinin çok sık yapıldığı uygulamalarda performans artışı sağlayabilen, üçüncü parti arama motoru uygulamaları olan Sphinx ve Elasticsearch incelenmiştir. Sphinx ile yapılan test işlemlerinde; MySQL'e göre yanıt süresinde yaklaşık yüzde 81, CPU tüketim miktarında ise yüzde 96 oranında azalma gözlenmiştir. Elasticsearch ile yapılan işlemlerde ise, MySQL'e göre yanıt süresinde yaklaşık yüzde 54, CPU tüketim miktarında da yüzde 61 oranında azalma tespit edilmiştir.

Sunucu tabanlı optimizasyon incelemelerinde, Apache web sunucusuna alternatif olarak en çok kullanılan web sunucu yazılımlarından olan Nginx test edilmiş ve Apache web sunucusundan yüzde 1,15 daha kısa yanıt süresine sahip olduğu gözlenmiştir. PHP işleyiciler konusunda yapılan incelemelerde, Apache'nin varsayılan olarak kullandığı DSO uygulamasının; PHP-FPM'e göre yaklaşık 3,5 kat daha yüksek yanıt süresine, yüzde 26 daha fazla CPU ve yüzde 28 daha az bellek tüketimine sahip olduğu fark edilmiştir. Sunucu tarafında yapılan optimizasyon çalışmalarından biri de PHP'nin

varsayılan yorumlayıcısı Zend Engine ile Facebook mühendisleri tarafından geliştirilen HHVM üzerinde gerçekleştirilmiştir. PHP-FPM işleyicisi ve Zend Engine 2.6.0 yorumlayıcısından oluşan sistem; FastCGI ve HHVM ile güçlendirilmiş sisteme göre yaklaşık 100 kat daha uzun yanıt süresine, 243 kat daha fazla CPU ve 14 kat daha fazla bellek kullanımına sahip olmuştur. Sunucu tabanlı optimizasyon konusunda yapılan son çalışmada, popüler web hızlandırıcılardan biri olan Varnish incelenmiştir. Graziano, P. (2013), Varnish'in performansa etkisini ölçmek amacıyla yaptığı testlerde yanıt süresinde yüzde 89 oranında azalma sağlamıştı. Bu çalışmada yapılan testlerde ise Varnish'in kullanılmasıyla birlikte, sayfaya gönderilen yanıt süresi yaklaşık olarak yüzde 92 ve CPU kullanım miktarı ise yüzde 93 oranında azalmıştır. Elde edilen bu sonuçlar; sunucu tarafında yapılan birkaç değişikliğin, uygulama ve sunucu performansına ne derece büyük etkiler sağladığını görmek açısından oldukça iyi bir örnek olmuştur.

Tez çalışmasına başlarken; uygulanacak yöntemlerin uygulamalarda performansa fark edilebilir düzeyde etki etmesi beklenmiştir ve bu amaçlara tez çalışması sonrasında ulaşılmıştır. Sonuçların güvenilirliğinin sağlanması için test metotlarının, uygulama performansını ölçmek için kullanılan en popüler ve sonuçların güvenilirliği açısından en uygun metotlar arasında olmasına önem gösterilmiştir. Test işlemleri yapılan uygulamanın, işlemlerin yapıldığı bilgisayar üzerinde oluşturulan sanal sunucu üzerinde çalışması, test sonuçlarının internet bağlantı hızından bağımsız gerçekleştirilmesini sağlamıştır. Çalışmada yer alan çoğu test sonucu, birkaç kere gerçekleştirilen test işlemlerinin ortalaması alınarak elde edilmiştir. Tüm bu durumlar göz önünde bulundurulduğunda, araştırma bulgularının doğruluğunun sağlandığı söylenebilir.

Çalışmada incelenen işlemlerin, her uygulamada farklı kullanım alanlarıyla, bir arada kullanılmaları da mümkündür. Bu şekilde uygulama içerisindeki farklı durumlar ve yapılarda birden fazla iyileştirme yapılarak, performans konusunda büyük oranda artışlar sağlanabilmektedir. Bu çalışmada incelenen durumların, uygulama performansına yanıt süresi ve kaynak tüketimi bakımından yaptığı etki Tablo 5.1'de ayrıntılı olarak listelenmektedir.

Tablo 5.1: Tez çalışmasında incelenen durumların, PHP ve MySQL tabanlı uygulamalarda performansa etkisi

Uygulanan işlem	Performans etkisi
Çok sık kullanılan PHP operatör ve fonksiyonlarının yerine alternatif operatör ve fonksiyonların kullanımı	% 3 ile % 97 aralığında değişen yanıt süresi düşüşü
PHP kodlarının her istekte yeniden yorumlanmamasını sağlayan işlem kodu önbellekleme mekanizmaları kullanımı	% 50 ile % 72 aralığında değişen yanıt süresi düşüşü
Bellekte saklanan verileri, daha hızlı erişilen önbelleğe aktaran memory caching mekanizması (Memcached) kullanımı	% 43 ile % 74 aralığında değişen yanıt süresi düşüşü
Bazı MySQL sorgularının optimize edilmesi (LIMIT kullanımı, LEFT JOIN yerine INNER JOIN kullanımı)	%33 ile % 99 aralığında değişen yanıt süresi düşüşü
Çok sık güncellenmeyen verinin çekildiği sayfalarda; MySQL yerine anahtar-değer tipli NoSQL sistemi olan Redis kullanımı	%75 ile % 87 aralığında değişen yanıt süresi düşüşü
Olay kaydı, günlük tutma gibi işlemler için MySQL yerine doküman tabanlı NoSQL sistemi olan MongoDB kullanımı	UPDATE ve INSERT işlemlerinde % 99'a varan yanıt süresi düşüşü
Arama işlemlerinin çok sık yapıldığı sistemlerde; arama işlemlerinin MySQL yerine arama motoru sunucusu ile yapılması	% 55 ile % 96 aralığında değişen yanıt süresi, % 60 ile % 96 aralığında değişen CPU tüketim miktarı düşüşü
Apache web sunucusuna alternatif olarak Nginx kullanımı	% 1 yanıt süresi, % 40 bellek tüketim miktarı düşüşü
Apache'nin varsayılan PHP işleyicisi DSO yerine PHP-FPM kullanımı	% 67 yanıt süresi, % 18 CPU tüketim miktarı düşüşü
PHP 5.6'nın varsayılan yorumlayıcısı Zend Engine v2.6.0 yerine HHVM v3.12 kullanımı	% 99 yanıt süresi, % 99 CPU tüketim miktarı, % 93 bellek tüketim miktarı düşüşü
Web sunucu hızlandırıcısı olarak görev yapan Varnish Cache kullanımı	% 91 ile % 98 aralığında değişen yanıt süresi, % 93 CPU tüketim miktarı düşüşü

6. SONUÇ

PHP ve MySQL tabanlı uygulamaların performansına etki eden birçok faktör bulunmaktadır. Uygulamanın üzerinde çalıştığı sunucunun donanımında yapılacak iyileştirmeler belirli bir noktadan sonra ciddi maliyetlere neden olacağı için, öncelikli olarak söz konusu uygulamada darboğaz oluşturan durumların tespiti ve ilgili çözümlerin uygulanması düşünülmelidir. Bu tez çalışmasında da uygulamanın çalıştığı sunucunun donanımsal iyileştirmeleri dışında, yapılabilecek diğer iyileştirmeler araştırılmıştır. Bu konuda ise ağırlıklı olarak; veri tabanı ile önbellekleme mekanizmaları incelenmiş olup, uygulamanın performansında yüksek oranlarda artışlar gözlenmiştir. Bunların yanı sıra; yapılan incelemeler sonrasında web hızlandırıcı uygulanması, web sunucu, PHP işleyici ve PHP yorumlayıcısı değişikliği de uygulama performansını ciddi düzeylerde artıran etkenler olduğu tespit edilmiştir.

Çalışmada gerçekleştirilen testlerde; uygulama kaynak kodları analizi için XDebug Profiler, MySQL sorgu analizi için mysqlslap uygulaması kullanılırken, uygulamanın yanıt süresi ölçümünde ise Apache JMeter ve ab uygulamaları kullanılmıştır. Analiz ve test işlemlerinde kullanılan bu uygulamalar; sonuç güvenilirliği açısından en popüler araçlar arasındadır. İnternet bağlantısından bağımsız olarak yerel sanal sunucu üzerinde gerçekleştirilen testler de, elde edilen performans sonuçlarının doğruluğunu sağlamaktadır.

Bu araştırmada; uygulamalarda performans konusunda etkili olabilecek tek bir etken değil, birbirinden bağımsız farklı araçların kullanıldığı birçok etken detaylı olarak incelenmiştir. Yapılan araştırma sonuçlarına göre; günlük tutma gibi işlemlerde MongoDB, çok sık güncellenmeyen verilerin çekildiği sayfalarda Redis veya Memcached, site içi arama işlemlerinin yoğun olarak yapıldığı sistemlerde Elasticsearch veya Sphinx kullanımının; sadece PHP tabanlı değil; söz konusu araçların birlikte çalışabildiği diğer yazılım dilleriyle hazırlanmış uygulamalarda da ciddi oranlarda performans artışı sağlayacağı düşünülmektedir.

KAYNAKÇA

Kitaplar

Chodorow, K. & Dirolf, M. (2010). *MongoDB: The Definitive Guide*. 1. Baskı. California: O'Reilly Media, Inc., ss 3-4.

Curioso, A., Bradford, R. & Galbraith, P. (2010). *Expert PHP and MySQL*. Indiana: Wiley Publishing, Inc., s. 142.

Harrison, G. & Feuerstein, S. (2006). *MySQL Stored Procedure Programming*. 1. Baskı. California: O'Reilly Media, Inc., s. 5.

Kuč, R. & Rogoziński, M. (2013). *ElasticSearch Server*. 1. Baskı. Birmingham: Packt Publishing, s. 9.

Macedo, T. & Oliveira, F. (2011). *Redis Cookbook*. 1. Baskı. California: O'Reilly Media, Inc.

Padilla, A. & Hawkins, T. (2010). *Pro PHP application performance: tuning PHP web projects for maximum performance*. New York: Springer Science+Business Media, s. 56, 97.

Paro, A. (2015). *ElasticSearch Cookbook*. 2. Baskı. Birmingham: Packt Publishing, s. 11.

Schwartz, B., Zaitsev, P. & Tkachenko, V. (2012). *High Performance MySQL*. 3. Baskı. California: O'Reilly Media, Inc., s. 233.

Valade, J. (2010). *PHP and MySQL for dummies*. 4. Baskı. New Jersey: Wiley Publishing, Inc.

Diğer Yayınlar

- Abramova, V. & Bernardino, J. 2013. NoSQL databases: MongoDB vs cassandra. *C3S2E '13 Proceedings of the International C* Conference on Computer Science and Software Engineering*. ACM veri tabanı [erişim tarihi 05.02.2016].
- Abu Bakar, K., Shahrill, M. H. M. & M. Ahmed. 2010. Performance Evaluation of a Clustered Memcache. *Information and Communication Technology for the Muslim World International Conference*. IEEE veri tabanı [erişim tarihi 18.02.2016].
- Adams, K., Evans, J., Maher, B., Ottoni, G., Paroski, A., Simmers, B., Smith, E. & Yamauchi, O. 2014. The hiphop virtual machine. *OOPSLA '14 Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*. ACM veri tabanı [erişim tarihi 07.03.2016].
- Bai, J. 2010. Feasibility analysis of big log data real time search based on Hbase and ElasticSearch. *2013 Ninth International Conference on Natural Computation (ICNC)*. IEEE veri tabanı [erişim tarihi 11.02.2016].
- Bolton, R. 2014, Why Every PHP Application Should Use an OpCache [online], <https://blog.appdynamics.com/php/why-every-php-application-should-use-an-opcache/> [erişim tarihi 23.12.2015].
- Elasticsearch, Community Contributed Clients, <https://www.elastic.co/guide/en/elasticsearch/client/community/current/index.html> [erişim tarihi 10.02.2016].
- Elasticsearch, Use Cases, <https://www.elastic.co/use-cases> [erişim tarihi 10.02.2016].
- Graziano, P. 2013. Speed up your web site with varnish. *Linux Journal*. **2013** (227). ACM veri tabanı [erişim 10.03.2016].
- HHVM, <http://hhvm.com/> [erişim tarihi 27.02.2016].
- Kroth, B., 2014, CS764 Project Report: Adventures in Moodle Performance Analysis [online], University of Wisconsin-Madison, http://pages.cs.wisc.edu/~bpkroth/cs764/bpkroth_cs764_project_report.pdf [erişim tarihi 19.02.2016].
- Lerner, R. M. 2010. At the forge: Redis. *Linux Journal*. **2010** (197). ACM veri tabanı [erişim 02.02.2016].
- Memcached, About Memcached, <http://memcached.org/about> [erişim tarihi 20.01.2016]

MongoDB, MongoDB Licensing, <https://www.mongodb.org/licensing> [erişim tarihi 04.02.2016].

MongoDB, Who's using MongoDB?,
<https://www.mongodb.org/community/deployments?group=innovation-awards>
[erişim tarihi 04.02.2016].

MySQL, MySQL 5.7 Release Notes, <https://dev.mysql.com/doc/relnotes/mysql/5.7/en/>
[erişim tarihi 25.12.2015].

Netcraft, February 2016 Web Server Survey,
<http://news.netcraft.com/archives/2016/02/22/february-2016-web-server-survey.html> [erişim tarihi 17.02.2016].

Nginx, Tested OS and platforms, http://nginx.org/en/#tested_os_and_platforms [erişim tarihi 22.02.2016].

Nginx, <https://www.nginx.com/customers> [erişim tarihi 22.02.2016].

Nginx, Basic HTTP server features, <http://nginx.org/en/> [erişim tarihi 22.02.2016].

NoSQL Databases, <http://nosql-database.org/> [erişim tarihi 28.01.2016].

Parker, Z., Poe, S. & Vrbsky S. V. 2013. Comparing NoSQL MongoDB to an SQL DB. *ACMSE '13 Proceedings of the 51st ACM Southeast Conference*. ACM veri tabanı [erişim tarihi 05.02.2016].

PHP, News Archive – 2015, <http://php.net/archive/2015.php> [erişim tarihi 16.12.2015].

PHP, History of PHP, <http://php.net/manual/tr/history.php.php> [erişim tarihi 22.12.2015].

PHP, OPcache, <http://php.net/manual/tr/book.opcache.php> [erişim tarihi 22.12.2015].

PHP, Usage Stats for January 2013, <http://php.net/usage.php> [erişim tarihi 23.12.2015].

Redis, Introduction to Redis, <http://redis.io/topics/introduction> [erişim tarihi 29.01.2016].

Redis, FAQ, <http://redis.io/topics/faq> [erişim tarihi 29.01.2016].

Redis, How fast is Redis?, <http://redis.io/topics/benchmarks> [erişim tarihi 29.01.2016].

solid IT, Method of calculating the scores of the DB-Engines Ranking, http://db-engines.com/en/ranking_definition [erişim tarihi 16.01.2016].

Sphinx, Sphinx features, <http://sphinxsearch.com/docs/latest/features.html> [erişim tarihi 08.02.2016].

Suzumura, T., Trent, S., Tatsubori, M., Tozawa, A. & Onodera T. 2008. Performance comparison of PHP and JSP as server-side scripting languages. *Middleware '08 Proceedings of the 9th ACM/IFIP/USENIX International Conference*. ACM veri tabanı [erişim tarihi 06.03.2016].

TIOBE, TIOBE Programming Community Index Definition, http://www.tiobe.com/tiobe_index?page=programminglanguages_definition [erişim tarihi 14.04.2016].

XCache, Introduction to XCache, 2013, <https://xcache.lighttpd.net/wiki/Introduction> [erişim tarihi 26.12.2015].



ÖZGEÇMİŞ

Adı Soyadı : Vural ÖKCÜ

Sürekli Adresi : Gültepe Mah. Erdoğan Cad. No:80/6 Küçükçekmece / İstanbul

Doğum Yeri ve Yılı : İstanbul, 1988

Yabancı Dili : İngilizce

İlk Öğretim : Özel Avrupa İlköğretim Okulu, 2002

Orta Öğretim : Bağcılar Lisesi, 2005

Lisans : Ege Üniversitesi, 2010

Yüksek Lisans : Bahçeşehir Üniversitesi

Enstitü Adı : Fen Bilimleri Enstitüsü

Program Adı : Bilgi Teknolojileri

Çalışma Hayatı :

Aral Oyun Konsol ve Aksesuar A.Ş. / 11.2014 - ...

İkon Menkul Değerler A.Ş. / 01.2014 – 11.2014

Duran Doğan Basım ve Ambalaj A.Ş. / 07.2011 – 04.2013