

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**VERİ TABANI SANAL SUNUCU MAKİNASI ÇOKLU KAYNAKLARININ
GERİ BESLEME İLE KONTROLÜ**

DOKTORA TEZİ

Özgür ARMAĞAN

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

HAZİRAN 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**VERİ TABANI SANAL SUNUCU MAKİNASI ÇOKLU KAYNAKLARININ
GERİ BESLEME İLE KONTROLÜ**

DOKTORA TEZİ

**Özgür ARMAĞAN
(504042103)**

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Leyla GÖREN SÜMER

HAZİRAN 2014

İTÜ, Fen Bilimleri Enstitüsü'nün 504042103 numaralı Doktora Öğrencisi **Özgür ARMAĞAN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “**VERİ TABANI SANAL SUNUCU MAKİNASI ÇOKLU KAYNAKLARININ GERİ BESLEME İLE KONTROLÜ**” başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Prof. Dr. Leyla Gören SÜMER**
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Prof. Dr. Mehmet Turan SÖYLEMEZ**
İstanbul Teknik Üniversitesi

Doç. Dr. İbrahim B. KÜÇÜKDEMİRAL
Yıldız Teknik Üniversitesi

Prof. Dr. Bülent ÖRENCİK
Beykent Üniversitesi

Yrd. Doç. Dr. Gülay Günel
İstanbul Teknik Üniversitesi

Teslim Tarihi : 25 Nisan 2014
Savunma Tarihi : 12 Haziran 2014

Eşime ve çocuğuma,

ÖNSÖZ

Sayın Prof. Dr. Leyla Gören Sümer Hanım’a doktora tez çalışmaları süresince göstermiş olduğu destek ve vermiş olduğu rehberlik için teşekkür etmek isterim.

Haziran 2014

Özgür Armağan
(Kontrol ve Bilgisayar Mühendisi)

İÇİNDEKİLER

	<u>Sayfa</u>
ÖNSÖZ	vii
İÇİNDEKİLER	ix
KISALTMALAR	xiii
ÇİZELGE LİSTESİ	xv
ŞEKİL LİSTESİ	xvii
SEMBOL LİSTESİ	xxiii
ÖZET	xxv
SUMMARY	xxvii
1. GİRİŞ	1
2. LİTERATÜR ÖZETİ VE ÖN TASARIM ÇALIŞMASI	9
2.1 Literatür Araştırması	10
2.1.1 IBM lotus notes sunucu.....	10
2.1.2 Kuyruk sistemleri (Queueing Systems)	11
2.1.3 Apache HTTP sunucu (Server)	13
2.1.4 Yönlendirici tampon alanı taşmaları için rastgele erken uyarı sistemi (Random early detection of router overloads)	16
2.1.5 İş yükü dengeleme (Load balancing)	18
2.1.6 Duraksız akış mimarisi (Streaming media).....	19
2.1.7 Hizmet tipine göre ön bellekte farklı boyutta tampon alanlar yaratma (Caching with differentiated service)	21
2.2 Literatürdeki Diğer Araştırmalar	23
2.3 Kontrol Teorisinin Bilgisayar Sistemlerine Uygulanmasında Karşılaşılan Problemler	25
2.4 Çevrimiçi Ödeme Sisteminde Tıkanıklığı Fark Eden Geri Beslemeli Bir Kontrol ve Uyarıcı Sistem	26
2.4.1 Ödeme sistemi istemci sunucu haberleşmesi	26
2.4.2 Çevrimiçi ödeme sisteminde bir kontrol probleminin tanımlanması.....	27
2.4.3 Çevrimiçi ödeme sistemi için geri beslemeli bir sistem ve aç-kapa kontrolör tasarımı	28
2.4.4 Çevrimiçi sisteme ilişkin sonuçlar	31
3. SİSTEM TANIMA, MODELLEME VE İNTERNET TRAFİK VERİSİNİN MODELLENMESİ ÖN ÇALIŞMASI	33
3.1 Sistem Tanıma ve Modelleme Yöntemlerinin İnternet Veri Trafikinin Modellenmesinde Kullanılması.....	34
3.1.1 İnternet trafik verisinin MATLAB ortamına aktarılması ve verinin gösterilmesi	35
3.1.2 İnternet trafik verisi modelleme öncesi işlem adımları.....	36
3.1.3 Model yapısının seçimi	38
3.1.4 Sonuçlar	43

4. SANAL VERİTABANI SUNUCUSU SİSTEM ALT YAPISI VE MODELLENMESİ	45
4.1 Sanal Veri Tabanı Sunucusu Sistem Alt Yapısı	46
4.1.1 Algılayıcılar.....	51
4.1.2 Sürücüler	52
4.1.3 Kontrolör	53
4.1.4 Veri tabanı sunucu iş yükü aracı - SQLStress	53
4.1.5 Test ortamı sistem girişleri ara yüzü	56
4.2 Sistem Tanıma	57
4.2.1 Giriş-çıkış değişkenleri ve sistem tanıma deneyi için giriş işareti seçimi	57
4.2.2 Sistem tanıma deneyleri, model yapısı ve model yapılarının karşılaştırılması	58
4.3 Sistem Modeli.....	64
4.3.1 Çoklu-SISO modelin parametreleri.....	66
4.3.2 CPU kullanımı sistem çıkışına ilişkin SISO sistemin transfer fonksiyonları, basamak girişi ve sistem kararlılığı.....	67
4.3.3 Bellek kullanımı sistem çıkışına ilişkin SISO sistemin transfer fonksiyonları, basamak girişi ve sistem kararlılığı.....	69
4.3.4 MIMO modelinin parametreleri	71
4.3.5 MIMO modelin transfer fonksiyonu gösterilimi	71
4.4 Veri Tabanı Sunucu Sistem Modeline İlişkin Elde Edilen Sonuçlar	73
5. KONTROLÖR TASARIMI: PI KONTROLÖR	75
5.1 CPU Kullanımı sistemine ilişkin SISO model temelli Kontrolör Tasarımı	76
5.2 Bellek Kullanım sistemine ilişkin SISO model temelli Kontrolör Tasarımı....	78
5.3 Test Ortamının Gerçeklemesi, PI Kontrolör İle Yönetilen Sistem Deney Sonuçları.....	79
5.4 Win7 ve WinXP Veri tabanı Sanal Sunucuları Paralel Çalışma Deneyleri	88
6. KONTROLÖR TASARIMI: INTEGRAL ETKİLİ LQG SERVO KONTROLÖR	97
6.1 Kalman Filtresi Tasarımı	98
6.1.1 CPU tahsis – CPU kullanım SISO sistemine ilişkin Kalman filtresi durum parametreleri ve Kalman kazancı	99
6.1.2 Bellek tahsis - bellek kullanım SISO sistemine ilişkin Kalman filtresi durum parametreleri ve Kalman kazancı	100
6.2 İntegral Özellikli LQG Kazancı	101
6.2.1 LQG kazancını oluşturmak üzere CPU tahsis-CPU kullanım SISO sistemi için simülatörde gerçekleştirilen deneyler ve sonuçları	102
6.2.2 LQG kazancını oluşturmak üzere bellek tahsis-bellek kullanım SISO sistemi için simülatörde gerçekleştirilen deneyler ve sonuçları.....	105
6.3 İntegral Etkili LQG Servo Kontrolörün Oluşturulması.....	108
6.3.1 LQG servo kontrolörün CPU için oluşturulması.....	109
6.3.2 CPU tahsis-CPU kullanım SISO sistemine ilişkin simülatör ve emülatörden elde edilen deney sonuçları	111
6.3.3 LQG servo kontrolörün bellek için oluşturulması.....	116
6.3.4 Bellek tahsis-bellek kullanım SISO sistemine ilişkin simülatör ve emülatörden elde edilen deney sonuçları	118
6.4 Win7 ve WinXP Veri tabanı Sanal Sunucuları Paralel Çalışma Deneyleri ...	120
6.5 Deney Sonuçlarının Değerlendirilmesi	128
7. SONUÇLAR	131
KAYNAKLAR.....	133

EKLER.....	141
ÖZGEÇMİŞ.....	189

KISALTMALAR

API	: Application Programming Interface
APP	: Application
AR	: Auto Regressive
ARX	: Auto Regressive External
ATM	: Asynchronous Transfer Mode
BFL	: Buffer Fill Level
CPU	: Central Processing Unit
DBVM	: Database Virtual Machine
GB	: Gigabyte
GHz	: Gigahertz
HTTP	: Hypertext Transfer Protocol
I/O	: Input/Output
IIS	: Internet Information Services
IP	: Internet Protocol Address
IT	: InformationTechnology
Kbps	: Kilobit per sceond
LAN	: Local Area Network
LQ	: Linear-Quadratic
LQG	: Linear-Quadratic Gaussian
LQI	: Linear-Quadratic Integral
LQR	: Linear-Quadratic Regulator
MB	: Megabyte
Mbps	: Megabit per second
MHz	: Megahertz
MIMO	: Multi Input Multi Output
NLARX	: NonlinearAutoRegressive External
OE	: Output Error
OS	: Operating System
PI	: Proportional Integral
QoS	: Quality of Service
RED	: Random Early Detection
RIS	: TheNumber of RPCs In The Server.
RLS	: Recursive Least-Squares
RPC	: Remote Procedure Call
SISO	: Single Input Single Output
SIT	: System Identification Toolbox

SP : Stored Procedure
TCP : Transmission Control Protocol
VM : Virtual Machine
XML : Extensible Markup Language

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Veri tabanında tutulan log tablosu yapısı.....	29
Çizelge 2.2 : Kontrolörün log tablosundaki bilgileri kullanarak dakika bazında ürettiği istatistiksel bilgi.	30
Çizelge 4.1 : MIMO sistem ve çoklu SISO sistemler için ARX ve diğer sistem tanıma teknikleri ile elde edilen model uygunluğu çizelgede gösterilmektedir. <i>R2</i> model uyumunun başarımını göstermektedir.	64
Çizelge 6.1 : CPU tahsis-CPU kullanım SISO sistemi için Kalman durum parametreleri ve Kalman kazançları.....	100
Çizelge 6.2 : Bellek tahsis-bellek kullanım SISO sistemi için tasarım parametreleri ve Kalman kazançları.	101
Çizelge 6.3 : CPU tahsis-CPU kullanım SISO sistemine ilişkin LQ en uygun geri besleme kazancı.....	103
Çizelge 6.4 : CPU SISO sistemi Kalman filtresi durum uzayı parametreleri.....	105
Çizelge 6.5 : Bellek tahsis-bellek kullanım SISO sistemine ilişkin LQ en uygun geri besleme kazançları ve karşı düşen öz değerler.....	106
Çizelge 6.6 : Bellek SISO sistemi Kalman filtresi durum uzayı parametreleri.	108
Çizelge 6.7 : Geri beslemeli durum ve geri beslemeli olmayan durum için hataların karesel ortalamasına göre kıyaslama.	129

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Geleneksel sunucu ile sanallaştırılmış sunucu arasındaki fark. Sanal ortamda bir sunucu birden fazla VM üzerinde bulundurulabilir.....	3
Şekil 1.2 : Sanal makina içinde işletim sistemi ve uygulama (APP) bulundurur.....	4
Şekil 1.3 : Sanal ortamdaki veri tabanı sunucu sistemi için geri besleme kontrol sistemi.....	7
Şekil 1.4 : Sanal veri tabanı sunucusu üzerindeki donanım, yazılım ve kontrol elemanları.....	7
Şekil 2.1 : IBM Lotus Domino Sunucusundaki sistem elemanları ve veri akış şeması.....	10
Şekil 2.2 : IBM Lotus Domino Server için kontrol sistemi.....	11
Şekil 2.3 : Tek sunucu kuyruk sistemi (single-server queueing system).....	12
Şekil 2.4 : Kuyruk sisteminde transducer elemanının yanıt süresi tahmininde kullanılması.....	13
Şekil 2.5 : Apache HTTP sunucu mimarisi ve oturum akışı.....	15
Şekil 2.6 : Apache HTTP sunucu hedef sistemi için düzenleyici kontrol.....	16
Şekil 2.7 : RED (Random early detection) ile çalışan yönlendirici.....	17
Şekil 2.8 : Yönlendiricide tampon doluluk seviyesi geri besleme kontrolü.....	17
Şekil 2.9 : İş yükü dengeleme sistemi mimarisi.....	18
Şekil 2.10 : İş yükü dengeleme için geri besleme kontrolü.....	19
Şekil 2.11 : Duraksız akış mimarisi için bir örnek. Bu şekilde iki kuyruk sistemi mevcuttur, örneğin CPU ve disk.....	21
Şekil 2.12 : Duraksız akış mimarisinde hizmet seviye amaç ölçütünün elde edilmesini sağlayan geri besleme kontrol sistemi.....	21
Şekil 2.13 : Farklı hizmeti destekleyen ön bellek sistemi.....	22
Şekil 2.14 : Farklı hizmeti destekleyen ön bellek sistemi için geri besleme kontrol sistemi.....	23
Şekil 2.15 : İstemci sunucu haberleşmesi.....	27
Şekil 2.16 : Çevrimiçi ödeme sistemi istemci-sunucu arasındaki haberleşme genel şeması.....	27
Şekil 2.17 : Ödeme sistemi için geri besleme kontrol sistemi.....	29
Şekil 3.1 : Verinin MATLAB çalışma ortamına aktarılması.....	35
Şekil 3.2 : Tdata ile isimlendirilen veride, çıkışı ifade eden paket sayısının zamana göre değişimi.....	36
Şekil 3.3 : Tdata ile isimlendirilen çıkış verisinin periodogram üzerinde veri spektrumu.....	36
Şekil 3.4 : Mavi renkteki grafik tdata'yı gösterir, kırmızı renkteki bilgi bozuk veriden temizlenmiş veri tdatade 'yi ifade eder.....	37
Şekil 3.5 : Kırmızı renkteki tdatade çalışma verisi, mavi renkteki tdatdv doğrulama verisidir.....	38

Şekil 3.6 : Kırmızı renkteki tdatade çalışma verisi için veri spektrumunu, mavi renkteki tdatdv doğrulama verisi için veri spektrumunu gösterir.....	38
Şekil 3.7 : AR model yapısının sisteme uygulanması için parametre girişi yapılır. Sistem mertebesi dört olarak seçilmiştir.	39
Şekil 3.8 : AR model ile elde edilen sistem çıkışı ile ölçülen gerçek çıkışın gösterimi, siyah renk gerçek çıkışı, yeşil renk modellenen sistem çıkışını ifade eder.	40
Şekil 3.9 : Modellenen sistem için kutup-sıfır dağılımı.....	40
Şekil 3.10 : Durum uzayı model yapısı seçildi. Model derecesini 1-10 arası uygun bir değer olarak seçmek üzere 1:10 seçimi yapıldı.	41
Şekil 3.11 : Bir durum değişkeni modelleme için yeterli görüldü, diğer durumlar dikkate alınmadı. Sistem derecesi 1 olarak belirlendi.	42
Şekil 3.12 : Durum uzayı model yapısı kullanılarak ve n4sid metodu ile elde edilen modelin çıkışı kırmızı ile ölçülen sistem çıkışı siyah ile gösterilmektedir.	42
Şekil 3.13 : Durum uzayı model yapısı kullanılarak n4sid metodu ile elde edilen sisteme ilişkin sıfır-kutup dağılımı.	43
Şekil 4.1 : Veri merkezi genel mimarisi.	46
Şekil 4.2 : 192.68.10.10 IP'li ESX sunucu üzerinde yaratılan sanal makinalar.	47
Şekil 4.3 : ESX sunucu yapılandırması gösterilmektedir.	47
Şekil 4.4 : ESX sunucu makinası Red Hat işletim sistemi üzerinde kurulmuş bir makinadır. Tüm sanal platform, bu sistem üzerinde yer alır.	48
Şekil 4.5 : Win7 sanal makinesine ilişkin kaynaklar gösterilmektedir.	49
Şekil 4.6 : Konak, sanal makina ve disklerin ilişkisini gösteren ağ şemaları.	49
Şekil 4.7 : Konak, sanal makina ve disklerin ilişkisini gösteren detaylı ağ şemaları.	50
Şekil 4.8 : Sanal makinalar ve bu makinaların kontrolüne ilişkin genel şema.	51
Şekil 4.9 : Veri tabanı sanal makinası için CPU kaynak tahsisinin yapıldığı yönetim ara yüzü.	53
Şekil 4.10 : Sanal veri tabanı sunucusuna bağlantı sağlamak üzere ön ekran aracılığıyla parametreler girilebilir.	54
Şekil 4.11 : Sanal veri tabanı sunucusu üzerinde iş yükü yaratan ara birimin parametre tablosu.	54
Şekil 4.12 : CPU ve bellek referanslarının belirlendiği kullanıcı ara yüzü.	56
Şekil 4.13 : Tamamlanan veri tabanı isteklerini gözlemleyebilmek için DS SQL sunucu veri tabanındaki kayıtlar sorgulanabileceği sorgulama ara yüz ekranı.	56
Şekil 4.14 : Bellek tahsis (M_a) – CPU kullanım (C_u).	59
Şekil 4.15 : CPU tahsis (C_a) – Bellek tüketimi (M_c).	59
Şekil 4.16 : C_a sistem girişi ve C_u sistem çıkışının zamana göre değişimi.	60
Şekil 4.17 : M_a sistem girişi ve M_c sistem çıkışının zamana göre değişimi.	60
Şekil 4.18 : W iş yükü ve C_u sistem çıkışının zamana göre değişimi.	61
Şekil 4.19 : W iş yükü ve M_c sistem çıkışının zamana göre değişimi.	61
Şekil 4.20 : C_a sistem girişi ve C_u sistem çıkış verileri ortalama değer çıkartıldıktan sonra tahmin ve doğrulama kısımları olmak üzere iki kısma ayrılır; yeşil kısım tahmin verisi, mavi kısım doğrulama verisidir.	61
Şekil 4.21 : M_a sistem girişi ve M_c sistem çıkış verileri ortalama değer çıkartıldıktan sonra tahmin ve doğrulama kısımları olmak üzere iki kısma ayrılır; yeşil kısım tahmin verisi, mavi kısım doğrulama verisidir.	62

Şekil 4.22 : SISO sistem simülasyonu için MATLAB SIMULINK modeli.	63
Şekil 4.23 : MIMO sistem simülasyonu için MATLAB SIMULINK modeli.....	63
Şekil 4.24 : SISO sistem C_a sistem girişi - C_u sistem çıkışı için sıfır-kutup dağılımı.	68
Şekil 4.25 : SISO sistem C_a sistem girişi - C_u sistem çıkışı için basamak cevabı. .	69
Şekil 4.26 : SISO sistem M_a sistem giriş - M_c sistem çıkış kutup dağılımı.	70
Şekil 4.27 : SISO sistem M_a giriş - M_c çıkış sisteminin basamak cevabı.	70
Şekil 5.1 : Sanal veri tabanı sunucusu çoklu SISO sistemlerinin geri besleme ile kontrolüne ilişkin simülink şeması.....	81
Şekil 5.2 : CPU SISO sistem için emülatör ortamında referans değer ve CPU tahsis gösterimi.....	82
Şekil 5.3 : CPU SISO sistem için emülatör ortamında referans değer ve CPU kullanım gösterimi.	82
Şekil 5.4 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis, CPU kullanım gösterimi.....	83
Şekil 5.5 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis, CPU kullanım ve referans ile sistem çıkışı olan CPU kullanımı arasındaki farktan kaynaklanan hata bilgilerinin gösterimi.....	83
Şekil 5.6 : CPU SISO sistem için simülatör ortamında referans değer, CPU tahsis, CPU kullanım gösterimi.....	84
Şekil 5.7 : CPU SISO sistem için simülatör ve emülatör ortamları için referans değer, CPU tahsis gösterimi.....	84
Şekil 5.8 : CPU SISO sistem için simülatör ve emülatör ortamlarından elde edilen referans değer, CPU kullanım verisi gösterimi.	85
Şekil 5.9 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis -kontrol girişi- gösterimi.	85
Şekil 5.10 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tüketim -sistem çıkışı- gösterimi.	86
Şekil 5.11 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis, bellek tüketim gösterimi.....	86
Şekil 5.12 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis, bellek tüketim ve referans ile sistem çıkışı olan bellek kullanımı arasındaki hata bilgilerinin gösterimi.	87
Şekil 5.13 : Bellek SISO sistem için simülatör ortamında referans değer, bellek tüketim gösterimi.....	87
Şekil 5.14 : Bellek SISO sistem için emülatör ve simülatör ortamında referans değer, bellek tüketim gösterimi.	88
Şekil 5.15 : WinXP sanal makinası emülatör CPU sonuçları.....	89
Şekil 5.16 : WinXP sanal makinasına etkiyen iş yükü.	89
Şekil 5.17 : Win7 sanal makinası emülatör CPU sonuçları.....	90
Şekil 5.18 : Win7 sanal makinasına etkiyen iş yükü.	90
Şekil 5.19 : WinXP sanal makinası emülatör bellek sonuçları.....	91
Şekil 5.20 : WinXP sanal makinasına etkiyen iş yükü.	91
Şekil 5.21 : Win7 sanal makinası emülatör bellek sonuçları.....	92
Şekil 5.22 : Win7 sanal makinasına etkiyen iş yükü.	92
Şekil 5.23 : WinXP sanal makinası emülatör CPU sonuçları.....	93
Şekil 5.24 : Win7 sanal makinası emülatör CPU sonuçları.....	93
Şekil 5.25 : PI kontrolör ile yönetilen WinXP makinasına ilişkin bellek sonuçları. .	94
Şekil 5.26 : WinXP sanal makinasına etkiyen iş yükü.	94

Şekil 5.27 : PI kontrolör ile yönetilen Win7 makinasına ilişkin bellek sonuçları.	95
Şekil 5.28 : Win7 sanal makinasına etkiyen iş yükü.	95
Şekil 6.1 : LQG düzenleyici.	97
Şekil 6.2 : İntegral etkili LQG kontrolör ile yönetilen geri beslemeli sistem.	98
Şekil 6.3 : Kalman Filtresi.	99
Şekil 6.4 : LQI.	102
Şekil 6.5 : Simülatör CPU tahsis kontrol işareti.	104
Şekil 6.6 : Simülatör CPU kullanım çıkış işareti.	104
Şekil 6.7 : Simülatör bellek tahsis kontrol işareti.	107
Şekil 6.8 : Simülatör bellek kullanım çıkış işareti.	107
Şekil 6.9 : İntegral etkili servo kontrolör.	108
Şekil 6.10 : Simülatör veri tabanı sunucusu CPU modeline ilişkin birim basamak girişi cevabı.	111
Şekil 6.11 : Sanal ortamdaki veri tabanı sisteminin LQG servo kontrolör ile geri besleme kontrolüne ilişkin simülasyon.	112
Şekil 6.12 : CPU SISO sistem için simülatör ortamında referans değer ve CPU tahsis bilgilerinin gösterimi.	112
Şekil 6.13 : CPU SISO sistem için simülatör ortamında referans değer ve CPU kullanım bilgilerinin gösterimi.	113
Şekil 6.14 : CPU SISO sistem için simülatör ortamında referans değer CPU tahsis ve CPU kullanım bilgilerinin gösterimi.	113
Şekil 6.15 : Simülatör sanal ortamdaki veri tabanı sunucusuna etkiyen iş yükü dağılımı.	113
Şekil 6.16 : CPU SISO sistem için emülatör ortamında referans değer ve CPU tahsis gösterimi.	114
Şekil 6.17 : CPU SISO sistem için emülatör ortamında referans değer ve CPU kullanım gösterimi.	114
Şekil 6.18 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis ve CPU kullanım gösterimi.	114
Şekil 6.19 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis, CPU kullanım ve referans ile sistem çıkışı olan CPU kullanımı arasındaki hata bilgilerinin gösterimi.	115
Şekil 6.20 : CPU SISO sistem için simülatör ve emülatör ortamları için referans değer, CPU tahsis gösterimi.	115
Şekil 6.21 : CPU SISO sistem için simülatör ve emülatör ortamları için referans değer, CPU kullanım gösterimi.	115
Şekil 6.22 : Simülatör ortamda veri tabanı sunucusu bellek modeline ilişkin birim basamak girişi cevabı.	118
Şekil 6.23 : Bellek SISO sistem için emülatör ortamında referans değer ve bellek tahsis gösterimi.	118
Şekil 6.24 : Bellek SISO sistem için emülatör ortamında referans değer ve bellek kullanım gösterimi.	119
Şekil 6.25 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis ve bellek kullanım gösterimi.	119
Şekil 6.26 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis, bellek kullanım ve referans ile sistem çıkışı olan bellek kullanımı arasındaki hata bilgilerinin gösterimi.	119
Şekil 6.27 : Bellek SISO sistem için simülatör ve emülatör ortamları için referans değer, bellek kullanım gösterimi.	120
Şekil 6.28 : Bellek SISO sistemine etkiyen iş yükü.	120

Şekil 6.29 : WinXP sanal makinası emülatör CPU sonuçları.....	121
Şekil 6.30 : Win7 sanal makinası emülatör CPU sonuçları.....	121
Şekil 6.31 : WinXP sanal makinası emülatör bellek sonuçları.....	122
Şekil 6.32 : WinXP sanal makinasına etkiyen iş yükü.	122
Şekil 6.33 : Win7 sanal makinası emülatör bellek sonuçları.....	123
Şekil 6.34 : Win7 sanal makinasına etkiyen iş yükü.	123
Şekil 6.35 : PI kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları. .	124
Şekil 6.36 : LQG+I kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları.	124
Şekil 6.37 : WinXP sanal makinası emülatör CPU sonuçları.....	125
Şekil 6.38 : WinXP sanal makinasına etkiyen iş yükü.	125
Şekil 6.39 : Win7 sanal makinası emülatör CPU sonuçları.....	126
Şekil 6.40 : Win7 sanal makinasına etkiyen iş yükü.	126
Şekil 6.41 : WinXP sanal makinası emülatör bellek sonuçları.....	127
Şekil 6.42 : WinXP sanal makinasına etkiyen iş yükü.	127
Şekil 6.43 : WinXP sanal makinası emülatör bellek sonuçları.....	128
Şekil 6.44 : Win7 sanal makinasına etkiyen iş yükü.	128
Şekil 7.1 : Veri merkezindeki tüm sunucuların CPU ve bellek kaynaklarını yöneten merkezi kontrolör.	132
Şekil A.1 : Sanallaştırma kavramını ifade eden şema.	142
Şekil A.2 : Veri merkezi kavramına ilişkin şema.	142
Şekil A.3 : Konak sistemi.	143
Şekil A.4 : Sanal makina ve bu kavramla ilgili şemalar.	144
Şekil A.5 : Küme bir ya da birden fazla konak makinasından oluşabilir ve kümede yer alan konakların kaynaklarına sahiptir.	144

SEMBOL LİSTESİ

C_a	: CPU tahsis sistem giriş değişkeni
C_u	: CPU kullanım sistem çıkış değişkeni
d	: Sistem çıkışında görülen gecikme
e	: $J(u)$ maliyet fonksiyonu optimize eden K kazancı ile kontrol edilen kapalı çevrim sistemin öz değerleri
e_{ss}	: Sürekli hal hatası
$e(k)$	: Ayrık zamanlı hata işareti
$E(z)$	: Sistem hatası – z tanım bölgesi
$G(z)$	: Hedef sistem – z tanım bölgesi
$G_c(z)$	: Kontrolör transfer fonksiyonu - z tanım bölgesinde
$J(u)$	: Kontrol işaretini parametre alan maliyet fonksiyonu
K	: Sistem kazanç katsayısı
K_I	: PI Kontrolör integral kazanç katsayısı
K_P	: PI Kontrolör oran kazanç katsayısı
L	: Kalman kazancı
M_a	: Bellek tahsis sistem girişi
M_c	: Bellek tüketim sistem çıkışı
n	: Sistem mertebesi
P	: Riccati denkleminin çözümü
Q, R, N	: LQI ağırlık katsayıları
Qn, Rn, Nn	: Kalman filtresi ağırlık katsayıları
$r(k)$	: Ayrık zamanlı giriş işareti
$R(z)$	: Giriş işareti – z tanım bölgesi
S	: $J(u)$ maliyet fonksiyonu Riccati denkleminin çözümü
t	: Zaman
Ts	: Ayrık sistem örnekleme zamanı
$T(z)$	: Kapalı çevrim transfer fonksiyonu – z tanım bölgesi
$u(k)$	: Ayrık kontrol işareti
$u(t)$	: Sürekli zamanlı kontrol işareti
$U(z)$	: Kontrol işareti – z tanım bölgesi
W	: İş yükü
x	: Sistem durum değişkeni
x_i	: İntegratörden gelen durum değişkeni
$\hat{x}$	: Tahmin edilen sistem durumu
$\hat{y}$	: Tahmin edilen sistem çıkışı
$y(k)$	: Ayrık zamanlı çıkış işareti
$y(t)$	: Sürekli zamanlı çıkış işareti

$Y(z)$: Sistem çıkışı – z tanım bölgesi

VERİ TABANI SANAL SUNUCUSU ÇOKLU KAYNAK KULLANIMLARININ GERİ BESLEME İLE KONTROLÜ

ÖZET

Sanallaştırma ve bulut bilişim son zamanlarda dikkat çeken konulardandır. Bilgisayar sistem kaynaklarının ve uygulamalarının verimliliğini ve kullanılabilirliğini sanallaştırma yoluyla arttırmak mümkündür. Sanallaştırma bir fiziksel makine üzerinde birden çok sanal makinanın çalışmasını mümkün kılar. Fiziksel makina üzerindeki her bir sanal makina fiziksel makinanın kaynaklarını kullanır. Yeni nesil veri merkezleri sanallaştırmayı kullanarak tasarlanmaktadır. Bu mimaride kaynaklar genel bir kaynak havuzunda tutulur. Sanal makineler üzerindeki uygulamalar, zaman içinde değişen ihtiyaçlara göre bu kaynakları paylaşır. Bununla birlikte günümüz sanallaştırma teknolojileri karmaşık hizmet amaçlarına ulaşmak için ihtiyaçlara göre farklı kaynaklar sağlamada yetersiz kalabilmektedir. Kaynak tahsisi ve birden fazla kaynakların sanal veri merkezindeki sanal makineler arasında kullanımının kontrolü ve yönetimi halen açık bir problemdir. Bulut bilişim ortamında az kaynak tahsisi, uygulamaların yavaş çalışmasına sebep olurken fazla kaynak tahsisi ise kaynakların ihtiyaca göre gereksiz yere kullanılmasına sebep olur.

Bu tez çalışmasında bir sanal makina üzerindeki veri tabanı sunucusunun CPU kullanımı ve bellek tüketimi, CPU ve bellek kaynaklarının dinamik olarak sanal makineye tahsisi ile kontrol edilmek ve yönetilmek istenmektedir. Sanallaştırılmış veri tabanı sunucu sisteminin CPU kullanımı ve bellek tüketimini kontrol etmek için geliştirilen altyapı başarılı olduğu takdirde veri merkezindeki tüm sanallaştırılmış makinelerin kaynaklarının yeterli miktardaki dinamik kaynak tahsisi ile kontrol edilebileceği düşünülmüştür. Böylelikle yeterli miktarda kaynak tüketimi sağlanarak veri merkezindeki tüm uygulamalar için servis kalitesi garanti edilebilecektir ve ihtiyacı olan diğer uygulamalar için daha fazla CPU ve bellek kapasitesi sağlanabilecektir. Tez çalışması kapsamında hedeflenen temel amaç, özellikle yoğun iş yükü altındaki veri tabanı sunucusu sanal makinasının CPU kullanımı ve bellek tüketimini kontrol etmektir. İlgili literatürde var olan çalışmalardan farklı olarak, CPU kullanımı ve bellek tüketimi çoklu çıkışları, çoklu sunucu parametreleri olan CPU tahsisi ve bellek tahsisi parametreleri gerçek zamanlı olarak değiştirilerek kontrol edilmeye çalışılmıştır. Sanal ortamda, zamanla değişen iş yükü altında çalışan geri besleme kontrol sistemi gerçekleştirilmiş ve doğruluğu test edilmiştir.

Tez çalışmasında veri tabanı sanal sunucusunun lineer ve zamanla değişmeyen olarak modellenemediği gösterilmiş, iki ayrı sistem modeli -MIMO ve çoklu SISO- elde edilmiştir. Modelleme yöntemi olarak sistem tanıma teknikleri ile kara-kutu yaklaşımı kullanılmıştır. Elde edilecek kontrol kuralının sisteme uygulanmasını mümkün kılan sürücü sistemi ve geri beslenmesi gereken büyüklüklerin algılanmasını sağlayan ölçme sistemi tasarlanmıştır. Kontrol çevrimi içinde CPU kullanımı ve bellek tüketimi bu algılayıcılar vasıtasıyla ölçülmüş ve istenilen CPU kullanımı ve bellek tüketimini sağlamak amacıyla çeşitli kontrolör tasarımları

yapılmıştır. Tasarlanan kontrolörlerin gerçek zamanlı testleri için VMware sanal makinalarının kullanıldığı gerçek bir veri merkezini emüle eden bir test ortamı oluşturulmuştur. Test ortamında veri tabanı sunucusu üzerinde SQL Server veri tabanı kurulmuş, zamanla değişen yoğun iş yükü etkisi yaratılmıştır. Bu koşullar altında veri tabanı sunucusunun CPU kullanımı ve bellek tüketimini istenen değerlerde tutmak üzere önerilen ve tasarlanan kontrolörler gerçek zamanlı bir geri besleme çevrimi kullanılarak test edilmiştir.

Deneyisel sonuçlar, kontrol teorisi ile oluşturulan yaklaşımın, özellikle yoğun iş yükü altındaki veri tabanı sunucusunun CPU ve bellek kaynak tüketimini istenilen seviyede tutma problemine dayanaklı bir çözüm oluşturduğunu göstermiştir.

FEEDBACK CONTROL FOR MULTI-RESOURCE USAGE OF VIRTUALISED DATABASE SERVER

SUMMARY

Virtualization and cloud computing have recently received significant attention. It's possible to improve the efficiency and availability of computer system resources and applications through virtualization. Virtualization lets us run multiple virtual machines on a single physical machine, with each virtual machine sharing the resources of that one physical computer across multiple environments. Next-generation data centers are being designed by considering virtualization. In this architecture, resources are pooled in a common shared infrastructure. Applications run on virtual machines share these resources as changing needs over time. However current virtualization technologies are inadequate in achieving complex service level objectives under varying needs for multiple resources. Resource allocation and control of multiple resource usages among virtual machines in the virtualized data center remains an open problem. Both under provision of resources that causes application slow-down and overprovision of resources that leads unnecessary resource allocation should be avoided in cloud computing environment.

Data centres are very important for application in the areas of virtualisation and cloud computing. A data centre is a facility used to house computer systems and associated components, such as network components and storage systems. Recently, data centres have become virtualised. Data centre virtualisation is a method of moving data and applications from physical servers to virtual ones. Server virtualisation uses software to make one server act as if it were a number of separate servers. In a virtualised environment, the resources are shared among the hosts and virtual machines. One of important parts of virtualisation is the virtual machine (VM). Each VM has an operating system and applications inside it and acts as an isolated software container. VMs may hold one or more applications. A VM can hold a database or database management system. Thus, a VM serves as a database server.

A Database Management System (DBMS) is a set of programs that enables the storage, modification, and extraction of information from a database. It also provides users with tools to add, delete, access, modify, and analyse data stored in one location. Databases are used in wide range of applications from smart card applications that have limited resources to enterprise applications that need more resources compared to the smart card applications. Database servers aim to process transactions in time. Database and database management system are essential in all areas of business, and therefore, they must be carefully managed. There can be changing workloads on database servers. However, transactions must be completed in a timely manner. So, QoS (quality of service) for database servers is an important consideration. CPU usage and memory consumption of database servers should be under control, and CPU and memory resources should not be over-utilised or under-utilised. We think that CPU usage and memory consumption can be kept under

control by proper allocation of CPU and memory resources in virtual database servers (DBVM).

In this thesis study, we focus on resource control in virtual database servers and aim to control CPU usage and memory consumption of virtual database servers through proper CPU and memory allocation in cloud computing environments. To control the system, a mathematical model of the system must be obtained. Because first principles methods for computing systems do not exist, methods derived from experiments and observations, rather than theory, are used to derive a mathematical model for the database system.

A black-box approach that uses the input-output relationship of the system is used to obtain the system model. The MATLAB System Identification Toolbox helped us to obtain the system model. To model a virtual database server, we first collected input and output data that reflects system dynamics from real systems. Second, mean values, trends and outliers were removed from the data, and model data were divided into estimation and validation parts. Third, different modelling structures were tried to determine the best fit, and a system model was predicted. Finally, the model was validated using validation data. In all experiments, a system sampling period of 20 seconds was chosen, which is sufficient considering the system response time to system inputs. The hypervisor calculates the average values of each system output in every sampling period to present those values as the measured system outputs.

From a system identification point of view, the ‘CPU and memory resource allocation’ might be considered the system inputs of a virtual database server, and the ‘CPU usage and memory consumption’ are taken as the system outputs. Because the workload also has an important effect on the system outputs, it has been regarded as a disturbance input in the modelling. To derive the mathematical model of the system, two input signals, ‘CPU and memory allocation’, that have sinusoidal forms and a third disturbance signal, ‘workload’, that takes random values, have been applied to the virtual database system. A MIMO system model describes the behaviour of the DBVM in terms of CPU allocation, memory allocation, CPU usage and memory consumption. The two SISO system model describes the behaviour of a DBVM where the first SISO model is defined as the ‘CPU allocation’ to the ‘CPU usage’, and the second one is defined as the ‘memory allocation’ to the ‘memory consumption’. The MIMO system and multiple SISO system models are validated and compared according to their fit ratio, which is an indicator of how well the model fitted real system dynamics.

In this thesis study, we address the problem of multiple resource usage among virtual machines in data centres via dynamic allocation of CPU and memory resources among the virtual machine. We think that, if our framework that was developed to control CPU usage and memory consumption in virtualised database servers was successfully applied, the resource usages of all of the virtual machines might be controlled through dynamic allocation of sufficient resources to the virtual machines in the data centre. Thus, QoS might be guaranteed for all applications in the data centre by maintaining sufficient resource consumption, and more CPU and memory resources can be provided for other applications. Added to the existing one, second database virtual machine was also created to show that other applications can share resources with the database virtual machine. The reason that the database server was chosen as the target system is the popularity, wide usage, and importance of database servers in data centre applications. The CPU usage and memory consumption need to

meet the desired level. To achieve this goal, a feedback control system was constructed.

To test our controllers, we built a test bed for a virtual data centre using VMware VMs. In the test bed, a virtual data centre is created on a host that has two cores containing four logical i7 CPU Q720 1.6 GHz (gigahertz) processors, 5 GB (gigabyte) memory, a 60 GB hard disk for the operating systems and a 140 GB hard disk for data. Two DBVMs reside on the host, and these virtual machines are entitled to a maximum of 4 GB memory and three logical i7 CPU Q720 1.6 GHz processors. The DBVMs share the resources of the host. One of the DBVMs runs on the Windows 7 operating system and the other DBVM runs on the Windows XP operating system. Structured Query Language (SQL) Server 2008 R2 is set up on both of the DBVMs. The SQL stress workload generator was used to create a synthetic workload on the SQL Server database to adjust the workload on the virtual database. The host and virtual machines are connected to a LAN (Local Area Network) through a 100 Mbps (megabits per second) virtual network adapter. The host and virtual machines in the data centre communicate with each other via transmission control protocol (TCP).

Sensors are used to measure the CPU usage and memory consumption system outputs in our test bed. The sensors are implemented using a C# program that calls a hypervisor application programming interface (API) to collect real-time CPU usage and memory consumption statistical data. Historical data related to the system outputs can also be accessed using these sensors. The hypervisor performance manager provides a service interface for obtaining statistical data about various aspects of the system performance, as generated and maintained by the system's performance providers. A performance provider is any managed object that generates performance metrics. The performance providers are managed entities, such as hosts, virtual machines, and computer resources. The sensors that we implemented are software scripts that obtain CPU usage and memory consumption statistical data from the hypervisor performance manager by making API calls to it.

In the test bed, the actuators that we implemented are software procedures used to allocate CPU and memory resources to the virtual machines. The limiting parameters of CPU and memory resources are used for actuation. The limit parameter is an upper limit on the memory or CPU resources the host makes available to a virtual machine. The virtual machine does not exceed this limit, even if the resources are available. This property is typically used to ensure consistent performance. The limit parameter can be adjusted by calling the hypervisor APIs in such a way that an instance of the server side object of a virtual machine can be created on the client side and, thus, allow us to change the limit properties of the server side objects. In the test bed, the CPU and memory allocations are made using hypervisor API calls that adjust the limit parameters of CPU and memory resources for the database virtual machines.

In every feedback cycle, the CPU usage and memory consumption system outputs are measured using sensors. After the measurement of the system outputs by the sensors, the controller calculates the appropriate memory and/or CPU allocation for the next cycle considering the error between each system output and related reference input. The actuators give the system the appropriate memory and CPU allocation. There are two controllers for each multiple-SISO model, while there is only one controller for the MIMO model. We focused on the multiple-SISO model

and designed the CPU and memory controllers separately for each SISO system. The controllers were tested in a simulator and used in real-time for the feedback control of the DBVM in our emulator.

A tool was used in the experiments to stress test the SQL Server database on the virtual machines. A mix of different types of statements can be defined to simulate actual workloads. The workload generator can simulate many virtual users. A transaction scheduler can sustain a specified number of transactions per second.

The inputs and system input-output data collected from the emulator (test bed) were transferred to a MATLAB workspace to simulate and analyse the system in the simulator. First, multiple SISO models of the system were used to describe the DBVM in the MATLAB Simulink. Second, both PI (proportional integral) and LQG+I (Linear-Quadratic Gaussian controller with integral action) controllers that satisfy the stability-accuracy-settling time-overshoot conditions were designed in the simulator. Third, a feedback control system was designed in the MATLAB Simulink [16] in such a way that the DBVM, controller and other control components were connected to construct a feedback loop. Finally, the simulator system was run by applying the same reference inputs, such as the CPU usage reference, memory consumption reference and varying workload, obtained from the emulator system.

In this thesis study, we demonstrated the application of control theory for resource management in virtual database servers. We presented a resource management architecture for virtual database servers in a cloud computing environment. Both MIMO and multiple SISO system models were obtained by applying different system identification techniques for the input and output data collected from the emulator. We emphasise that the relationship between the system inputs and outputs can be approximated using a linear model. The SISO models captured the system dynamics equally as well as the MIMO model. Therefore, the SISO models are preferred for the application of the feedback controls in the virtual database system. To control the system, we designed both PI and LQG+I controllers for each of the SISO systems. In addition to the controllers, actuators and sensors were also implemented to close the feedback loop. The feedback control system aimed to ensure that the resources were given to each virtual database machine according to the resource demands of that machine. The experimental results confirmed that control theoretical approach increased both the efficiency of the resource usage of the virtual database servers and the QoS in a cloud environment.

1. GİRİŞ

Son elli yılda kontrol kuramı ve teknikleri çok çeşitli alanlarda uygulanmış ve başarılı sonuçlar elde edilmiştir. Geri beslemeli kontrol kuramı ve tasarım teknikleri son yıllarda bilgisayar ve bilişim sistemlerini ilgilendiren konularda da kullanılmaya ve bu sistemlerde ortaya çıkan bazı problemlerinin çözümünde de kullanılmaya başlamıştır. Bununla birlikte, kontrol kuramının bilgisayar sistemlerinin kaynak yönetiminde kullanımına çok seyrek rastlanmaktadır. Bu tez çalışmasının amacı; bilgisayar sistemlerindeki dinamik kaynak yönetimi probleminin çözümünde kontrol teorisi ve tekniklerini uygulamak olarak özetlenebilir. Yapılan edebiyat araştırmasının sonucunda bu problemin tanımlandığı ancak dinamik bir çözümün var olmadığı ve kontrol tekniklerinin de kullanılmadığı görülmüştür.

Tez çalışması amacına paralel olarak kontrol teorisinin bilgisayar sistemlerinin yönetimi ile ilgili olarak kullanılabilirliği ya da var olan bilgisayar problemlerinin kontrol yaklaşımı ile çözümü konusunda literatür araştırması yapılmıştır. Ayrıca kontrol kuramının bilgisayar problemlerinin çözümünde kullanılıp kullanılamayacağını ya da hangi tekniklerin kullanılabileceğini göstermek üzere literatürdeki çözüm bekleyen bilgisayar sistemlerine ilişkin problem ve konular incelenmiştir.

Kontrol kuramının bilgisayar problemlerinin çözümünde kullanılıp kullanılamayacağını belirlemek, bir tasarım deneyimi edinmek ve bilgisayar sistemlerinin kontrolü konusunda bilgi ve tecrübe sahibi olmak için Web servis uygulaması olan bir çevrimiçi fatura ödeme sisteminde, işleyişte tıkanıklık olduğu zaman bu tıkanıklığı fark edip sistemi çevrimdışı çalışma yöntemine geçiren ve tıkanıklık ortadan kalktığı zaman sistemi tekrar çevrimiçi çalışma yöntemine alan aç-kapa geri besleme kontrol sistemi tasarlanmış ve gerçekleştirilmiştir. Yapılan bu çalışmanın istenilen sonucu vermesi, kontrol teorisinin bilgisayar sistemlerini ilgilendiren problemlere uygulanabilirliğini göstermesi açısından önemli bir adım olduğu düşünülmüştür ve elde edilen sonuçlar bilgisayar sistemlerinin işleyişi ve yönetimi konusunda bilgi vermiştir.

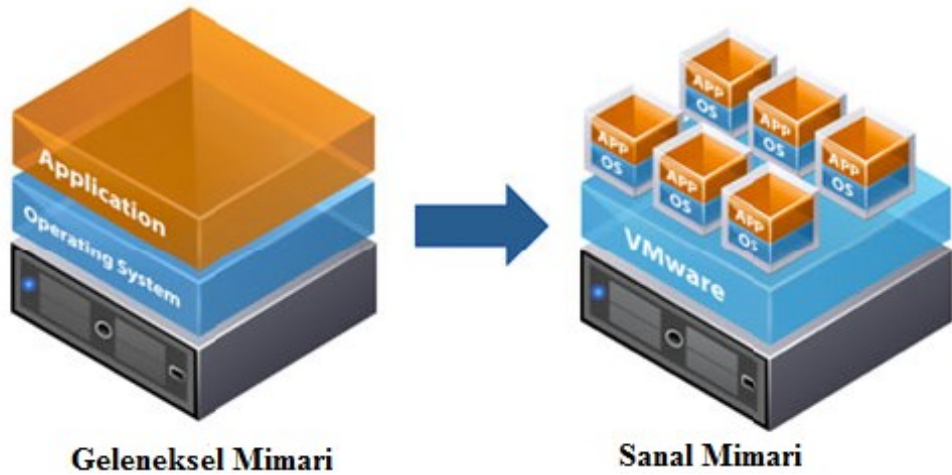
Ön çalışma olarak sunucular, kuyruk yapıları, bellek, ağ gibi bilgisayar sistemi ya da karmaşık bir bilgi işlem sisteminin alt sistemlerinin matematiksel modelinin türetilmesi için sistem tanıma ve modellemedeki izlenen iş adımlarını incelemek ve uygulanan yöntemleri görmek amacıyla, sistem kontrol literatüründe var olan sistem tanıma ve modelleme yöntemleri üzerine araştırma yapılmıştır. Kontrol tekniklerinin bu problemlere uygulanabilmesi için bilgi işlem sistemlerinin davranışını analiz etmek ve matematik modelinin elde edilmesi gerekir. Bu amaçla, MATLAB'ın sistem tanıma konusundaki gelişmiş ara yüzünden [1] ve işlem kabiliyetinden yararlanmak suretiyle farklı tanıma yöntemleri Berkeley Laboratuvarı ile internet arasındaki veri trafiğinin [2,3] matematik modelinin elde edilmesi problemine uygulanmış ve elde edilen sonuçlar tartışılmıştır.

Bilgisayar sistemleri ile ilgili olarak veri merkezi (data center) yönetimi teknolojik açıdan önemli bir konudur. Bu konuda çalışmalar yapılmakta ve en son teknolojiler kullanılmaktadır. Geline son durumda bulut mimarisi (cloud computing) ve sanallaştırma (virtuliazation) kavramları yaygınlaşmıştır [4]. Bilgi teknolojileri (IT) firmaları sistemlerini bulut mimarisi alt yapısına uygun hale getirmektedirler [5]. Sanallaştırma kavramının öne çıkmasına rağmen, sanallaştırma ile birlikte ortaya çıkan kaynak yönetimi gibi yeni problemlere henüz bir çözüm getirilememiştir. Bu problemlerin birçoğuna geri beslemeli kontrol kuramı kullanılarak çözümler bulunabilir veya problemlerin çözümünde iyileştirmeler sağlanabilir.

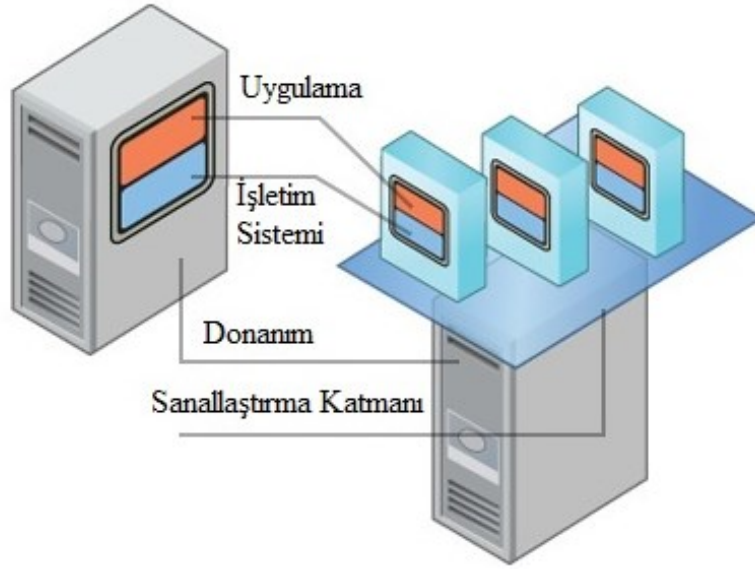
Bu tez çalışması esnasında pek çok bilgisayar sistemi ve bu sistemlerle ilgili problemler incelenmiş, kontrol teorisinin bu problemlerin çözümünde uygulanabilirliği konusunda bilgi sahibi olunmuştur. Bu bilgiler ışığında teknolojik açıdan önemli ve güncel bir konu olan sanallaştırma ve bulut bilişim konusundaki problemlere odaklanılmıştır.

Sanallaştırma ve bulut bilişim son zamanlarda çok ilgi çekmekte olan güncel konulardandır. Veri merkezleri, sanallaştırma ve bulut bilişimin en önemli uygulama alanlarından biridir. Veri merkezi; bilgisayar sistemleri, ağ elemanları ve depolama sistemleri gibi bilgisayar parçalarını bünyesinde bulunduran bir servistir. Veri merkezi, diğer bir deyişle depolama, yönetim ve verinin dağıtımı için fiziksel veya sanal olarak merkezileştirilmiş bir depodur. Veri merkezinin sanallaştırması, veri ve uygulamaların fiziksel sunuculardan genellikle farklı yerde bulunan sanal sunuculara taşınması yöntemidir. Sanal sunucu, aynı fiziksel makinenin birden fazla farklı

sunucu varmış gibi davranmasını sağlayan yazılımlar kullanır. Sanal ortamda kaynaklar sunucu ve sanal makinalar arasında paylaşılır. Veri merkezinde pek çok sunucu bulunabilir ve her bir sunucu Şekil 1.1’de gösterildiği gibi üzerinde birden fazla sanal makina bulundurabilir. Sanallaştırmada en önemli kısım, sanal makinadır (VM). Her bir VM Şekil 1.2’de gösterildiği gibi üzerinde işletim sistemi ve uygulama bulundurur ve izole edilmiş yazılım kabı gibi davranır. Her bir VM ayrı ve bağımsız olduğundan, birden fazla sanal makina tek bir sunucu üzerinde çalışabilir. Sanal makinalar üzerinde bir ya da daha fazla uygulama bulunabilir. Pek çok uygulama ve işletim sistemi bir sunucu üzerinde aynı anda çalışabilir. Bir VM, üzerinde veri tabanı/veri tabanı yönetim sistemi tutabilir. Bu durumda VM veri tabanı sunucusu olarak hizmet görür.



Şekil 1.1 : Geleneksel sunucu ile sanallaştırılmış sunucu arasındaki fark. Sanal ortamda bir sunucu birden fazla VM üzerinde bulundurabilir.



Şekil 1.2 : Sanal makina içinde işletim sistemi ve uygulama (APP) bulundurulur.

Veri tabanı yönetim sistemi, veri tabanında bilgi tutmayı, veri tabanındaki bilgiyi değiştirmeyi ve göstermeyi sağlar [6]. Ayrıca veri tabanı yönetim sistemi, bir yerde tutulan veri tabanına; kayıt ekleme, silme, değiştirme ve analiz etmeyi sağlayan araçları kullanıcılara temin eder. Veri tabanı, sınırlı kaynak kullanan akıllı kart uygulamalarından [7] önemli ölçüde kaynak tüketen büyük ölçekli kurumsal uygulamalara kadar farklı büyüklükteki uygulamalar için kullanılmaktadır. Veri tabanı üzerindeki iş yükü aktif kullanıcı sayısına ya da işlenen veri miktarına göre değişir. Veri tabanı sunucuları işlerin zamanında tamamlanmasını amaç edinir. Veri tabanı ve veri tabanı yönetim sistemleri tüm iş dalları için önem taşır ve dikkatli bir şekilde yönetilmelidir. Veri tabanı sunucular yaygın olarak kullanılır ve veri tabanı sunucuların veri merkezinde önemli işlevleri vardır. Bu sunucular üzerinde değişen iş yükü olabilir. İstenen, değişken iş yüküne rağmen, işlemlerin zamanında tamamlanmasıdır. Bu sebeple veri tabanı sunucular için servis kalitesi (QoS) önemli bir konudur. Ayrıca, veri tabanı sunucusu merkezi işlem birimi (CPU) kullanımı ve bellek tüketimi de kontrol altında olmalı, yani CPU ve bellek kaynakları gereğinden az, ya da çok tüketilmemelidir.

Sanallaştırılmış veri merkezinde, sanal makineler arasında ortak kaynakların kullanımının kontrolü ve kaynakların sanal makineler için en uygun olarak tahsis edilmesi problemi çözüm bekleyen açık bir konudur [5, 8, 9]. Bu tez çalışmasında bu problem üzerine odaklanılmıştır ve VM üzerinde yer alan veri tabanı sunucusunun CPU kullanımı ve bellek tüketimi, dinamik olarak CPU ve bellek kaynak tahsisi ile

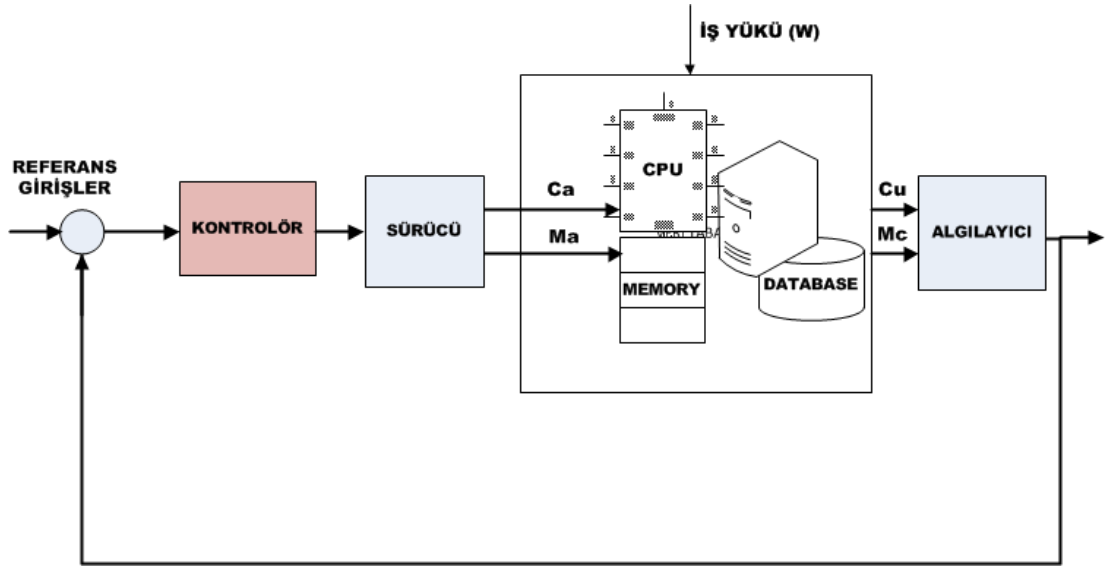
kontrol edilmesi hedeflenmiştir. CPU kullanımı ve bellek tüketimini kontrol etmek üzere oluşturulan alt yapı başarılı olduğu takdirde, veri merkezindeki tüm sanal makinaların CPU kullanımı ve bellek tüketimi, ilgili sanal makinalara dinamik olarak yapılan yeterli miktardaki kaynak tahsisi ile kontrol edilebileceği düşünülmüştür. Böylelikle, veri merkezindeki tüm uygulamalar için yeterli kaynak tüketimi sağlanabilir ve ihtiyacı olan diğer uygulamalar için daha fazla CPU ve bellek kaynakları sağlanabilir.

WinXP veri tabanı sanal sunucu makinası, Win7 veri tabanı sanal sunucu makinasıyla aynı kaynakları paylaşan diğer bir uygulamayı işletmek için fiziksel makina üzerinde oluşturulmuştur. Bu sebeple, asıl amacımız veri merkezindeki ve değişen iş yükü altındaki veri tabanı sanal sunucu makinalarının CPU kullanımı ve bellek tüketimini kontrol etmektir. Veri tabanı sunucusu, yaygın kullanımı ve veri merkezindeki önemi sebebiyle hedef sistem olarak seçilmiştir. Bu çalışmada, çoğu mevcut uygulamadan farklı olarak, veri tabanı sunucusunun CPU kullanımının ve bellek tüketiminin çoklu sistem çıkışları, çoklu sunucu parametreleri olan bellek tahsisi ve CPU tahsisi ile gerçek zamanlı olarak kontrol edilmesi amaçlanmıştır.

Ele alınan bu probleme, kontrol kuramının uygulanabilmesi için, sistemin matematik modeline gereksinim vardır. Bilişim sistemlerinin matematik modeli sistem tanıma teknikleri ile elde edilebilir. Bir sistem tanıma deneyi için, '*CPU ve bellek kaynak tahsisi*' sanal veri tabanı sunucusunun sistem girişleri, '*CPU kullanımı ve bellek tüketimi*' ise sistem çıkışları olarak dikkate alınır. '*İş yükü*', sistem çıkışlarında önemli bir etkiye sahip olduğundan dolayı modellemede bozucu giriş olarak dikkate alınmıştır. Sentetik iş yükü yaratıcısı, iş yükünü simüle etmede kullanılır. İş yükü; *SEÇME (LİSTELEME)*, *EKLEME*, *DÜZELTME*, *SİLME SQL* ifadelerinin ya da *PROSEDÜR* ve *FONKSİYONların* SQL Server veri tabanında çalıştırılmasıyla oluşturulur. Sistemin matematiksel modelini elde etmek için iki giriş işareti olan '*CPU ve bellek tahsis*' sistem girişleri sinüzoidal biçimde, bozucu işareti olan '*iş yükü*' rastgele değerler alacak şekilde sanal veri tabanı sistemine uygulanmıştır. Bu koşullar altında yapılan deneylerde, sistemin '*CPU kullanımı ve bellek tüketimi*' çıkışlarının da aynı frekansta sinüzoidal işaretler olduğu gözlenmiştir. Bu nedenle sistemin belirli giriş aralıklarında doğrusal bir davranışa sahip olduğu kararına varılmıştır. Tez çalışması kapsamında veri tabanı sanal sunucu makinasının çok giriş - çok çıkış (MIMO) modeli ve iki ayrı tek giriş-tek çıkış (SISO) sistem modelleri

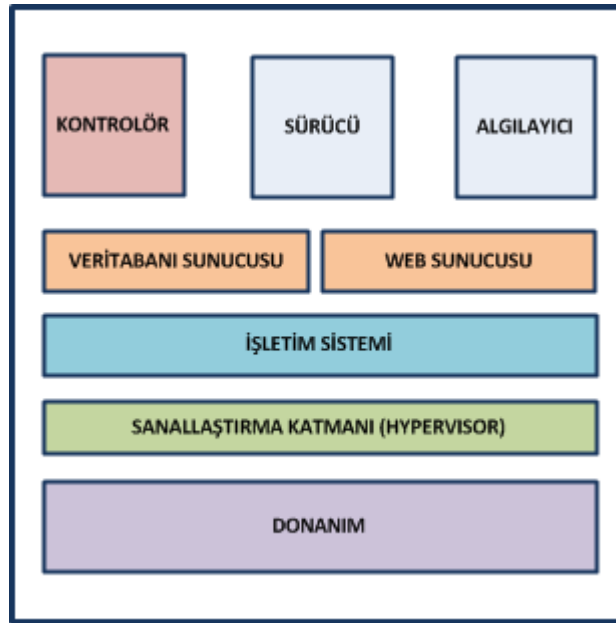
oluşturulmuştur. MIMO sistem modeli, CPU tahsis, bellek tahsis sistem girişlerinden CPU kullanım, bellek tüketim sistem çıkışlarına sistem davranışını tanımlar. İki ayrı SISO sistem modelinden biri sistemin, CPU tahsis girişinden, CPU kullanım çıkışına olan davranışını tanımlarken, diğer SISO sistem modeli, sistem davranışının bellek tahsis sistem girişinden bellek tüketim sistem çıkışına olan tarafını tanımlar. Yapılan deneylerde, CPU kullanımına ilişkin sistem çıkışının, asıl olarak CPU tahsis sistem girişinden etkilendiği diğer sistem girişinden ise hemen hemen hiç etkilenmediği gözlemlenmiştir. Bu sebeple, bellek tahsis sistem girişi bu SISO modelinden çıkarılmıştır. Bellek tüketimine ilişkin sistem çıkışı ise sadece bellek tahsisine ilişkin sistem girişinden etkilenir. Bu sebeple CPU tahsis sistem girişi ikinci SISO sistem modeline dâhil edilmemiştir. MIMO sistem modeli ve çoklu SISO sistem modellerinin doğruluğu test edilmiş ve gerçek sistem dinamiklerini ifade etmedeki başarıları karşılaştırılmıştır. Elde edilen sistem modelinin geçerli olduğu giriş aralığı yani sistem giriş ve çıkışlarının alabileceği maksimum ve minimum değerler, yapılan deneylerle belirlenmiştir. Sistem modelini elde etmede sistemin giriş-çıkış ilişkisinden faydalanan kara-kutu yaklaşımı kullanılmıştır [10]. MATLAB sistem tanıma aracı (SIT) sistem modelini elde etmede yardımcı olmuştur. Sistem modelini elde etmek için lineer olmayan sistem tanıma teknikleri de dâhil olmak üzere birçok farklı yöntem denenmiş, başarıımı en iyi olan model kontrolör tasarımı kullanılmıştır.

Uygulamalarda, veri tabanı sunucusunun CPU kullanımı ve bellek tüketimi istenilen seviyede tutulmak istenir. Bu amacı gerçekleştirmek için bulut ortamında zamanla değişen yoğun iş yükü altında çalışan, iki ayrı geri besleme kontrol sistemi tasarlanmış ve gerçekleştirilmiştir. Bu kontrol sistemleri; oransal-integral (PI: Proportional-Integral) ve integral etkili Linear-Quadratic Gaussian (LQG+I) kontrolörlerdir. Tasarlanan kontrolörlerin gerçek uygulamalara eş bir deney ortamında testlerini yapılmasını sağlamak üzere sanal veri merkezinde bir test ortamı yaratılmıştır. Algılayıcı, sürücü ve kontrolör Şekil 1.3'te gösterildiği gibi geri besleme çevrimi içinde uygun bir şekilde birbirine bağlanarak kapalı çevrimli sistem oluşturulmuştur. Tasarlanan kontrolörler önce MATLAB ortamında simülasyon ile doğrulanmış daha sonra gerçek ortam olan emülatör sistemde gerçekleştirilmiş, test edilmiş ve kullanılmıştır.



Şekil 1.3 : Sanal ortamdaki veri tabanı sunucu sistemi için geri besleme kontrol sistemi.

Donanım, yazılım ve kontrol elemanları Şekil 1.4'te blok diyagram olarak gösterilmiştir.



Şekil 1.4 : Sanal veri tabanı sunucusu üzerindeki donanım, yazılım ve kontrol elemanları.

2. LİTERATÜR ÖZETİ VE ÖN TASARIM ÇALIŞMASI

Kontrol kuramı ve kontrol tasarım teknikleri son yıllarda bilgisayar sistemlerini ilgilendiren konularda kullanılmaya ve bilgisayar sistemleri ile alakalı problemlerin çözümde uygulanmaya başlanmıştır. Bu ön çalışma kapsamında, kontrol tekniklerinin bilgisayar sistemlerini ilgilendiren bazı problemlerin çözümünde uygulanmasına ilişkin ilgili literatür incelenmiştir. Ayrıca bu çalışmada kontrol kuramı ve teknikleri bir bilgisayar sistemi olan çevrimiçi fatura ödeme sistemi ile ilgili bir problemin çözümü için kullanılmıştır. Çevrimiçi ödeme sistemindeki tıkanıklıkları fark eden ve sistem yöneticisini uyaran bir geri beslemeli kontrol sistemi tasarlanmış ve uygulaması gerçekleştirilmiştir. Bulunan sonuçlar ayrıntılı olarak tartışılmıştır.

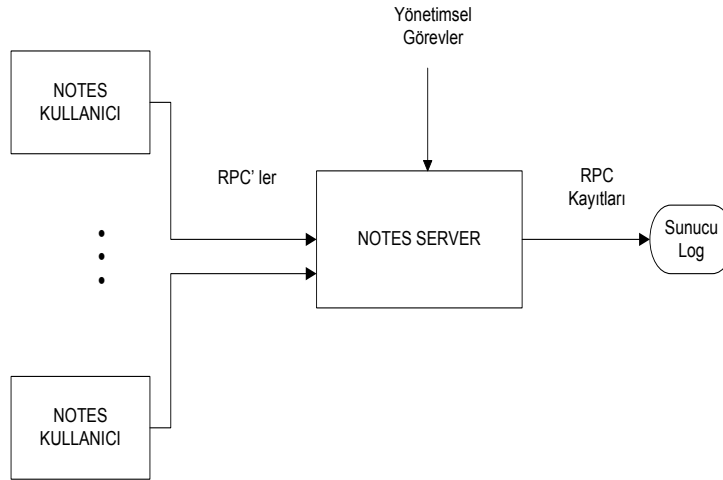
Kontrol teorisinin özellikle veri ağları, işletim sistemleri, middleware olarak isimlendirilen Web hizmet sağlayıcı ve veri tabanı hizmet sağlayıcıları ayrıca multimedya ve güç yönetim sistemleri gibi bilgisayar sistemleri ile alakalı olan konularda kullanılması ve bu konulara uygulanması ile ilgili çalışmalara doksanlı yılların başlarından itibaren başlanmıştır [11]. Kontrol kuramının veri ağları konusu ile alakalı akış kontrol problemlerine uygulanması araştırmacıların ilgisini çeken bir konu olmuştur. Bu konuda yapılan çalışmalardan ilki kuyruklardaki veri paket akışını düzenleyen *rate allocating sunucu* kavramının oluşmasına yardımcı olan çalışmalardır [12]. Diğer bazı çalışmalar ise kontrol kuramının veri transferi sırasında Transmisyon Kontrol Protokol (TCP)'deki kısa süreli seviye değişimlerinin düzenlenmesi ile ilgilidir [13]. Son zamanlarda TCP'nin ayrıntılı modelleri, sıvı akış yaklaşımı kullanılarak elde edilmiş ve bu yaklaşım ağdaki yönlendiricilerin (router) tampon alan yönetimi konusunda kullanılmıştır [14]. Kontrol kuramının yönlendiricilerde veri depolama yönetiminde ve asenkron veri aktarım yöntemi (ATM) veri ağlarında kullanılması konusunda çalışmalar 90'lı yıllarda çok artmıştır [15, 16].

Bilgi işleme ve depolama sistemlerindeki kontrol problemleri üç sınıfa ayrılabilir; Beklenen hizmet kalitesinin sağlanması, kaynak kullanımını düzenlenmesi, aşırı iş yükü ya da yanlış sistem tasarımı gibi sebeplerden oluşabilecek aşırı kaynak kullanımının düzenlenmesidir [17, 18, 19]. Diğer bir problem de tüm sistemin en uygun şekle sokulmasıdır. Ön tasarım çalışmasında kaynak kullanımı düzenlenmesi problemine geri beslemeli bir sistemle çözüm bulunmaya çalışılmış ve elde edilen sonuçlar tartışılmıştır.

2.1 Literatür Araştırması

İlgili literatürde kontrol kuramı ve tasarım tekniklerinin temel bazı bilgisayar sistemi problemlerinin çözümünde kullanıldığı görülür [20, 21, 11, 22, 23, 24, 25, 26, 27, 28, 17, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]. Bunlardan bazılarına bu bölümde yer verilmiştir.

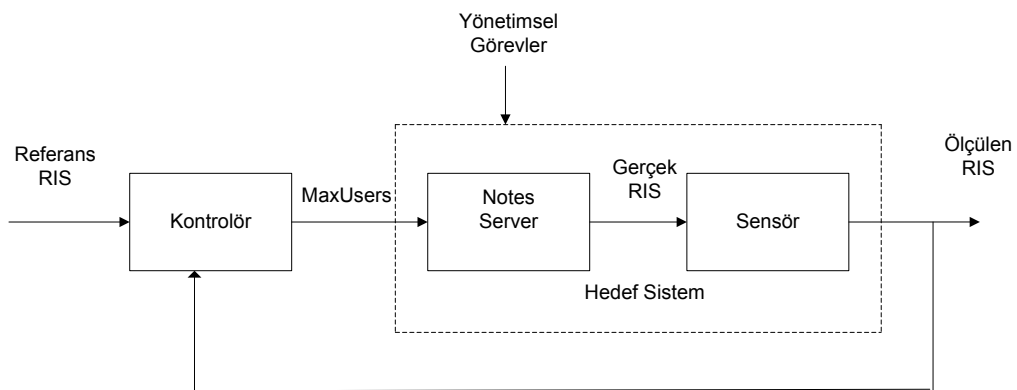
2.1.1 IBM lotus notes sunucu



Şekil 2.1 : IBM Lotus Domino Sunucusundaki sistem elemanları ve veri akış şeması.

IBM Lotus Domino Sunucusu bir e-posta sistem sunucudur. Bu sunucu, Notes Sunucu olarak da isimlendirilir. Lotus Notes [18, 23], Şekil 2.1’de gösterildiği gibi bir kullanıcı sunucu uygulamasıdır. Kullanıcılar Notes kullanıcı makineleri vasıtasıyla sunucuya bağlanıp e-posta ve diğer uygulamalara erişim sağlarlar. Kullanıcılar sunucu üzerinde bulunan e-posta veri tabanına bağlanarak e-postalarına erişirler, veri tabanı üzerinde değişiklik, ekleme, düzeltme yapabilirler. Kullanıcı ile sunucu arasındaki iletişim uzaktan izlek çağırımları (RPC) şeklinde olur. Sunucu

üzerinde tamamlanan her işlem “log” tablolarında kayıt altına alınır. Kullanıcılar tarafından başlatılan işlemler dışında yönetimsel işlemler de sunucu tarafından gerçekleştirilir. Sistem yöneticileri, sunucu tarafından gerçekleştirilen işlemlerin sayısını (sistemin yükünün bir ölçüsü olarak) kontrol etmekten sorumludurlar. Bu sayı (RIS) olarak isimlendirilir. RIS sunucu tarafından gerçekleştirmekte olan işlerin sayısını gösterir. Eğer RIS çok artarsa, CPU, bellek ve diğer kaynakların kullanımı aşırı artar ve bu da sistem performansının düşmesine sebep olur. RIS bir konsol arayüz aracılığıyla dinamik olarak ayarlanabilen *MaxUsers* parametresiyle sınırlandırılabilir. *MaxUsers* parametresi sunucuda açılan oturum sayısını ya da diğer bir deyişle kullanıcılar tarafından Notes sunucuda açılan bağlantı sayısını ifade eder. Sistem yöneticisi *MaxUsers* parametresini ayarlayarak sunucuda gerçekleştirilen iş sayısını (RIS) kontrol eder. Notes sunucusu için kontrol amacı RIS parametresini düzenli bir seviyede tutmaktır. Şekil 2.2, kontrolörün bu düzenlemeyi nasıl yaptığını göstermektedir. Hedef sistem IBM Lotus Domino Sunucu ve sunucu üzerindeki log tablosundan yararlanarak RPC istatistiklerini elde etmeyi sağlayan algılayıcıdan oluşur. Sistem yöneticisi RIS için referans girişi belirler. Kontrol hatası referans girişi ile ölçülen RIS arasındaki farktır. Kontrolör kontrol hatasından yola çıkarak hedef sisteme uygulanacak kontrol işaretini (*MaxUsers*) üretir. Algılayıcı işlerin (RPC) tamamlanmasını ve log tablosuna yazılmasını bekler. Bu beklemler geri besleme kontrol sistemi üzerinde önemli etkiye sahiptir [18].

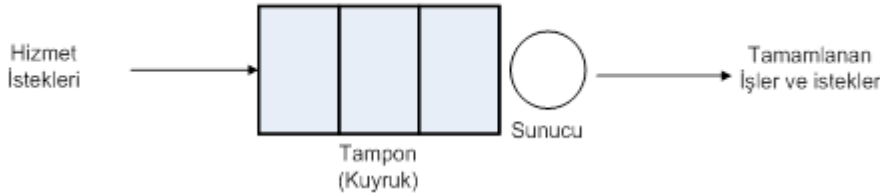


Şekil 2.2 : IBM Lotus Domino Server için kontrol sistemi.

2.1.2 Kuyruk sistemleri (Queueing Systems)

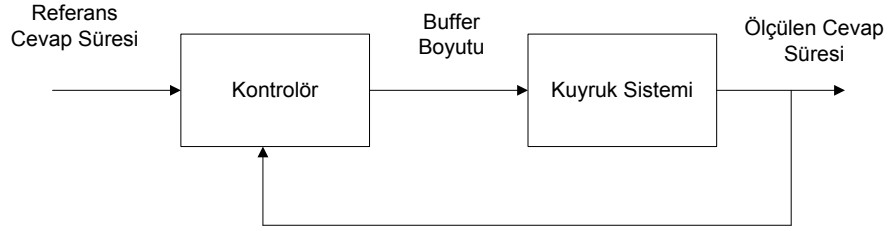
Kuyruk sistemleri, bilgisayar sistemlerinin performansının belirlenmesinde yaygın olarak kullanılır [34, 35, 21, 19]. İş istekleri ya da kullanıcı istekleri geldiği zaman

bunlar kuyrukta tamponlanır. Şekil 2.3'te dikdörtgenler iş isteklerini ifade eder. Sırası gelen iş süreç edilmek üzere kuyruktan alınır, bu iş Şekil 2.3'te daire olarak gösterilmektedir. İstekler ait oldukları hizmet sınıfına göre birbirinden ayrılır. Bu ayırımın sebebi isteğin bağlı olduğu hizmet sınıfına göre kaynak ayırmaktır. İnternette bir alış veriş sitesi bu sisteme örnek olarak gösterilebilir. Bir ürün alma ihtimali yüksek olan kullanıcılara daha fazla kaynak ayırarak, sadece sitede dolaşan, herhangi bir ürün alma ihtimali düşük olduğu düşünülen ya da siteye ilk defa giren ve siteye üye olmayan kullanıcılara daha az kaynak ayırarak kullanıcı istekleri ait oldukları hizmet sınıfına göre birbirinden ayrılmış olur. Kuyruk modeli hizmet seviyesinin amaç ölçütü olduğu problemlere uygulanabilir. Örneğin kredi kartı sorgulamaları beş saniye içinde tamamlanmalıdır. Kredi kartı sorgulamalarının beş saniyeyi geçmesini engellemek için diğer iş isteklerine bir sınır getirilebilir ya da örneğin Hypertext Transfer Protocol (HTTP) Redirect komutu ile başka bir sisteme yönlendirilebilir. Kuyruk sistemlerinde çıkış zamanı, kuyruktaki işlerin sayısı ya da diğer bir deyişle tampon alanın büyüklüğü kontrol edilerek istenilen seviyede tutulabilir. Eğer tampon alanda yer kalmadıysa istekler yönlendirilir.

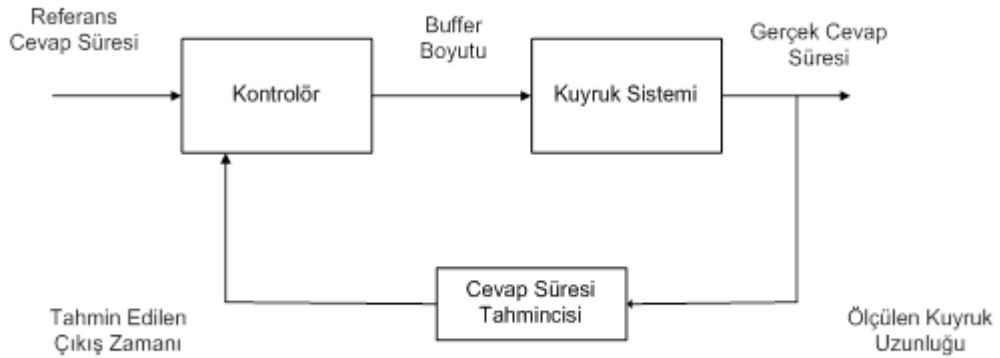


Şekil 2.3 : Tek sunucu kuyruk sistemi (single-server queueing system).

Şekil 2.4 (a) kuyruk sistemi için çıkış zamanını düzenleyen geri besleme kontrol sistemini göstermektedir. Bu sistemin amacı olması gereken çıkış sürelerine uyararak maksimum sayıda isteğe yanıt verebilmektir [34, 35]. Kuyruk sisteminde referans giriş istenilen çıkış zamanıdır. Kontrol işareti ise tampon (buffer) büyüklüğüdür. Kontrol hatası ise referans çıkış zamanından ölçülen çıkış zamanı çıkartılarak hesaplanır. Şekil 2.4(a)'da gösterilen sistemde istenmeyen bazı durumlar oluşabilir. Uzun süreli işlemler sebebiyle çıkış zamanı ölçümünde gecikmeler oluşur. Bu gecikmelerin sebebi çıkış zamanının kayıt edilebilmesi için işin tamamlanmış olması gerekliliğidir. İş tamamlanamadığı için çıkış zamanı kaydedilemez ve ölçülemez bu da geri besleme kontrol sisteminde istenmeyen gecikmelere sebebiyet verir.



(a) Cevap süresinin ölçümü



(b) Çıkış zamanının bir transducer elemanı ile tahmin edilmesi

Şekil 2.4 : Kuyruk sisteminde transducer elemanının yanıt süresi tahmininde kullanılması.

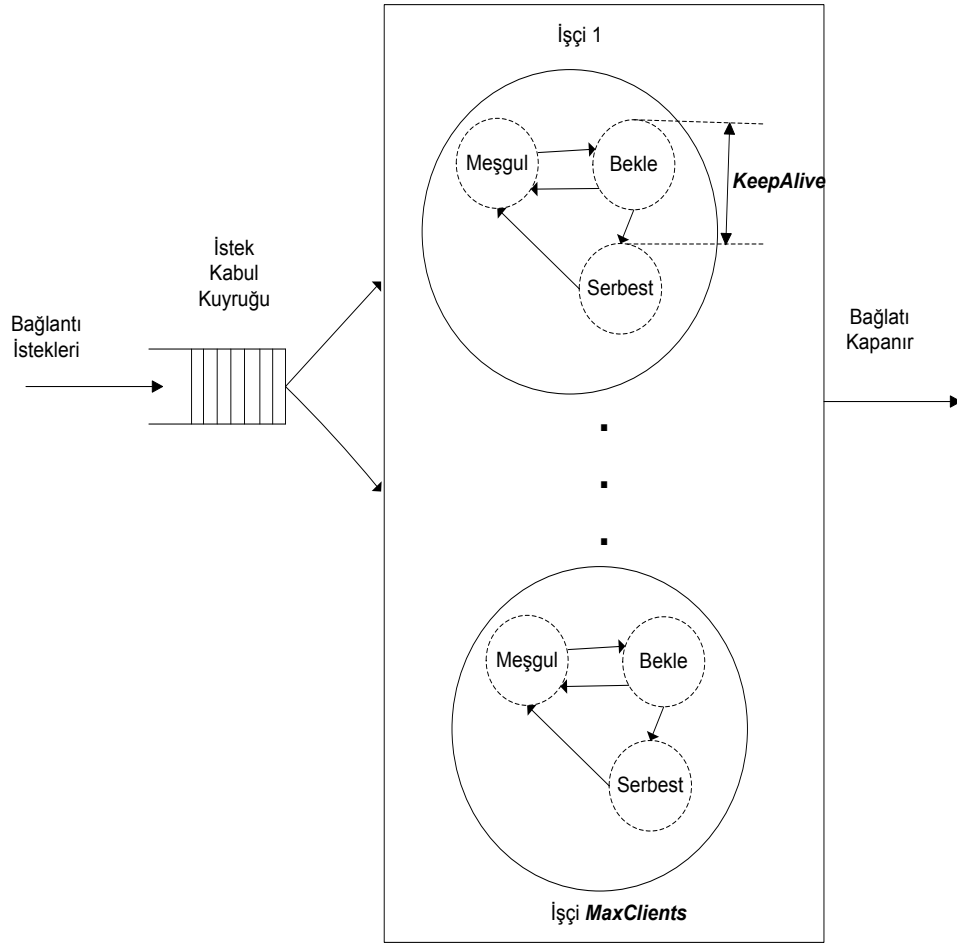
Şekil 2.4(b) çıkış süresinin ölçümünün kuyruk uzunluğu bilgisi kullanılarak transducer tarafından hesaplanmasını göstermektedir. Böylelikle işin tamamlanması için beklemek yerine çıkış zamanı tahmin edilerek anında ölçülebilmektedir.

2.1.3 Apache HTTP sunucu (Server)

Web sunucuları bilgi paylaşımında ve elektronik ticarete önemli bir yere sahiptir [20, 22, 24, 25, 28, 17, 29, 30, 32]. En tipik yapıda son kullanıcılar, kullanıcı iş istasyonları (client workstations) vasıtasıyla Hypertext veri transfer protokolü (HTTP) ile Web sunucuya istekte bulunurlar. Bu istekler bir sayfanın taranması ya da bir dosyanın indirilmesi gibi istekler olabilir. Web sunucuları bu isteklere uygun cevabı dönmeye çalışırlar.

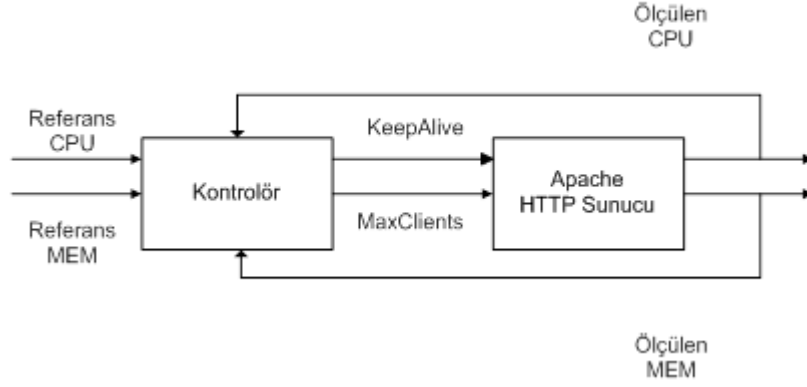
Apache HTTP sunucusu Şekil 2.5'te görüldüğü gibi işçilerin içinde bulunduğu bir havuz gibidir. İşçiler ya bir süreç ya da bir thread olarak çalışır ve kullanıcı isteklerini gerçekleştirmeye çalışır. Kullanıcı istekleri sunucudaki kuyrukta tutulur. Bu istekler bir işçi işini tamamlayıp açığa çıkana kadar kuyrukta bekler. İşçi eğer 'idle' durumunda ise açığa çıkmıştır, yeni bir iş üstlenebilir. İşçi eğer bir işi tamamlamakla meşgul ise işçi bu durumda 'busy' durumundadır. Yaygın olarak kullanılan HTTP 1.1 protokolü 'persistent' bağlantı sağlayabilir. Bunun anlamı şudur: İş tamamlandıktan sonra sunucuya olan bağlantı koparılmaz işçi bekleme 'wait' durumuna geçer ve bağlantı açık kalır. Böylelikle aynı kullanıcıdan gelen sonraki istekler daha verimli ve hızlı bir şekilde gerçekleştirilir. İşçi bekleme 'wait' durumundayken bağlı olduğu kullanıcı dışındaki kullanıcıların isteklerini kabul etmez.

İki yapılandırma parametresi HTTP sunusunun kaynak kullanımını ve performansını etkiler. Bunlardan biri sunucuya bağlı olan maksimum kullanıcı sayısı (*MaxClients*) parametresidir. *MaxClients* parametresi işçi havuzunun büyüklüğüne sınır getirir. Böylelikle sunucunun iş yapma kapasitesine de sınır getirilmiş olur. *MaxClients* parametresi için daha büyük değer belirlenirse HTTP sunucusu daha fazla kullanıcı isteğine yanıt verebilir. Eğer bu parametre çok büyük seçilirse bu sefer de CPU ve bellek kullanımları aşırı derecede artar ve sunucunun istekleri gerçekleştirme süresi uzar dolayısıyla performans düşer.



Şekil 2.5 : Apache HTTP sunucu mimarisi ve oturum akışı.

Diğer yapılandırma parametresi *KeepAlive* parametresidir [20]. Bu parametre bir işçinin hizmet verdiği kullanıcı ile olan bağlantısı kesilmeden önce bekleme ‘wait’ durumundan serbest ‘idle’ duruma geçmesi için işçiye tanınan maksimum zamanı kontrol eder. Eğer bu parametrenin değeri çok büyük olursa CPU‘dan tam olarak yararlanılamaz. Çünkü isteklerinin gerçekleştirilmesini isteyen diğer kullanıcılar sunucuya bağlantı kuramaz. Time out süresini kısaltmak işçilerin daha uzun zaman meşgul ‘busy’ durumunda daha kısa zaman ‘wait’ bekleme durumunda kalmasını sağlar. Bu CPU kullanımını artırır. Eğer time out süresi çok kısa olursa işler tamamlanmadan -prematüre iken- bağlantı kopar bu da CPU kullanımını artırır çünkü aynı kullanıcıdan gelen sonraki istekler sebebiyle bağlantı tekrar kurulmaya çalışılacaktır.



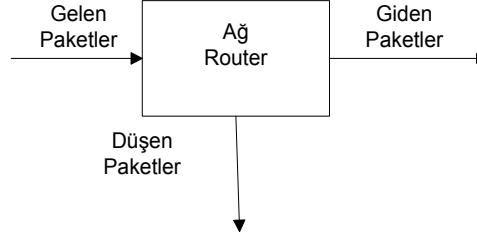
Şekil 2.6 : Apache HTTP sunucu hedef sistemi için düzenleyici kontrol.

Web sunucuların yönetiminde bir kontrol problemi CPU ve bellek gibi kaynakların kullanımını düzenlemektir [25]. Şekil 2.6, HTTP sunucu için bu düzenlemenin nasıl sağlandığını göstermektedir. Sistem yöneticileri CPU ve bellek kullanımları için referans değerleri belirler. Kontrolör bu iki referans giriş ve ölçülen CPU ve bellek kullanımlarından yararlanarak, *KeepAlive* ve *MaxClients* kontrol işaretini üretir.

Apache HTTP sunucu bir MIMO sistemdir. Kontrol işaretleri olan *KeepAlive* ve *MaxClients* kurularak ölçülen çıkış büyüklükleri olan CPU ve bellek kullanımları istenilen değerde tutulmaya çalışılır. Her iki kontrol girişi CPU kullanımını etkilemektedir. Bu sistem için durum uzay modeli oluşturulabilir ve sistem durum uzayında incelenebilir [40].

2.1.4 Yönlendirici tampon alanı taşmaları için rastgele erken uyarı sistemi (Random early detection of router overloads)

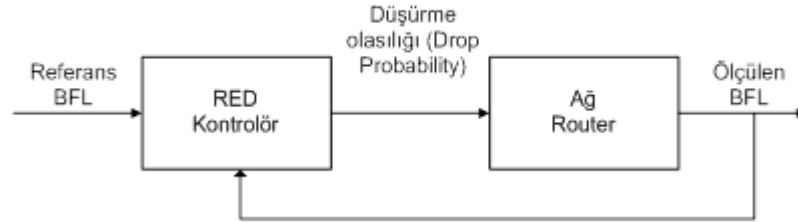
İnternet için TCP merkezi bir elemandır. TCP (transmission control protocol) ağ düğümleri arasında bir uçtan diğer uca iletişim sağlar. TCP tasarımcıları, ağda tıkanıklıklar olduğu zaman trafiği düzenlemekle uğraşırlar. Bu düzenlemenin olduğu yerlerden biri veri paketlerini yönlendiren bir fiziksel ağ elemanı olan yönlendiricilerdir. Yönlendiricilerin belli bir tampon genişlikleri vardır. Bu sebeple trafik tıkanıklıklarında taşmayı engellemek için bazı paketler ihmal edilir [11]. Yönlendirici tampon alanında taşma olmadan önce trafik zaten tıkanmıştır. Bu durumu engellemek üzere *random early detection* (RED) kritik tampon alanlarının ne kadarının dolduğunu hesap eder [14]. Bu oran tampon doluluk seviyesi tampon alan doluluk seviyesi (BFL: *buffer fill level*) olarak gösterilir.



Şekil 2.7 : RED (Random early detection) ile çalışan yönlendirici.

Şekil 2.7 Yönlendiricide taşma olmadığı durumda rastgele olarak paket düşürme durumunu açıklar. Eğer BFL küçükse hiçbir paket düşürülmez, her paket eksiksiz yönlendirilir fakat BFL zamanla büyürse sürekli olarak daha fazla paket düşürülür. Düşürülen paketlerdeki veri aktarılamamış olur.

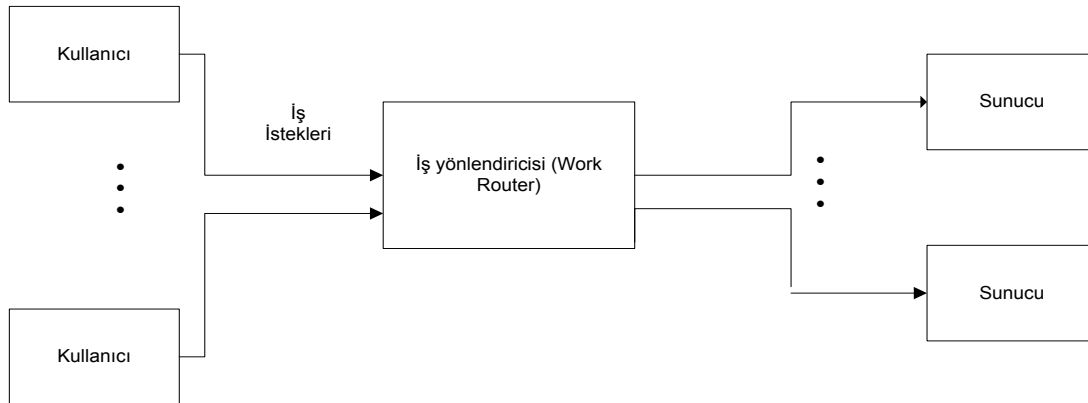
RED ağ trafiğini azaltır ve ağ akışını düzenler. Şekil 2.8’de RED sistemi bir geri besleme kontrol sistemi olarak gösterilmektedir. Yönlendiriciden elde edilen tampon doluluk seviye bilgisi referans tampon doluluk seviye bilgisi ile karşılaştırılarak yönlendirici için bir paket düşürme oranı hesaplanır. Bu oran yönlendirici için bir kontrol işaretidir. BFL büyükse paket düşürme olasılığı artar ve yönlendiricide daha fazla paket düşürülür.



Şekil 2.8 : Yönlendiricide tampon doluluk seviyesi geri besleme kontrolü.

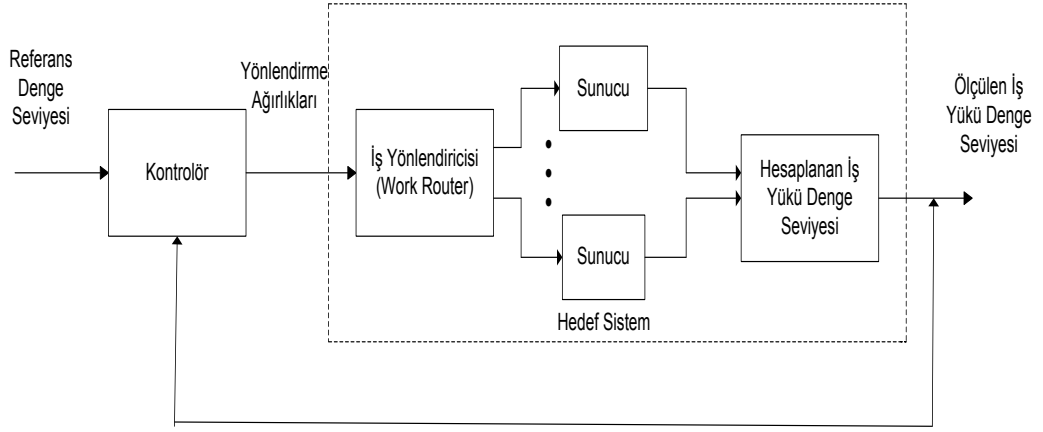
2.1.5 İş yükü dengeleme (Load balancing)

Yüksek performanslı bilgisayar sistemi kurmak için kullanılan yöntemlerin en yaygın olanlarından biri benzer şekilde yapılandırılmış sunucuları bir arada kullanmaktır. Benzer yapılandırmaya sahip sunucuların oluşturduğu sistem topluluğu *sunucu çiftliği* (*server farm*) ya da *küme* (*cluster*) olarak bilinir [5, EK A]. Bu sistemlerde iş yükünü dengelemek için geri beslemeli kontrol sistem önerisi veren bazı çalışmalar az da olsa literatürde vardır [11, 26, 36, 37]. Şekil 2.9 iş yükü dengeleme sisteminin mimarisini gösterir. Dosya indirme, Web sayfasına erişim gibi iş isteklerinde bulunan pek çok bilgisayar kullanıcısı vardır. Tüm iş isteklerinin, aynı sunucu ya da sunuculara gönderilmesi yerine, yönlendiriciler vasıtasıyla iş yükü seviyelerine göre en uygun sunucuya gönderilmesi tercih edilir. İş yüklerine ve/veya iş yapma kapasitelerine göre, her bir sunucuya bir ağırlık oranı atanır. Böylece yüksek ağırlık oranına sahip sunuculara daha çok istek yönlendirilmesi imkânı elde edilmiş olur.



Şekil 2.9 : İş yükü dengeleme sistemi mimarisi.

Şekil 2.10 sunuculara dengeli iş yükü dağıtımını sağlayan geri besleme kontrol sistemini gösterir [11]. Hedef sistem kesikli çizgi ile çevrelenmiş bloklardan oluşur. *Hesaplanan iş yükü denge seviyesi bloğu* vasıtasıyla belirlenen bir ya da daha çok ölçüte göre sunucuların iş yüklerinin denge durumu hesaplanır. Örneğin denge seviyesi hesaplanırken sunucuların maksimum ve minimum kullanımları arasındaki fark belirleyici olabilir. Yönlendirme ağırlıkları (*routing weights*) [11] kontrol girişidir. Ölçülen çıkış ise denge seviyesidir. Kontrolör, yönlendirme ağırlıklarını referans denge seviyesi ile ölçülen denge seviyesi arasındaki farka göre hesaplar.



Şekil 2.10 : İş yükü dengeleme için geri besleme kontrolü.

2.1.6 Duraksız akış mimarisi (Streaming media)

Günümüzde çoklu ortam (multimedya-video ve/veya ses) verilerinin ağ üzerinden iletiminin popüler hale gelmesiyle duraksız akış uygulamalarında QoS gittikçe önem kazanmaktadır. Çoklu ortam servislerinin sağlanmasında yüksek bant genişliği gereksinimi sınırlı ağ kaynaklarının en etkin şekilde kullanımını oldukça önemli hale getirmiştir. Duraksız akış uygulamalarında veri sunucudan gönderilirken aynı zamanda istemcide tamponlanarak görüntülenmektedir. Çoklu ortam verilerinin ağ üzerinden iletmek için duraksız akış uygulaması veya yükleme kullanılabilir. Duraksız akış uygulamasında veri gönderimi ve oynatımı paralel olarak gerçekleştirilir, yükleme yönteminde ise, kullanıcı tüm veriyi yükledikten sonra oynatıma geçer. Duraksız akış uygulaması, yükleme yönteminde yaşanan uzun süreli bekleme gereksinimini ortadan kaldırmaktadır [38, 39].

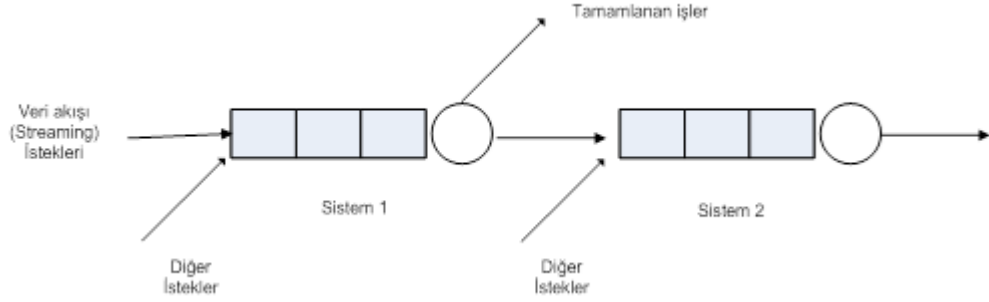
Duraksız akış uygulamasının başarımını etkileyen kısıtlar şunlardır:

1. Bant genişliği: Çoklu ortam verilerinin, diğer veri türlerine oranla çok daha yüksek bant genişliğine ihtiyaç duyarlar. Ancak, sınırlı ağ kaynakları bu kadar büyük bir bant genişliği ihtiyacını karşılamada yetersiz kalmaktadır. Uygulamalar sırasında çeşitli nedenlerden dolayı işletim ve iletişim sistemleri kaynaklarının yetersiz kaldığı durumlarda yaşanan darboğaz, uygulamaların başarımını olumsuz yönde etkilemektedir.
2. Gecikme: İstemcide görüntülemenin sürekliliğinin korunması açısından, uçtan uca gecikme değerlerinin belli bir üst sınırı asmaması gerekmektedir. Gecikmeler görüntüleme sırasında sürekliliğin bozulmasına yol açarlar. Tıkanıklıklar gecikme sınırlarının aşılmasına neden olabilmektedir.

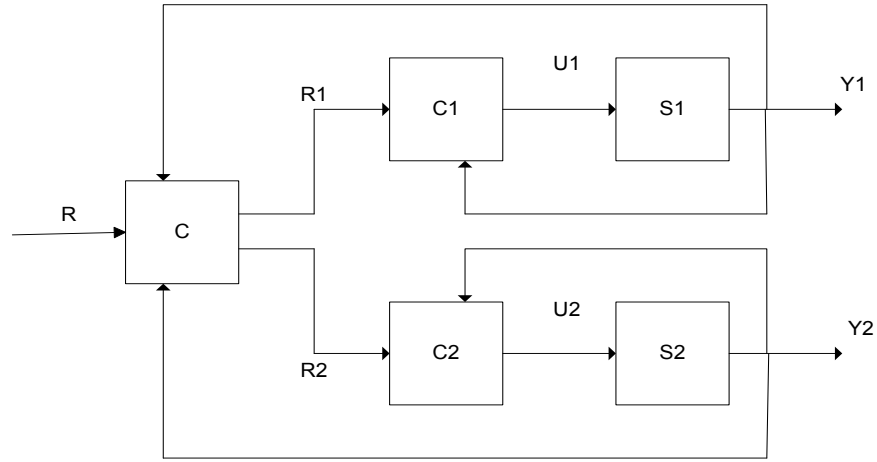
3. Kayıplar: Kayıp paketler, görüntüleme kalitesinin bozulmasına yol açabilecekleri gibi kesilmesine de neden olabilirler. Bu nedenle iyi bir görüntüleme kalitesi için kayıp oranının belli değerlerin altında tutulması gerekmektedir. Ancak, günümüz ağ kaynakları bu tip bir garantinin sağlanması konusunda da yetersiz kalmaktadır. Özellikle, tıkanıklık sırasında gözlenen yüksek kayıp oranları görüntüleme kalitesinde önemli bozulmaların yaşanmasına neden olmaktadır.

4. Heterojenlik: Video konferansları, Internet TV gibi uygulamalarda bire yayım tekniğinin kullanılması aynı videonun bir kaç kopyasının ağ üzerinde taşınmasına yol açtığından bant genişliği kullanımını olumsuz yönde etkilemektedir. Bu yüzden bu tür uygulamalarda çoğalayım (unicast) tekniği tercih edilir. Ancak, çoğalayım uygulamalarının önemli bir sorunu, ağ üzerindeki heterojenlik nedeniyle bireyiyim (multicast) tekniğinde yaşanan esnekliğin sağlanamamasıdır. Bireyiyim tekniğinde her alıcı birbirinden bağımsız olarak gönderici ile iletişime girerek değişik kalitede video akış isteyebilir. Ancak, çoğalayım mekanizmasında videonun tek bir kopyası gönderildiği için tüm alıcılar, aynı kaliteye sahip tek bir video işaretini alırlar.

İnternet üzerinden müzik, internet üzerinden televizyon ve canlı müzik gibi uygulamalar bilginin sürekli olarak belli bir seviyede aktarımını gerektirir. Hizmet seviyesi amaç ölçütü olarak seçilir ve yanıt sürelerini belli seviyelerin altında tutmaya çalışılır. Böylelikle hizmet kalitesi istenilen seviyede tutulmuş olur, örneğin video görüntüsü bozulmaz, ses kesilmez. Duraksız akış mimarisi Şekil 2.11 ile gösterilmiştir. Kullanıcı istekleri duraksız akış istekleri ve diğer istekler olmak üzere iki sınıfa ayrılır. Duraksız akış istekleri her iki kuyruktan kısıtlı zamanda geçerek tamamlanmış olmalıdır. Böylelikle hizmet seviyesi amaç ölçütü yerine getirilmiş olur. Şekil 2.12 ile duraksız akış mimarisinde amaç ölçütü için geri besleme kontrol sistemi gösterilmektedir. Şekil 2.12'deki S1 ve S2 Şekil 2.11'deki sistem 1 ve sistem 2'ye karşılık gelir. Şekil 2.12'de her bir sistem ayrı olarak C1 ve C2 kontrolörler ile yönetilir. Daha yüksek seviyede bir kontrolör bu kontrol sistemleri için referans girişleri düzenler. R1, R2 referans girişler, U1, U2 kontrol girişleri, Y1, Y2 sistem çıkışlarıdır.



Şekil 2.11 : Duraksız akış mimarisi için bir örnek. Bu şekilde iki kuyruk sistemi mevcuttur, örneğin CPU ve disk.



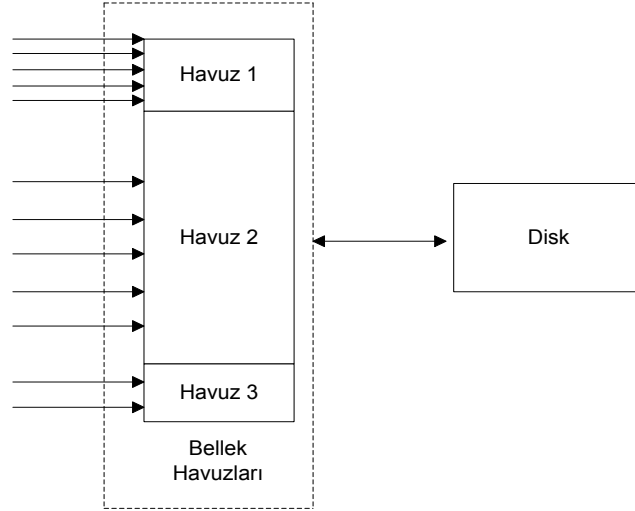
Şekil 2.12 : Duraksız akış mimarisinde hizmet seviye amaç ölçütünün elde edilmesini sağlayan geri besleme kontrol sistemi.

2.1.7 Hizmet tipine göre ön bellekte farklı boyutta tampon alanlar yaratma (Caching with differentiated service)

Bilgisayar sistemlerinde performansı arttırmak için yaygın olarak, ön bellek kullanılır. Web sunucu sayfaları ana bellekte tutulduğu takdirde diske erişim olmadığı için Web sayfalarına çok daha hızlı erişilir. Web ortamında ön bellek kontrolü ile ilgili olarak, literatürde bazı araştırmaların sonuçları verilmiştir [20, 11, 27, 31, 33]. Diğer taraftan ana bellekte tutulabilecek sayfa sayısı sınırlıdır. ‘Vuru’ (‘hit’) ifadesi diske erişim gerektirmeyen bellek erişimini ifade eder. Erişilmek istenen sayfalar bellekte tutuluyorsa kullanılır. ‘Iska’ (‘miss’) ifadesi ise bellekte tutulamayan sayfalar için disk erişimi gerektiren bellek erişimidir. Bu durumda sayfalar ana bellekte tutulmadığından ya da kısmen tutulduğundan disk erişimi gereklidir.

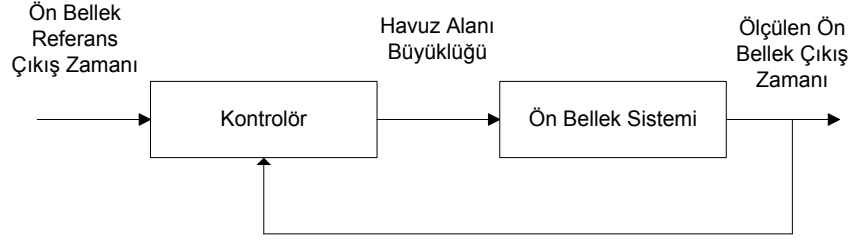
İyi bir ön bellek sistemi, erişilme olasılığı en yüksek olan sayfaları ana bellekte tutar. Ön bellek sistemlerinin performansı, ortalama erişim zamanıyla ölçülür. Ön bellekteki sayfalara erişim zamanı diskteki sayfalara erişim zamanından çok daha az olduğu için performans daha ziyade vuru olasılığı ile belirlenir. Ön bellek yönetimi için doğru algoritmalar kullanıldığı takdirde, vuru olasılığı ön bellek büyüklüğü arttıkça artar.

Web içerikli bir servis sağlayıcısı birkaç farklı sınıfta kullanıcıya hizmet veriyor olabilir. Kullanıcı sınıflarındaki bu farklılık kullanıcıların hizmet karşılığı ödediği ücretle alakalı olabilir. Kullanıcılara farklı hizmet sunmak ön bellek sistemi gerektirir. Bu sistem vasıtasıyla kullanıcılara ödedikleri ücretle orantılı olarak hizmet sunulur. Bazı kullanıcılar sayfalara daha hızlı erişebilir çünkü bu sınıftaki kullanıcılara daha geniş bellek alanı ayrılır ve vuru olasılığı yüksek olur, diskten okuma az olduğundan yanıt alma süresi kısalmır. Diğer kullanıcılar için daha az ön bellek alanı ayrıldığından vuru olasılığı azalır dolayısıyla diskten okumalar artar bu da yanıt süresini uzatır. Şekil 2.13 hizmet farklılığı durumunda ön bellek sistemini çalışmasını gösterir.



Şekil 2.13 : Farklı hizmeti destekleyen ön bellek sistemi.

Şekil 2.14 ön bellek sistemi için geri besleme kontrol sistemini göstermektedir. Her bir bellek havuzu ayrı olarak kontrol edildiğinden dolayı çoklu kontrol girişi mevcuttur. Her bir hizmet sınıfı için çıkış ölçüldüğünden dolayı çoklu çıkış mevcuttur.



Şekil 2.14 : Farklı hizmeti destekleyen ön bellek sistemi için geri besleme kontrol sistemi.

2.2 Literatürdeki Diğer Araştırmalar

Kontrol kuramının bilgisayar sistemlerine uygulanmasını ele alan diğer araştırmaların bazıları Web sunucularda belli bir performansın sağlanması üzerinedir [20, 22, 24, 25, 27, 28, 17, 29, 31, 33]. Dolaylı veya mutlak anlamda, bağlantı gecikmesinin belirli bir değerden düşük olmasını garanti etmek üzere tasarlanan Web sunucu mimarisi, [20]'de sunulmuştur. Değişen iş yükü altındaki Web sunucusu için performans garantisi ve servis farklılığı konusu geri besleme kontrol kuramı bakış açısıyla [22]'de ele alınmıştır. [24]'de ise kontrol kuramı kullanılarak tasarlanmış bir Web sunucusunda performans garantisinin nasıl sağlandığı ile ilgili bilgi verilmektedir. Web sunucusundaki farklı hizmet sınıfları için, dolaylı anlamda bağlantı gecikmesinin belirli bir değerden düşük olma garantisi sağlayan uyarlamalı bir mimarinin tasarımı ve gerçekleştirilmesi [29] numaralı referansta verilmiştir. Bir internet sunucusunun matematik modellenin elde edilmesi ve geri besleme kontrol kuramı kullanarak performans kontrolüne ilişkin bir yöntem [28]'de yer verilmiştir. [25, 17]'de apache Web sunucusunun CPU ve bellek kullanım oranları 'MaxClients' ve 'KeepAliveTimeout' parametreleri ile kontrol edilmesine ilişkin bir yöntem geliştirilmiştir. Web vekil ön bellekleri (proxy cash memory) için kaynak yönetim mimarisi tasarım yöntemlerinden bazılarına [27, 31, 33] kaynaklarından ulaşılabilir.

Literatürde veri tabanı sistemlerinin geri besleme ile kontrolü üzerine bazı araştırma çalışmaları vardır, [41, 42, 43, 44, 45]. IBM DB2 veri tabanı sunucu olanaklarının yönetimi konusundaki bir öneri [41]'de sunulmuştur. Veri tabanı sistemlerinin geri besleme kontrolüne ilişkin diğer bir yaklaşım [42]'de verilmiştir. Değişen iş yükü altındaki gerçek zamanlı veri tabanı sistemlerinin beklenen servis kalitesini sağlaması için önerilen bazı yaklaşımlar [43, 44, 45]'te tartışılmıştır.

Lotus Notes e-posta sunucunun geri besleme kontrolüne ilişkin çalışma [23]'te açıklanmaktadır. Bu çalışmada herhangi bir anda mail e-posta sunucusuna bağlanmasına izin verilen kullanıcı sayısı kontrol işareti, gerçekleştirilmeyi bekleyen işlerin sayısı ise sistem çıkışı olarak düşünülmüştür. Lotus Notes mail sunucusunda hizmet seviyesi amaç ölçütünü yerine getirmede kullanılan klasik kontrol yaklaşımı [18]'de açıklanmıştır.

Bilgisayar sistemlerinin, kontrol kuramı kullanılarak tasarlanmış kontrolörler ile kontrol edilebilmesi için sistemin sağlaması gereken özellikler [46]'da açıklanmıştır. Kendi kendini yöneten sistemler oluşturmak için bir mimari önerisi ve kontrol teorisine dayalı analitik bir temel sağlayan bir yaklaşıma [47]'de yer verilmiştir. Bilgisayar sistemleri ile ilgili kontrol problemlerinin sınıflandırılması, önem sırasına göre sıralanması ve büyük ölçekli bilgisayar sistemleri için önerilen bazı kontrol mimarileri [48]'de sunulmuştur.

İş yükü dengeleme problemine çözüm getirmek üzere, lineer quadratik düzenleyici (LQR: Linear-Quadratic Regulator) yöntemi ile tasarlanmış bir kontrolör önerisi [26]'da sunulmuştur. İstenilen CPU kullanımını elde etmek üzere uçtan uca (end to end utilization: EUCON) kontrol algoritması [49]'da anlatılmaktadır. [50]'de, hizmet seviyesinin amaç ölçütü olduğu bir ağ sisteminde, bir ağ sunucusunun performansının geri besleme ile kontrolünde kullanılan kuyruk modelinden bahsedilmektedir. Ağ kontrol yapıları ve ağ gecikmeleri ile ilgili bazı çalışmalar [51, 52]'de bulunabilir. Gerçek zamanlı planlamada geri besleme kontrolünün uygulanması ve gerçek zamanlı planlama algoritmaları ile ilgili yaklaşımlar [53, 54, 55]'te sunulmuştur. İşletim sistemi modelleme ve kontrolü ile ilgili bir yaklaşım [56]'da önerilmiştir.

Bulut mimarisi ve veri merkezi üzerine gerçekleştirilen bazı araştırma çalışmaları da ilgili literatürde bulunmaktadır. Bu araştırmalar asıl olarak veri merkezindeki kaynak tahsis problemleri ile ilgilidir [57, 8, 58, 59, 60, 61, 9, 4]. Bu tez çalışması da temel olarak çoklu kaynakların veri tabanı sanal sunucusuna tahsisi üzerinedir. Aynı konuda [8]'de verilen yöntem, bu tez çalışmasında yapılandırılmış farklı olarak, zaman serisi analiz teknikleri kullanarak, beklenen iş yükü parametrelerinin, ölçülen sistem metriklerinden tahmin edilmesine ve sunucu kaynakları bu tahmin değerlerine göre tahsis edilmesine dayanmaktadır. Sanal ortamda kaynak arz ve talep konusunda geri besleme kontrol yapısı kullanmayan farklı bir yaklaşım [60]'da açıklanmaktadır.

Bu tez çalışması kapsamında kaynak tahsisi probleminin çözümü için izlenen yaklaşım ve geliştirilen yöntemlerin, [58, 59, 61]'de yer alanlardan önemli farklılıkları vardır. Literatürdeki bu çalışmalar, Apache Web sunucu için geliştirilen yöntemler iken, bu tez çalışmasında farklı mimaride sanal sunucu üzerindeki veri tabanı sunucusunun kaynaklarının kontrolü problemine çözüm getirilmiştir. Ayrıca önerilen yöntemler simülasyonlara ek olarak gerçek zamanlı yürütülen deneyler ile doğrulanmıştır. Literatürde var olan yaklaşımlardan diğer bir farklılık, kullanılan modelleme yaklaşımı açısındandır. Şöyle ki gerçek sisteme bir uyan bir emülatör sistem oluşturulmuş ve bu sistem üzerinde yapılan deneyler sonucu sistemin hem çok giriş-çok çıkışlı (MIMO) hem de etkileşimi olmayan iki ayrı tek giriş tek çıkış sistemden (SISO) oluşan modeller oluşturulmuş ve doğrulukları karşılaştırılıp irdelenmiştir.

Veri merkezindeki kullanımı (utilization) yüksek tutmak için kaynak paylaşımını dinamik olarak ayarlayan bir en uygun şekle sokma yöntemi ve çok katmanlı uygulama deneylerinin sonuçları, [57, 9, 4]'te yer almaktadır. Bilindiği gibi, çok katmanlı büyük ölçekli sistemlerde kaynak kullanımlarının en uygun şekle sokma algoritmaları, kaynak tahsislerin etkin olarak yapılabilirdiği varsayımı altında geliştirilir. Bu nedenle, bu tez çalışmasında, dinamik kaynak kontrolü problemi ele alınmıştır. Bu tez çalışmasında geliştirilen dinamik kaynak kontrolü sağlayan kontrolör tasarım yöntemlerinin, var olan bu en uygun şekle sokma algoritmalarının hayata geçirilmesinde ve yaygın olarak kullanılmasında çok önemli olduğunu düşünülmektedir. Geliştirilen iki yöntem de yapılan zorlama -aşırı iş yükü altında- deneylerinde başarılı sonuçlar vermiştir [62].

2.3 Kontrol Teorisinin Bilgisayar Sistemlerine Uygulanmasında Karşılaşılan Problemler

Nadiren de olsa kontrol kuramının, bilgi işlem sistemlerinin bazı problemlerinin çözümünde kullanılmasına rastlanır. Kontrol kuramının, bilgisayar mühendisleri tarafından yaygın bilinen bir konu olmaması nedeniyle, geri beslemeli kontrol yöntemleriyle kolaylıkla çözülebilecek birçok problem farklı yollardan etkin olmayan yöntemlerle yapılagelmiştir. Kontrol kuramının, bilgisayar sistemlerine uygulanmasında belli başlı, önemli çözülmesi gereken sorun vardır [11, 1, 40].

1-) Amaç ölçütlerinin belirlenmesi,

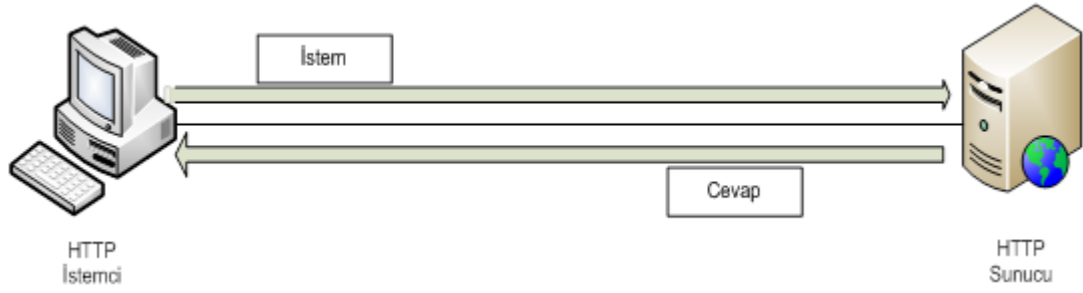
- 2-) Kontrol edilecek bilişim süreçlerinin matematik modelinin elde edilmesi,
- 3-) Uygun tasarım yönteminin seçilmesi.

Bu bölümde, kontrol teorisinin bilgisayar sistemlerine uygulanması konusunda literatürde var olan çalışmalarla ilgili bilgilere ve kontrol kuramının bilgisayar sistemlerine uygulanmasında karşılaşılan problemlere yer verilmiştir. Bilişim süreçlerinin kontrol edilmesinde, geri beslemeli bir sistemin kurulması ve çalıştırılması konusunda deneyim elde etmek amacıyla yapılan bir ön çalışma bir sonraki bölümde sunulmuştur.

2.4 Çevrimiçi Ödeme Sisteminde Tıkanıklığı Fark Eden Geri Beslemeli Bir Kontrol ve Uyarıcı Sistem

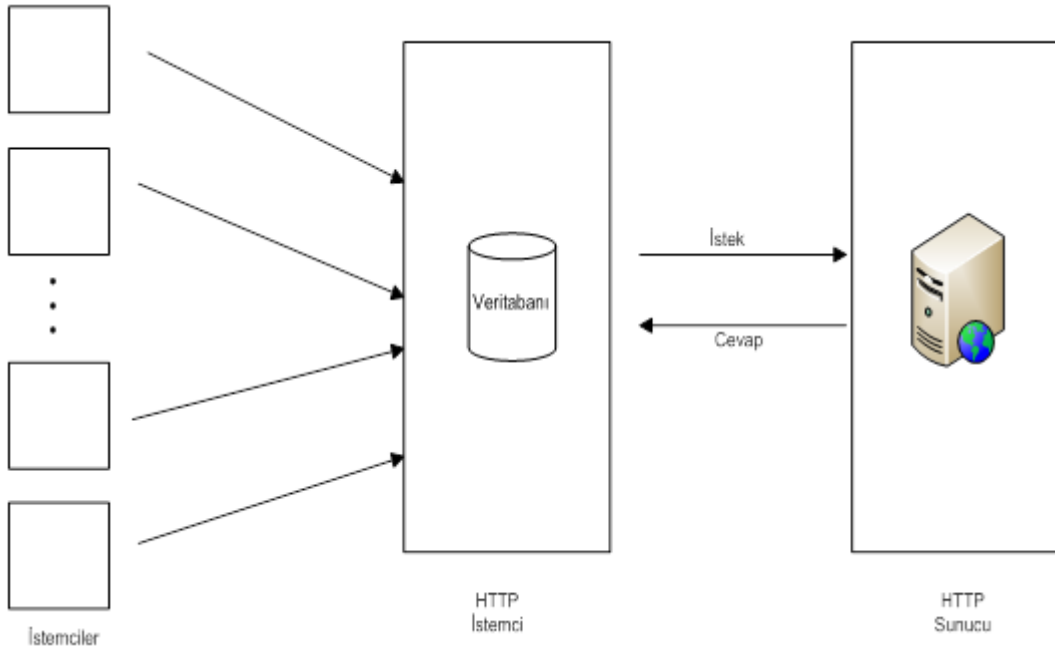
2.4.1 Ödeme sistemi istemci sunucu haberleşmesi

Üzerinde çalıştığımız ödeme sisteminde istemci - sunucu arasındaki iletişimde, istemci istekte bulunurken sunucu bu isteklere yanıt verir. Bu iletişim TCP, HTTP gibi protokoller ile gerçekleşir. Protokoller, karşılıklı iletişimde standartları ve iletişim kurallarını belirler. İstemci sunucu arasındaki iletişimde istekler bazı bilgilerin belli bir yapı ve formatta sunucuya gönderilmesi ile olur. İletişimde pek çok farklı veri formatı kullanılabilir. Tasarlanan ödeme sisteminde bu formatlardan en yaygın olanlarından biri Extensible Markup Language (XML) veri formatı kullanılmıştır. XML veri belli bir yapısı olan veridir. İstemci sunucunun göndereceği cevabın yapısını ve sunucunun hangi yapıda bilgi beklediğini bilir. Tasarlanan çevrimiçi fatura ödeme sisteminde istemci sunucunun beklediği yapıda bilgiyi sunucuya göndererek yanıt bekler. İstemci sunucu arasındaki çevrimiçi iletişim Şekil 2.15'te gösterilmektedir.



Şekil 2.15 : İstemci sunucu haberleşmesi.

Üzerinde çalışılan çevrimiçi fatura ödeme sistemi bir Web servis uygulamasıdır ve sunucu durumunda olan Web servis uygulamasına istemci tarafından farklı isteklerde bulunulabilir. Çevrimiçi ödeme sisteminde sunucunun fatura bilgisi sorgulama, ödeme, ödeme iptal, talimat, talimat iptal, mutabakat gibi metotları çağırılarak istemci tarafından gerçekleştirilen sorgulamalara sunucu tarafından çevrimiçi olarak yanıt dönülür. Bu iletişim Şekil 2.16’da gösterilmektedir.



Şekil 2.16 : Çevrimiçi ödeme sistemi istemci-sunucu arasındaki haberleşme genel şeması.

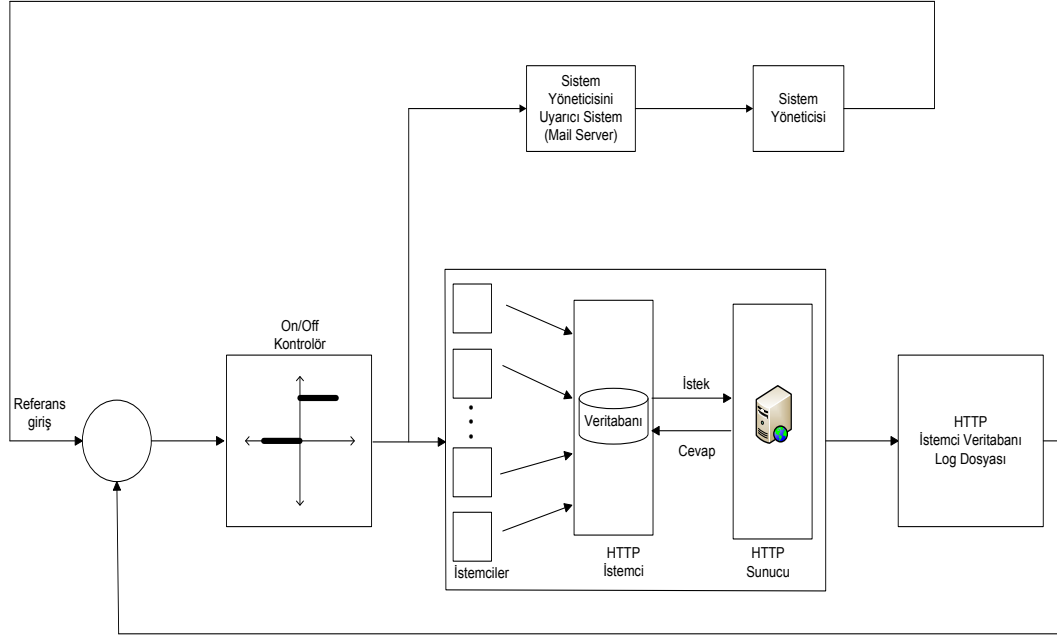
2.4.2 Çevrimiçi ödeme sisteminde bir kontrol probleminin tanımlanması

Çevrimiçi ödeme sisteminde istemci ile sunucu arasındaki bağlantıda işlemlerin sıklığı sebebiyle zaman zaman yoğunluk yaşanabilmektedir. Hatta bazı durumlarda yoğunluk sebebiyle sunucu istemcinin gönderdiği sorgulamalara yanıt veremez ve istemci tarafında işlemler *time-out*’a düşünceye kadar sunucudan yanıt beklemeye

devam eder. İstemciden sunucuya gönderilen her bir sorgulama, istemci sisteminde sunucuya bağlantı kuran bir *bağlantı (connection)* açar ve işletim sistemi üzerinde bir *süreç* olarak yer alır. Sunucudan yanıt gelmediği durumda, bu süreçler işletim sistemi üzerinde timeout olana kadar bekletilir, bu nedenle de bu süreçlerin sayısı hızla artabilir. Sunucu sorgulamalara yanıt veremediği durumda istemci–sunucu ödeme sisteminde istemci tarafındaki süreçlerdeki büyük artış sebebiyle istemci tarafında sistemin çalışması durma noktasına gelebilmektedir. Böyle bir durumda süreçlerin daha fazla artmasını önleyerek sistemdeki yavaşlama sorununu giderecek bir çözüme ihtiyaç duyulmaktadır. Diğer bir deyişle, istemci-sunucu arasındaki ödeme sisteminde, istemci tarafında yaşanabilecek tıkanıklığı çok geç olmadan fark eden, ilgili birimleri ve sistem sorumlularını uyaran ve tıkanıklığın yaşanmasını engelleyebilecek bir geri besleme kontrol sistemine ihtiyaç duyulmaktadır. Tasarlanacak olan geri besleme kontrolör vasıtasıyla istemci tarafındaki sistemin yavaşlamasına sebep olan durumun ortadan kaldırılabileceği öngörülmüş ve bir çözüm geliştirilmiştir.

2.4.3 Çevrimiçi ödeme sistemi için geri beslemeli bir sistem ve aç-kapa kontrolör tasarımı

İstemci-sunucu arasındaki ödeme sisteminde problem olduğu zaman istemci tarafında çalışma yöntemini çevrimdışı getirecek ve problemin ortadan kalktığını fark edip sistemi otomatik olarak tekrar çevrimiçi çalışma yöntemine alacak ve bu geçişleri yaparken sistem yöneticilerine gerekli uyarıları gönderecek bir geri besleme kontrol sistemi Şekil 2.17’de gösterildiği gibi tasarlanabilir.



Şekil 2.17 : Ödeme sistemi için geri besleme kontrol sistemi.

Şekil 2.17’deki geri besleme kontrol sisteminde, kontrolör sistemi çevrimiçi ya da çevrimdışı yöntemiyle çalışmasını sağlayacak işareti üreten bir sistem olarak tasarlanacaktır. Yani, kontrolörün ürettiği kontrol işareti bir anahtar görevi görecektir. Sunucuya gönderilen her bir XML formatındaki mesaja karşılık sunucudan yine XML formatında veri içeren mesaj dönülmektedir. Karşılıklı mesajlaşma sırasında sunucuya gönderilen her bir mesaj ve cevabı istemci tarafında ilgili log tablosunda saklanmaktadır. Bu tabloda işlem tarihi, işlem zamanı, sunucuya gönderilen mesaj, sunucudan gelen XML formatındaki mesaj gibi bilgiler tutulur. Log tablosunun yapısı Çizelge 2.1’de gösterilmektedir.

Çizelge 2.1 : Veri tabanında tutulan log tablosu yapısı.

İşlem tarihi	İşlem zamanı	Giden Mesaj	Gelen Mesaj
02.06.2009	10:52:23186	<fatsorgu>...</fatsorgu>	<sorgucevap>...</sorgucevap>
02/06/2009	10:54:87543	<ödeme>...</ödeme>	<ödemecevap>...</ödemecevap>

Log tablosundaki bilgiden yola çıkarak istemci-sunucu arasındaki işlemlerin ne kadarının başarılı sonuçlandığı bilgisi yüzdesel olarak elde edilebilir. Log tablosundaki bilgi, kontrolörün dakika dakika yapılan işlemlere dair istatistik hazırlaması bakımından da önemlidir. Kontrolör log tablosunda tutulan bilgiden yola çıkarak gün boyunca yapılan işlemlerle alakalı bir istatistiksel bilgi üretir. Bu istatistiksel bilgi dakika bazında toplam işlem adedi, başarılı işlem adedi, hatalı işlem

adedi, başarı yüzdesi, hata yüzdesini ifade eder. Çizelge 2.2’de saat 09:25 ile 09:46 zaman dilimi içerisinde yapılan işlemlerle alakalı olarak her bir dakikada gerçekleşen toplam işlem adedi, başarılı işlem adedi, hatalı işlem adedi, başarı yüzdesi, hata yüzdesi gösterilmektedir.

Çizelge 2.2 : Kontrolörün log tablosundaki bilgileri kullanarak dakika bazında ürettiği istatistiksel bilgi.

Row #	SAAT	DAKİKA	TOPLAMİSLEMADET	HATALIADET	BASARILIADET	HATA_YÜZDESİ
82	09	25	30	0	30	0%
83	09	26	45	0	45	0%
84	09	27	26	0	26	0%
85	09	28	50	0	50	0%
86	09	29	28	0	28	0%
87	09	30	35	0	35	0%
88	09	31	28	0	28	0%
89	09	32	38	37	1	97%
90	09	33	72	72	0	100%
91	09	34	14	14	0	100%
92	09	35	10	10	0	100%
93	09	36	10	10	0	100%
94	09	37	22	22	0	100%
95	09	38	47	47	0	100%
96	09	39	15	15	0	100%
97	09	40	10	10	0	100%
98	09	41	14	14	0	100%
99	09	42	21	21	0	100%
100	09	43	40	40	0	100%
101	09	44	11	11	0	100%
102	09	45	7	7	0	100%
103	09	46	12	12	0	100%

Sistem yöneticisi tarafından belirlenen oran ile log tablosunda saklanan bilgilerden hesaplanan oran karşılaştırılarak bir kontrol işareti üretilir. Kontrol işareti iki değer olacaktır. 1 değeri çevrimiçi çalışmayı sağlayacak, 0 değeri ise sistemi offline çalışma yöntemine geçirecektir. Kontrolör elemanı bir yazılımdır ve bu yazılım içinde bulunan dakikadan bir önceki dakikada yapılan tüm işlemleri göz önüne alarak işlemlerin başarılı gerçekleşme oranını -yüzde olarak- hesaplar. Eğer kontrolörün hesapladığı bu oran, sistem yöneticisinin belirlediği başarı oranının –referans değer-altındaysa sistemi offline çalışmaya geçiren işaret üretilir, eğer başarı oranına eşit veya büyükse çevrimiçi çalışma devam eder. Sistem offline çalışmaya alındığı zaman, dakikada bir sunucuya çevrimiçi olarak sorgulama mesajı gönderilir, yanıt alınırsa sistem tekrar çevrimiçi çalışma yöntemine getirilir. Kontrol işareti aynı zamanda e-posta sunucu için de bir giriş işaretidir. E-posta sunucusu, sistemin çalışma yöntemindeki değişiklikleri, işlemlerin başarı yüzdesi, belirlenen en son referans giriş bilgilerini e-posta yoluyla sistem yöneticisine raporlar. Sistem

yöneticisi bu sonuçları değerlendirir, gerekli gördüğü takdirde referans giriş parametresini bir ara yüz vasıtasıyla değiştirerek geri besleme kontrol sisteminin çalışmasına gerekli müdahalede bulunabilir. Referans işaretin belirlenmesinde sistem yöneticisinin tecrübesi önemlidir.

2.4.4 Çevrimiçi sisteme ilişkin sonuçlar

Kontrol edilen süreç sayısal bir sistem olduğu ve kontrolörün belirli zaman aralıklarında bir dizi hesaplamaları yapıp bir kontrol işareti üretmesi gerektiği için, geri beslemeli kontrol sistemi de ayrık zamanlı bir sistem olarak tasarlanmıştır.

Tasarımın başlangıcında önemli bir sorun, hesaplamaların hangi zaman aralıklarıyla yapılacağını belirlenmesi sorunudur, yani kontrol kuramı açısından örnekleme zamanının seçimi problemidir. Bu çalışmada, örnekleme zamanı bir dakika seçilerek sistem çalıştırılmıştır, yani kontrolör dakikada bir kontrol işareti üretecek şekilde tasarlanmıştır. Bu sistem için örnekleme zamanı, kontrolörün bir karar vermesi için yapacağı hesaplar için gerekli olan ortalama süre göz önüne alınarak seçilmiştir. Çünkü kontrolörde hesaplama gecikmeleri olursa bu durum sistemin işleyişini olumsuz etkileyebilir. Örneğin, örnekleme zamanı bir dakika seçildiyse ve kontrolörün sistemdeki log tablosundan yola çıkarak yapacağı hesaplamalar bir dakikayı geçiyorsa ve sistemde pek çok işlem hata almışsa, kontrolör sistemi çevrimdışına geçirmede geç kalmış olur bu da tasarlanan sistemin performansını olumsuz yönde etkiler. Diğer taraftan, örnekleme zamanını gereğinden büyük seçilmesi durumunda, gecikmeden dolayı sistem çıkışında istenmeyen sonuçlar örneğin sistemde yavaşlamalar veya takılmalar yaşanabilir. Bu ön çalışmada tasarlanan kontrolör aç-kapa kontrolördür ve istemci-sunucu arasındaki ödeme sistemi için işe yaradığı gözlemlenmiştir.

3. SİSTEM TANIMA, MODELLEME VE İNTERNET TRAFİK VERİSİNİN MODELLENMESİ ÖN ÇALIŞMASI

Tez çalışması kapsamında, sunucular, kuyruk yapıları, bellek, ağ gibi bilgisayar sistemi ya da karmaşık bir bilgi işlem sisteminin alt sistemlerinin matematiksel modelinin türetilmesi için kullanılabilecek yöntemleri incelemek ve bilgi sahibi olmak amacıyla, sistem kontrol literatüründe var olan sistem tanıma ve modelleme yöntemleri üzerine araştırma yapılmıştır. Bu amaçla bir ön tasarım deneyimi olması için, MATLAB'ın sistem tanıma konusundaki gelişmiş ara yüzünden ve işlem kabiliyetinden yararlanmak suretiyle farklı tanıma yöntemleri Berkeley Laboratuvarı ile internet arasındaki bir saat süre boyunca kaydedilen internet veri trafiğinin matematik modelinin elde edilmesi problemine uygulanmış ve elde edilen sonuçlar tartışılmıştır.

Kontrol teorisi ve teknikleri kuyruk sistemleri, Web sunucular, mail sunucular, yönlendiriciler, bilgisayar ağları, bellek yönetimi gibi bilgisayar sistemlerini ilgilendiren farklı konularda uygulanmıştır. Bilgisayar süreçlerini ilgilendiren bir problemin çözümü için kontrol kuramı kullanılarak bir çözüm önerilmesi için sürecin bir matematik modeli oluşturulmalıdır. Matematik modelin elde edilmesi için sistem tanıma teknikleri kullanılabilir. Ancak model belirlendikten sonra bu model temel alınarak ve çeşitli kontrol teknikleri kullanarak sistemi kontrol edecek bir kontrolör tasarımı yapılabilir. Model elde etmek için sistemde ölçülebilen giriş ve çıkış büyüklükleri kullanılmalıdır. Burada önemli ve zahmetli olan süreç modellenecek sisteme ilişkin doğru veriyi toplayabilmektir. Veri toplandıktan sonra sisteme ilişkin tahmini bir model oluşturmak üzere sistem tanıma metotları kullanılır. Matematik model doğrudan fizik kanunlarını kullanarak da elde edilebilir, bu durumda da türetilen model bilinmeyen parametreler içerir ve bunlar yine sistem tanıma teknikleri kullanılarak belirlenirler. Eğer sistemin matematik modeline ilişkin hiçbir ön bilgi yoksa kara-kutu (black-box), bir takım bilgiler elimizde mevcutsa gri-kutu (grey-box) tanıma yöntemleriyle model tahmini yapılır [10]. Sistem tanıma deneyine

başlamadan önce, bazı ön deneyler ve elde edilen verilerin analizi ile sistemin zamanda değişmez olup olmadığı, lineerliği, geçici hal davranışı, frekans cevabı incelenerek sistem ile ilgili bir takım bilgiler elde edilebilir. Sonraki aşamada sistemi tanımak için gerekli giriş işaretlerinin seçimi yapılır ve toplanan veriler; tanıma verisi ve doğrulama verisi olarak parçalara ayrılır. Verideki aykırılıklar (outliers), bozucu etkiler (aliasing effects) ve sapmalar (trends) veriden ayıklanır-süzülür-. Filtreleme sayesinde sistem tanıma işlemi, sistemin istenen frekans bandında modelinin elde edilmesi için yapılabilir ve ayrıca gürültü etkisinin tanıma işlemine olumsuz etkisi engellenebilir. Daha sonra, bir model yapısı seçilir ve modele ilişkin parametreler belirlenir. Eğer mümkün olabiliyorsa modelde indirgeme yapılır. En son olarak modelin doğruluğu test edilir. Verinin doğrulama için ayrılan kısmıyla çapraz doğrulama yapılır. Kutup-sıfır yerleşimi, bode diyagramları, simülasyon ve çapraz doğrulama ile modelin gerçek sistemi ne kadar yansıttığı test edilebilir. Yukarıda bahsi geçen sebeplerden dolayı bu tez çalışmasında ele alınan sistemin modellenmesinde sistem tanıma tekniklerinin kullanılması gerektiği düşünülmüştür. Bir sonraki bölümde sistem tanıma ve modelleme konularına değinilecek ve mevcut yöntemler irdelenecektir. Daha sonra internet trafik verisinin modellenmesi ile ilgili yapılan çalışma sunulacaktır.

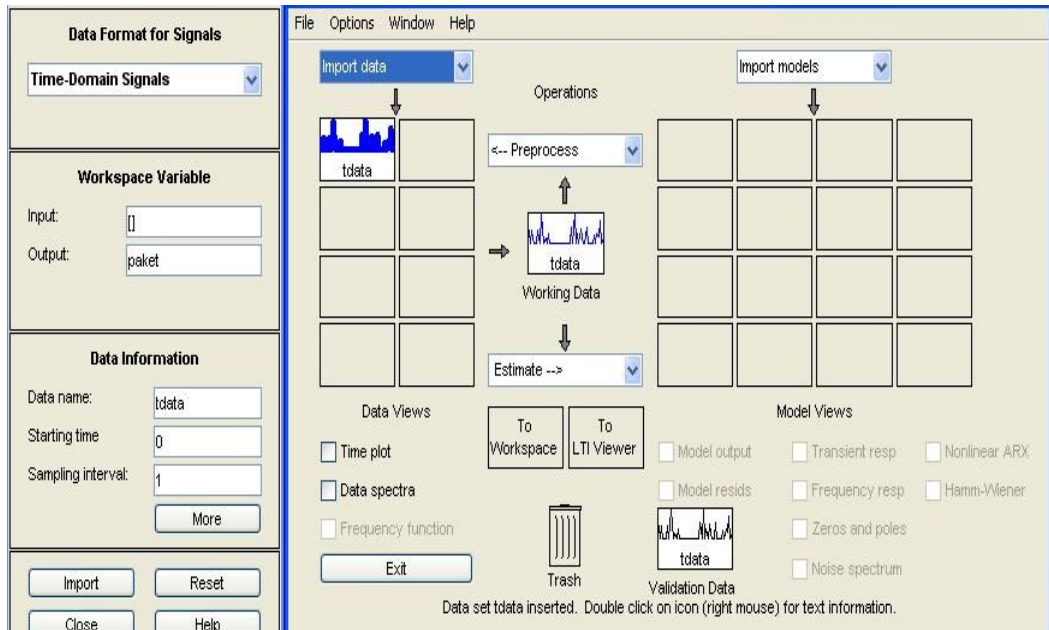
3.1 Sistem Tanıma ve Modelleme Yöntemlerinin İnternet Veri Trafikinin Modellenmesinde Kullanılması

Bu bölümde sistem tanıma yöntemleri ve bu yöntemlerin Lawrence Berkeley Laboratuvarı ile dış dünya-internet arasındaki veri trafiğine uygulanmasına yer verilmiştir. Veri trafiğinin modellenmesi konusunda literatürde var olan yöntemler [63, 64, 65, 66, 67] ve ağdaki veri trafiğinin gözlemlenmesinde ve kontrolünde kullanılabilecek bazı uygulamalar incelenmiştir [68, 69]. Bilişim süreçlerinin modellenmesi konusunda deneyim edinmek amacıyla, [2, 3] numaralı referansta verilen ve internet veri trafiğine ilişkin anlık paket sayıları bilgisini içeren veri kullanılarak sürecin matematik modelinin elde edilmesi sorunu ele alınmıştır. Bu veri, belli bir zaman aralığında ağda bulunan paket sayılarını göstermektedir. Veri paketlerinin aktarımı sırasında ağdaki anlık paket sayıları ölçülmüş ve bu veriden yola çıkarak ağdaki trafik modellenmeye çalışılmıştır. Bunun için farklı model yapıları seçilmiş, model tahminlerinde bulunulmuştur, çıkan sonuçlar tartışılmıştır.

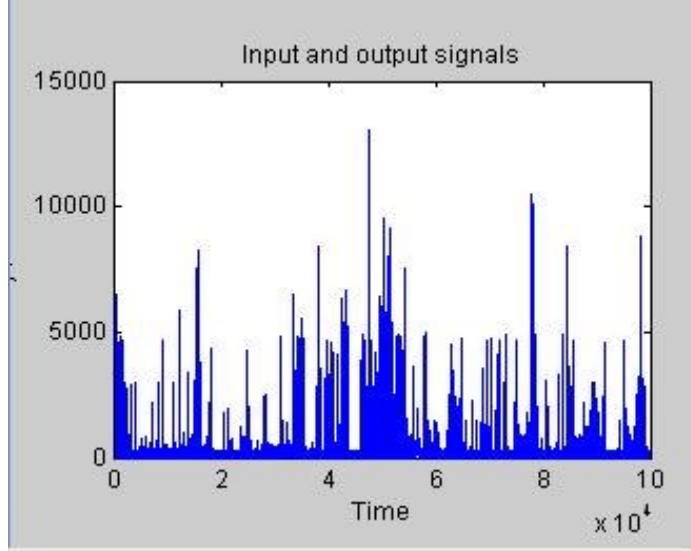
3.1.1 İnternet trafik verisinin MATLAB ortamına aktarılması ve verinin gösterilmesi

Veri iki kolondan oluşmaktadır. İlk kolon zamanı gösterir, diğer kolon ise belli bir zamandaki veri paket sayısını gösterir. Ağdaki trafiğin modelini elde etmek üzere MATLAB'ın ara yüzü ve sistem tanıma paketindeki fonksiyonlar kullanılmıştır. Model tahmini için, önce, söz konusu veri MATLAB çalışma ortamına aktarılmış ve bu veriye 'dataset_tez ismi' verilmiştir. Kullanılan veri sisteme bir giriş uygulanmadan doğrudan çıkıştan ölçülen bir veri olduğu için, zaman serisi -time series- olarak modellenenecektir.

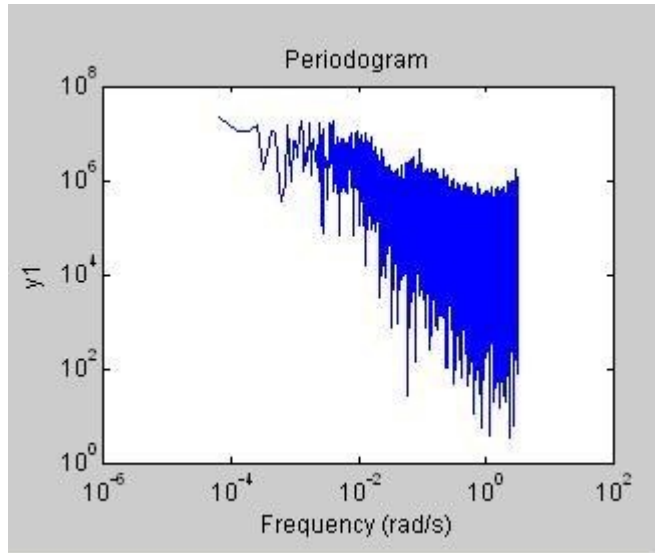
$Paket = dataset_tez(:,2)$ MATLAB komutu ile paket bilgileri paket isminde vektöre taşındı. *Paket* vektörü sistem çıkış bilgisini içermektedir. *Ident* komutu kullanılarak sistem tanıma ara yüzü açıldı. Model tahmini için aşağıdaki adımlar izlendi. İlk olarak *Paket* bilgisi MATLAB SIT'a aktarıldı. Aktarılan veri *tdata* ismi ile isimlendirildi. MATLAB ortamına aktarılan veri ile ilgili grafikler Şekil 3.1, Şekil 3.2 ve Şekil 3.3 ile gösterilmiştir.



Şekil 3.1 : Verinin MATLAB çalışma ortamına aktarılması.



Şekil 3.2 : Tdata ile isimlendirilen veride, çıkışı ifade eden paket sayısının zamana göre değişimi.



Şekil 3.3 : Tdata ile isimlendirilen çıkış verisinin periodogram üzerinde veri spektrumu.

3.1.2 İnternet trafik verisi modelleme öncesi işlem adımları

İnternet trafik verisi model yapısı seçimi ve model tahmininden önce bazı işlem adımlarından geçer, bu adımlar aşağıda açıklanmaktadır.

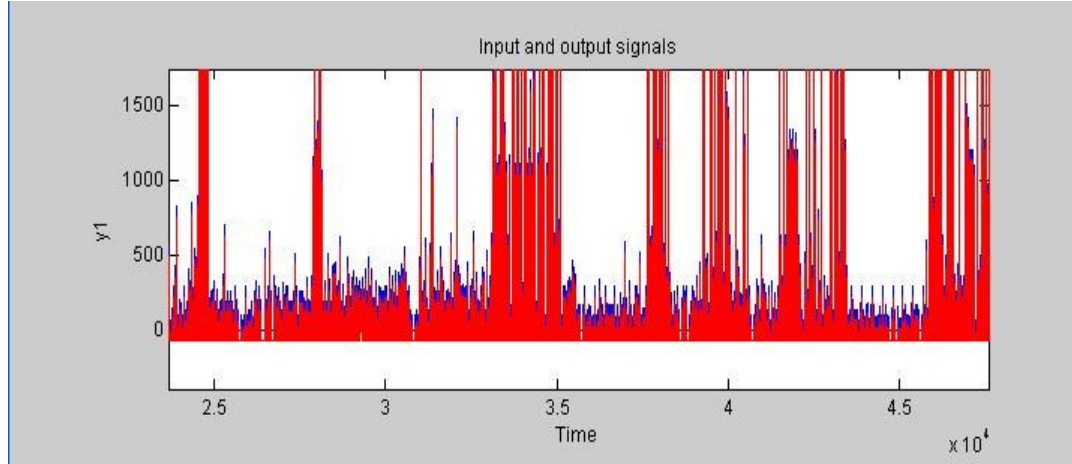
3.1.2.1 Verideki bozuk kısımların atılması

Tdata MATLAB ortamına aktarıldıktan sonra verideki bozukluklar veya aykırılıklar *detrend* MATLAB komutu ile yok edilir.

```
tdatade = detrend(tdata, 'constant');
```

```
tdatade = detrend(tdata);
```

Datadaki bozukluklar ve aykırılıkların ortadan kaldırılması ile ilgili olarak Şekil 3.4 bir fikir vermektedir.



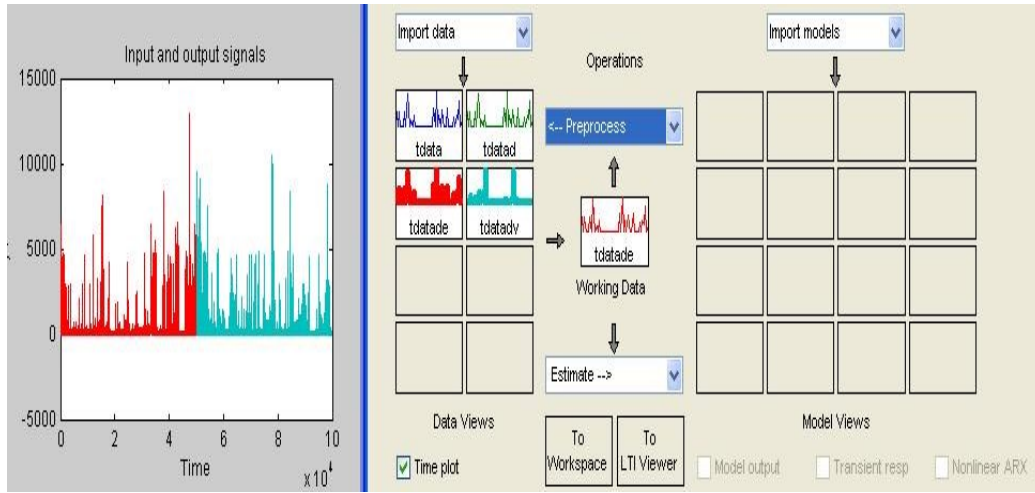
Şekil 3.4 : Mavi renkteki grafik tdata'yı gösterir, kırmızı renkteki bilgi bozuk veriden temizlenmiş veri tdatade 'yi ifade eder.

3.1.2.2 İnternet trafik verisinin çalışma ve doğrulama verisi olarak iki parçaya ayrılması

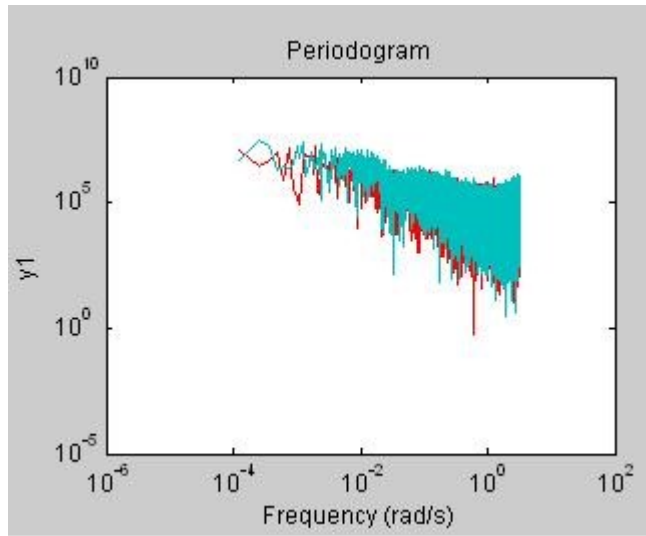
Veri, bir kısmı model tahmininde kullanmak üzere, diğer kısmı tahmin edilen modelin doğruluğunu test etmek üzere iki parçaya ayrıldı. Tdatade çalışma verisi, tdatadv ise doğrulama verisi olarak isimlendirildi. 99999 adet bilgi içeren tüm verinin ilk 50000'lik kısmı çalışma verisi olarak seçildi, geri kalan 49999 adet bilgi içeren kısmı ise doğrulama verisi olarak seçildi. Çalışma ve doğrulama verisine ilişkin bilgiler Şekil 3.5 ve Şekil 3.6 ile gösterilmiştir.

```
tdatade = iddata(tdata(1:50000,1));
```

```
tdatadv = iddata(tdata(50000+1:99999,1));
```



Şekil 3.5 : Kırmızı renkteki tdatade çalışma verisi, mavi renkteki tdatdv doğrulama verisidir.



Şekil 3.6 : Kırmızı renkteki tdatade çalışma verisi için veri spektrumunu, mavi renkteki tdatdv doğrulama verisi için veri spektrumunu gösterir.

3.1.3 Model yapısının seçimi

Model yapısının belirlenmesinde parametrik yöntemler ve parametrik olmayan yöntemler mevcuttur, aşağıda bazı parametrik yöntemlere değinilmiştir [10, 1].

3.1.3.1 AR model yapısı ile modelleme

AR model yapısı için model tahmininde kullanılan belli başlı iki metot mevcuttur. Bu metotlar ARX ve IV yapay değişken metotlarıdır. Bu metotlardan ARX metodu kullanıldığı zaman elde edilen model parametreleri ve bazı sonuçlar aşağıda belirtilmiştir. Veri time-series tipte veri olduğundan, giriş işareti bulunmaz. AR model yapısı, bu tip veri için uygundur. Bu yapıda model aşağıdaki formülle ifade

edilebilir. AR Model : $A(q)y(t) = e(t)$, u giriş işareti modelde bulunmaz. Şekil 3.7 ile model parametrelerinin seçimi gösterilmektedir.

Şekil 3.7 : AR model yapısının sisteme uygulanması için parametre girişi yapılır. Sistem mertebesi dört olarak seçilmiştir.

Aşağıdaki komutlar MATLAB’da çalıştırılarak model elde edilir.

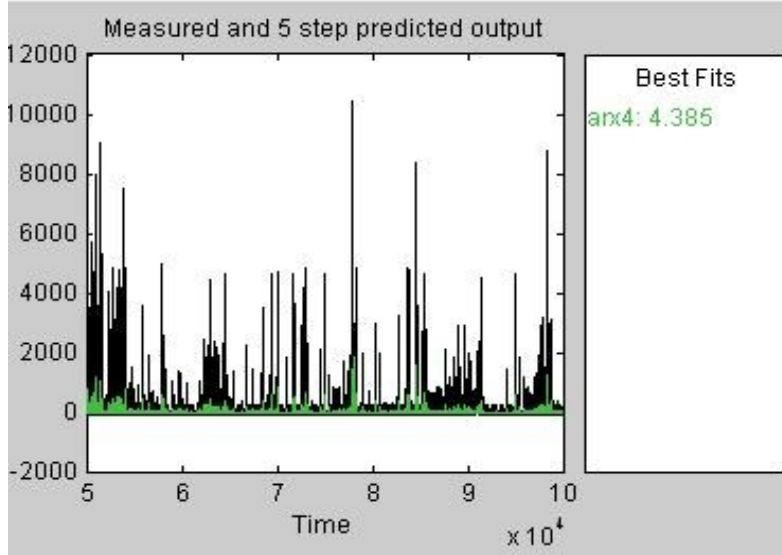
$tdatad = dtrend(tdata,0)$

$tdatade = tdatad([1:50000])$

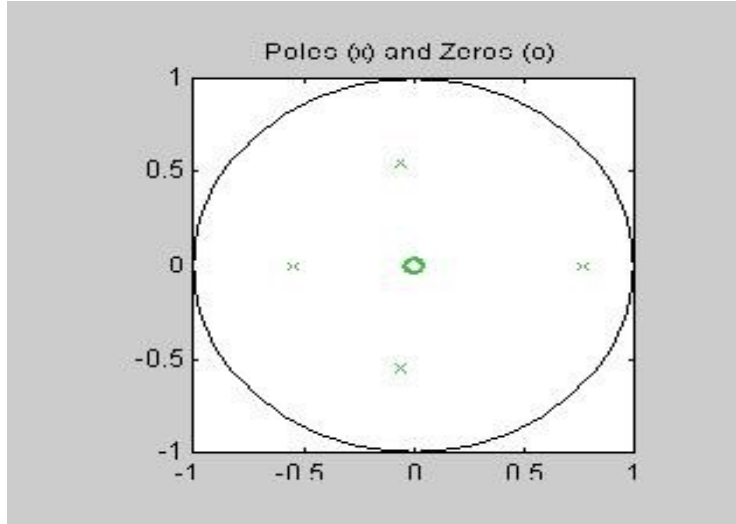
$arx4 = arx(tdatade,[4])$

Modelin tahmini sonunda ayrık zamanda AR Model $A(q)y(t) = e(t)$ yapısı için $A(q)$, (3.1) denklemini olarak elde edilmiştir. Modele ilişkin diğer bilgiler Şekil 3.8 ve Şekil 3.9 ile gösterilmiştir.

$$A(q) = 1 - 0,1171 q^{-1} - 0,1522 q^{-2} - 0,1128 q^{-3} - 0,1289 q^{-4} \quad (3.1)$$



Şekil 3.8 : AR model ile elde edilen sistem çıkışı ile ölçülen gerçek çıkışın gösterimi, siyah renk gerçek çıkışı, yeşil renk modellenen sistem çıkışını ifade eder.



Şekil 3.9 : Modellenen sistem için kutup-sıfır dağılımı.

3.1.3.2 Durum uzayı model yapısı ile modelleme

Durum uzayı modelinin tahmin edilmesinde iki farklı metot kullanılabilir. Bunlardan ilki n4sid yöntemi, diğeri ise pem yöntemidir. Sistem, diğeri bir parametrik model olan durum uzayı model yapısı kullanılarak n4sid metot ile modellendi ve sistem mertebesi bir olarak seçildi. Bunun için aşağıdaki MATLAB komutları çalıştırıldı. Bu metotlardan n4sid metodu kullanıldığı zaman elde edilen model parametreleri ve bazı sonuçlar aşağıdaki Şekil 3.10, Şekil 3.11, Şekil 3.12 ve Şekil 3.13 ile gösterilmiştir.

$$tdata = dtrend(tdata, 0)$$

$tdatade = tdatad([1:50000])$

$n4s1 = n4sid(tdatade, 1, 'N4Horizon', [15 \ 12 \ 12])$

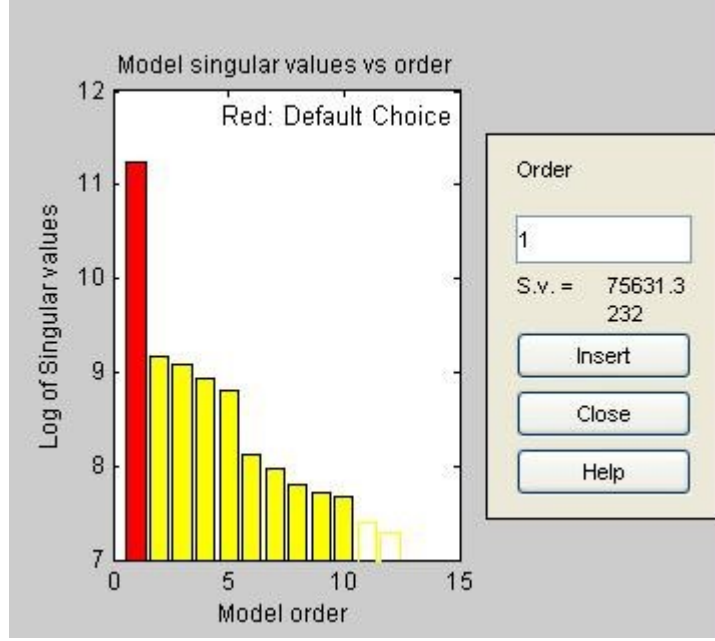
$$x(t + Ts) = Ax(t) + K e(t)$$

$$y(t) = Cx(t) + e(t) \quad (3.2)$$

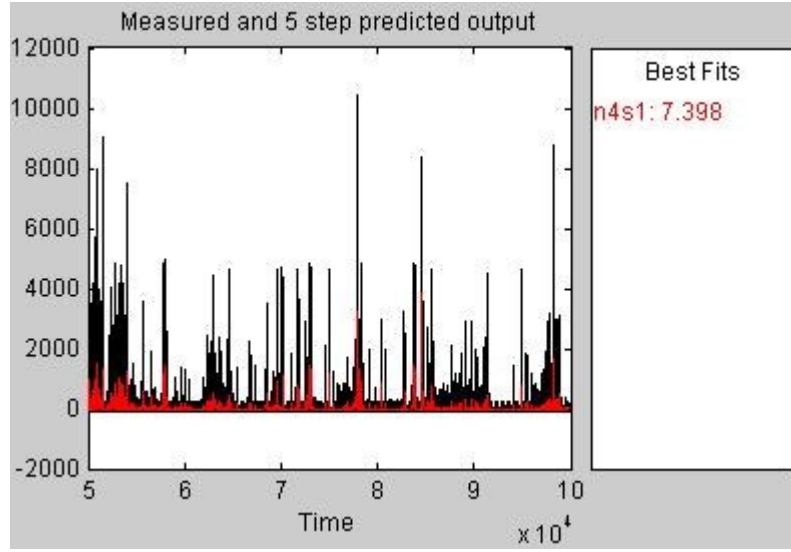
$$A = 0,9568, C = 46724, K = 2,4702e-006, x(0) = 0,00013906 \quad (3.3)$$

(3.2) sistemine ilişkin modelin parametre değerleri (3.3) denklemleri ile gösterilebilir.

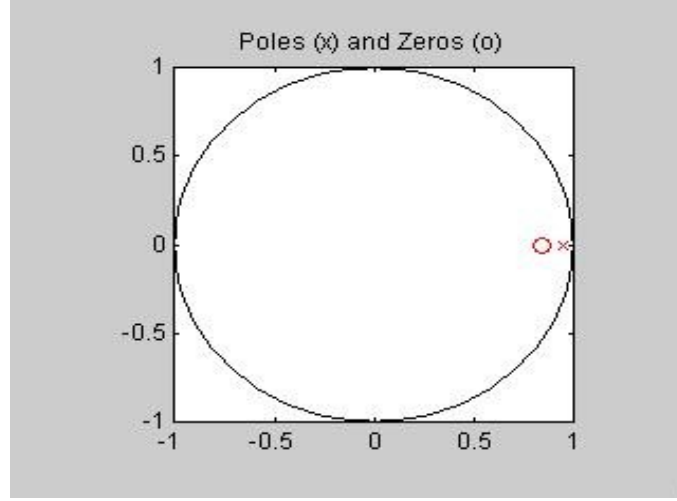
Şekil 3.10 : Durum uzayı model yapısı seçildi. Model derecesini 1-10 arası uygun bir değer olarak seçmek üzere 1:10 seçimi yapıldı.



Şekil 3.11 : Bir durum değişkeni modelleme için yeterli görüldü, diğer durumlar dikkate alınmadı. Sistem derecesi 1 olarak belirlendi.



Şekil 3.12 : Durum uzayı model yapısı kullanılarak ve n4sid metodu ile elde edilen modelin çıkışı kırmızı ile ölçülen sistem çıkışı siyah ile gösterilmektedir.



Şekil 3.13 : Durum uzayı model yapısı kullanılarak n4sid metodu ile elde edilen sisteme ilişkin sıfır-kutup dağılımı.

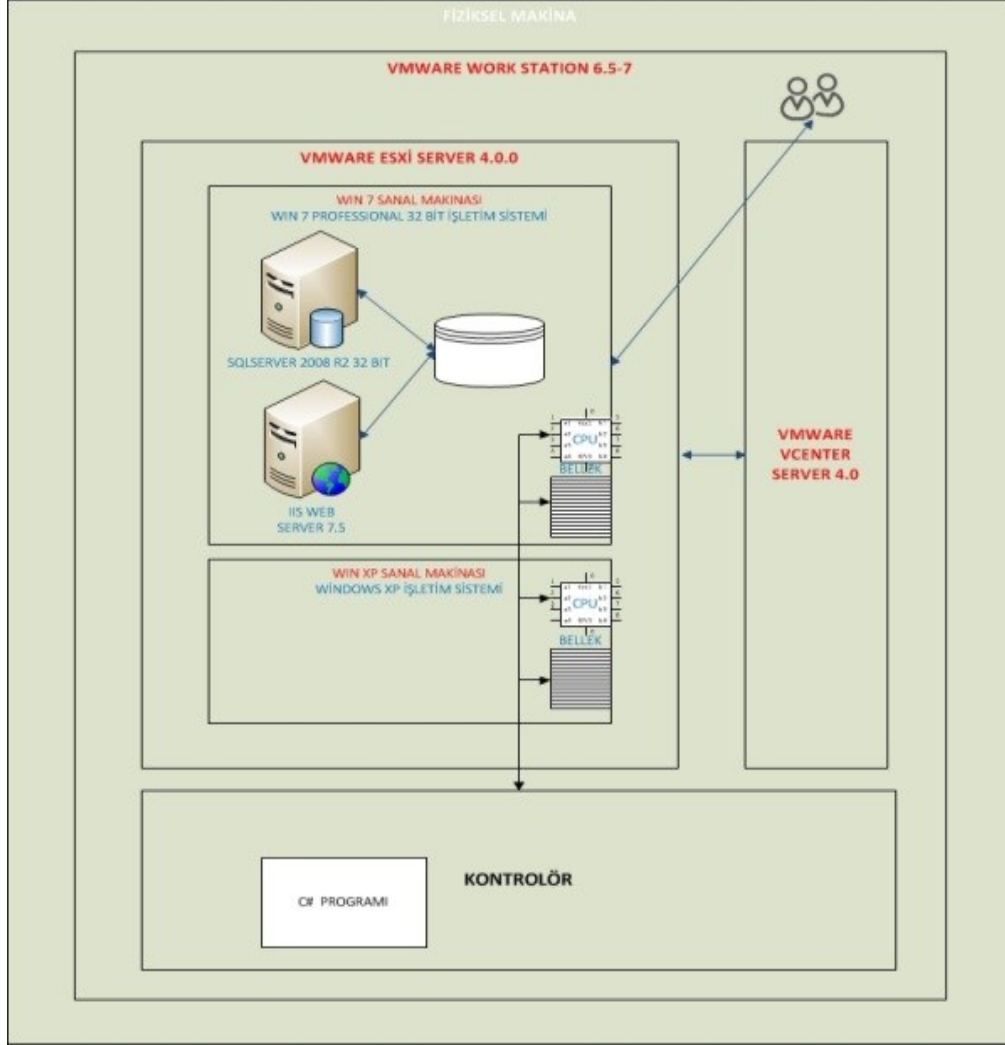
3.1.4 Sonuçlar

İnternet ile Berkeley Laboratuvarı arasındaki paket aktarımı verisi için, parametrik model yapıları ile her model yapısına ilişkin farklı yöntemler kullanarak modeller oluşturuldu. Bu model yapıları ve ilgili metotların bazıları ile ilgili çalışma ve sonuçlar bu bölümde sunuldu. Model seçimi öncesinde veri ile ilgili yapılması gereken işlemlerin içeriği anlaşıldı ve bizzat bu işlemler veriye uygulandı. Oluşturulan modellere ait parametreler hesaplandı. Time series veri, giriş bilgisi içermediğinden durum uzayı modeli ve AR modellere ilişkin farklı yöntemler kullanılarak modelleme yapıldı. Durum uzayı modeli ve AR model kullanılarak elde edilen sonuçlar grafik ile gösterildi. Model seçiminde dikkat edilmesi gereken noktalar tespit edildi. Model yapıları arasındaki farklılıklar anlaşıldı. Elde edilen modelin doğrulanması veya sağlamasının yapılmasına ilişkin yöntemler incelendi. Bu yöntemler veriye uygulandı. Böylelikle tez çalışması kapsamında ilk olarak yapılması gereken adım olan bilgisayar sistemlerinin matematik modellenin elde edilmesi ile ilgili deneyim edinilmiş ve herhangi bir bilgisayar sisteminin modellenmesinde izlenmesi gereken işlem adımları görülmüş oldu.

4. SANAL VERİTABANI SUNUCUSU SİSTEM ALT YAPISI VE MODELLENMESİ

Bu tez çalışması esnasında, pek çok bilgisayar sistemi ve bu sistemlerdeki süreçlere ilişkin problemler incelenmiş, kontrol kuramının bu problemlerin çözümünde uygulanabilirliği konusunda bilgi sahibi olunmuştur. Bu bilgiler ışığında teknolojik açıdan önemli ve güncel bir konu olan sanallaştırma ve bulut mimarisi konusundaki problemlerle ilgilenilmesinin uygun olacağına karar verilmiştir. Sanal ortamdaki sunucuların, özellikle veri tabanı sunucularının kaynak yönetiminin, sistem performansını etkilediği bilinmektedir [42, 43, 9, 70]. Veri merkezindeki sanal sunucuların yönetiminde amaç ölçütü, sınırlı kaynakları etkin bir şekilde kullanarak sistem çıkışlarını istenilen seviyede tutmak olarak düşünülebilir. Etkili kaynak kullanımıyla kastedilen; değişen iş yükü altındaki sanal makinaların her birine gelen iş istekleri karşılanırken kaynak sıkıntısı yaşanmaması, diğer taraftan iş isteklerinin en kısa sürede tamamlanması için gereken kaynak tahsisinin sağlanmasıdır. Tez çalışmasında üzerinde deneysel çalışmalar yapılabilmek için gerçek bir sanal sunucuya gereksinim vardır, bu amaçla, sanallaştırma için fiziksel makinaya VMware iş istasyonu kurulmuştur. Bu VMware iş istasyonu içine bir VMware ESXi konak sunucu (host), bu ESXi konak sunucu içine de veri tabanı sunucu makinaları kurulmuştur. Veri merkezinin genel mimari şeması Şekil 4.1’de gösterilmiştir. Bu çalışma kapsamında veri merkezi içerisindeki konak makina içinde bulunan sanal veri tabanı sunucu makinalarının CPU kullanımı ve bellek tüketimi, referans girişler dikkate alınarak ve CPU ve bellek kaynak tahsisi ile dinamik olarak kontrol edilmesi amaçlanmıştır. Sanal veri tabanı sunucusu ile aynı kaynakları kullanan diğer bir uygulamayı göstermek üzere veri merkezinde mevcuta ek olarak diğer bir veri tabanı sanal sunucusu oluşturulmuştur. Sanal veri tabanı makinasının CPU kullanımı ve bellek tüketimi kontrol edilebildiği takdirde veri merkezindeki tüm sanal makinaların kaynak kullanımı yeterli kaynak tahsisi ile kontrol edilebileceği düşünülmüştür. Böylelikle, ihtiyaca göre kaynak tahsisi ile veri merkezindeki tüm uygulamalar için servis kalite garantisi sağlanmış olacaktır. Ayrıca veri merkezindeki diğer

uygulamalar için ihtiyaç olduğu takdirde daha fazla CPU ve bellek kaynakları tahsis edilebilecektir. Bu amaç doğrultusunda ilk olarak sanal ortamdaki veri tabanı sunucusunun matematiksel modeli oluşturulmalıdır. Bu bölümde sanal veri tabanı sunucu makinasının alt yapısı ile ilgili bilgi verilecek ve modellenmesi çalışmaları sunulacaktır.

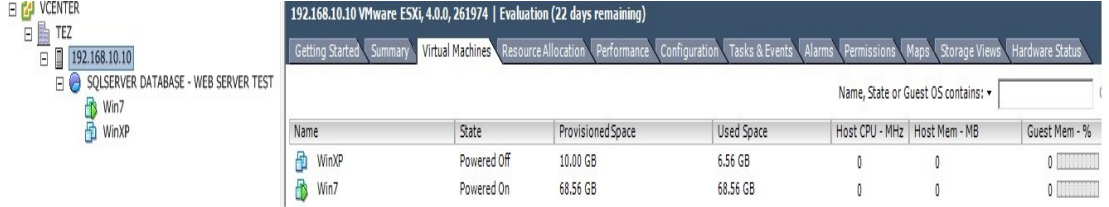


Şekil 4.1 : Veri merkezi genel mimarisi.

4.1 Sanal Veri Tabanı Sunucusu Sistem Alt Yapısı

Tez çalışması kapsamında VMware sanallaştırma teknolojisini kullanan ve test ortamı olarak işlev görecektir bir veri merkezi hazırlanmıştır. Veri merkezinde pek çok konak bulunabileceği gibi her konak makinenin üzerinde bir ya da daha fazla sanal makineler bulunabilir. Her bir sanal makinenin kendine özgü işletim sistemi, bellek, CPU, katı disk, cd-rom, ağ bağdaştırıcı gibi kaynakları bulunur. Emülatör olarak da

isimlendirdiğimiz test ortamında, 192.168.10.10 internet protokol adres (IP)'li bir konak makinası oluşturuldu. 192.168.10.10 konak makinası üzerinde 192.168.10.12 IP'li Win7 isimli veri tabanı sanal makinası ve 192.168.10.11 IP'li WinXP isimli diğer bir veri tabanı sanal makinası oluşturuldu. Kurulan sanal makinalar Şekil 4.2 ile gösterilmiştir.



192.168.10.10 VMware ESX, 4.0.0, 261974 | Evaluation (22 days remaining)

Getting Started Summary Virtual Machines Resource Allocation Performance Configuration Tasks & Events Alarms Permissions Maps Storage Views Hardware Status

Name, State or Guest OS contains: *

Name	State	Provisioned Space	Used Space	Host CPU - MHz	Host Mem - MB	Guest Mem - %
WinXP	Powered Off	10.00 GB	6.56 GB	0	0	0
Win7	Powered On	68.56 GB	68.56 GB	0	0	0

Şekil 4.2 : 192.68.10.10 IP'li ESX sunucu üzerinde yaratılan sanal makinalar.

Tüm sanal platformun üzerinde bulunduğu ESX konak sunucu makinası için 5 Gigabayt (GB) bellek, 4 adet işlemci, 60 Gigabayt (GB) katı disk ve 140 GB ikinci bir katı disk ayrıldı. ESX sunucu yapılandırması aşağıda Şekil 4.3'de ve Şekil 4.4'de gösterilmiştir.

ESX Server

State: Powered off
Guest OS: ESX Server 4.0
Location: C:\Virtual Machines\ESX Server\ESX Server.vmx
Version: Workstation 6.5-7.0 virtual machine

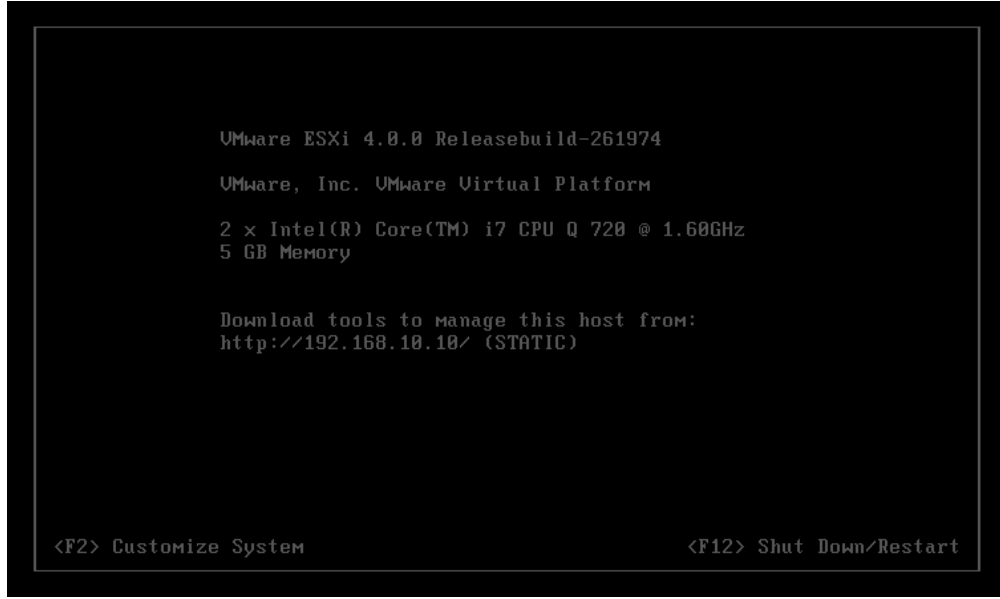
Commands

- Power on this virtual machine
- Edit virtual machine settings
- Enable ACE features (What is ACE?)

Devices

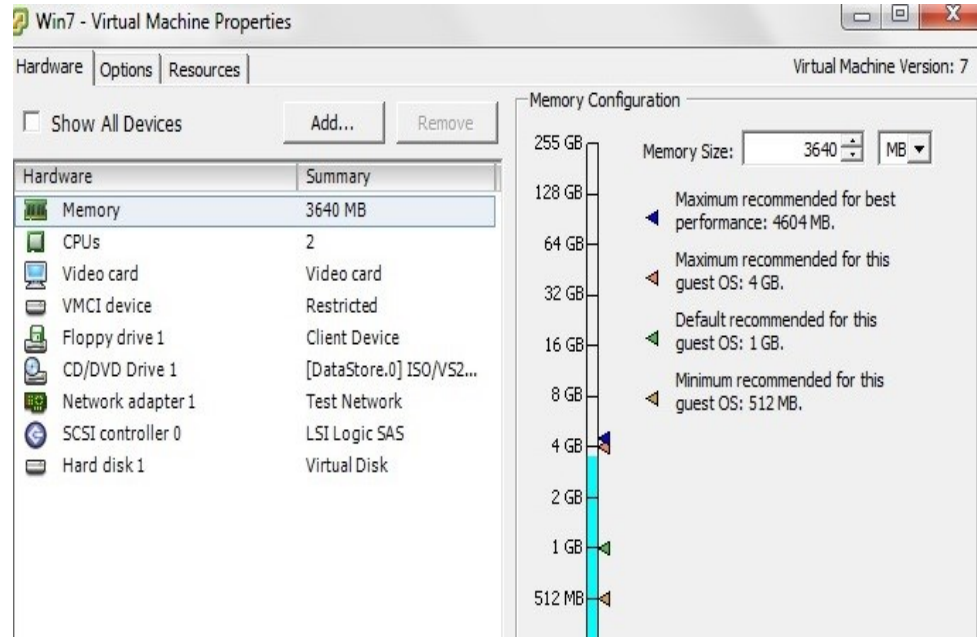
Device	Options
Memory	4608 MB
Processors	4
Hard Disk (SCSI)	60 GB
Hard Disk 2 (SCSI)	140 GB
CD/DVD (IDE)	Using file C:\ISO Images\Windows 7 Professional\Windows 7 Professional.iso
Network Adapter	Host-only
Network Adapter 2	Host-only
Network Adapter 3	Bridged
Display	Auto detect

Şekil 4.3 : ESX sunucu yapılandırması gösterilmektedir.



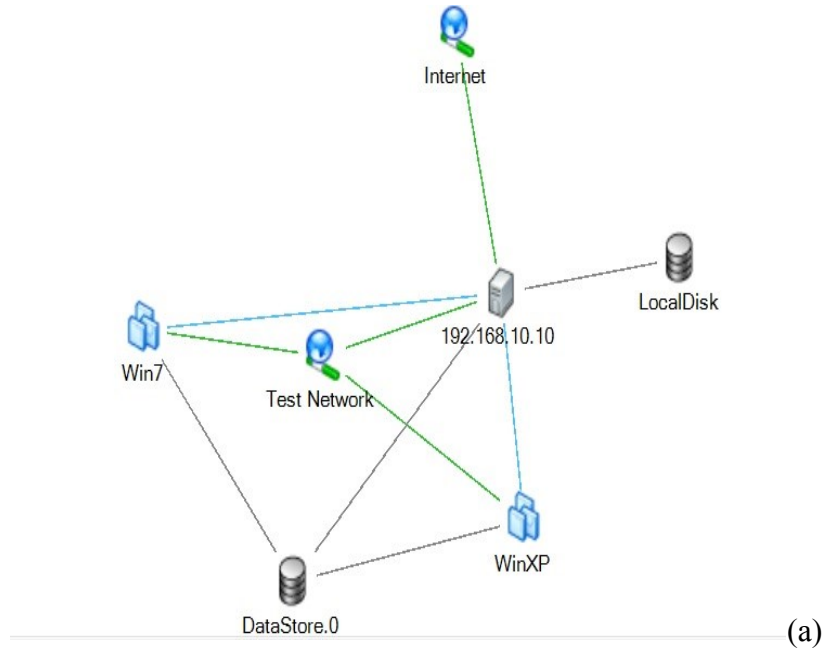
Şekil 4.4 : ESX sunucu makinası Red Hat işletim sistemi üzerinde kurulmuş bir makinadır. Tüm sanal platform, bu sistem üzerinde yer alır.

Sanal makinalar konak sunucusunun kaynaklarını kullanır. Her bir sanal makina birbirinden bağımsız olarak CPU, bellek ve disk kaynakları ayrılabilir. Şekil 4.5 ile veri tabanı sanal makinasına ilişkin kaynaklar gösterilmektedir. Veri tabanı sanal makinaları konak sunucu üzerinde yer alır ve bu sanal makinalar için maksimum 4 GB bellek, üç adet i7 CPU Q720 1,6 Gigahertz (GHz) mantıksal işlemci tahsis edilmiştir. Konak sunucu ve sanal makinalar ağa 100 Mega bit per second (Mbps) yerel ağ bağdaştırıcı ile bağlıdır. Win7 veri tabanı sanal makinası üzerinde Windows 7 işletim sistemi ve SQL Server 2008 R2 kurulmuştur [6]. Win7 makinasında SQL Server üzerinde 'DS' isimli veri tabanı oluşturulmuştur. DS veri tabanını kullanacak uygulama için DS veri tabanı üzerinde tablo, fonksiyon ve izlek objeleri oluşturulmuştur. WinXP veri tabanı sanal makinasında ise Windows XP işletim sistemi ve ve SQL Server 2008 R2 kurulmuştur. WinXP makinasında SQL Server üzerinde 'DS2' isimli veri tabanı oluşturulmuştur. Bir sanal makina Web sunucusu olarak da görev yapabilir. Sanal makina Web sunucu olarak görev yapacaksa internet bilgi sunucusu (IIS) sanal makina kurulanabilir. IIS ile ilgili teknik bilgiye [71] numaralı referans ile erişilebilir. Bu çalışmada veri tabanı sanal makinaları üzerine odaklanılmıştır.

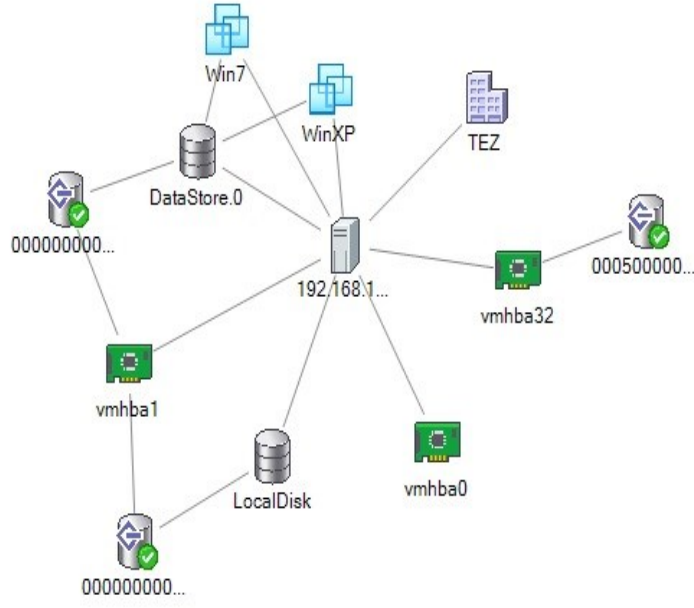


Şekil 4.5 : Win7 sanal makinesine ilişkin kaynaklar gösterilmektedir.

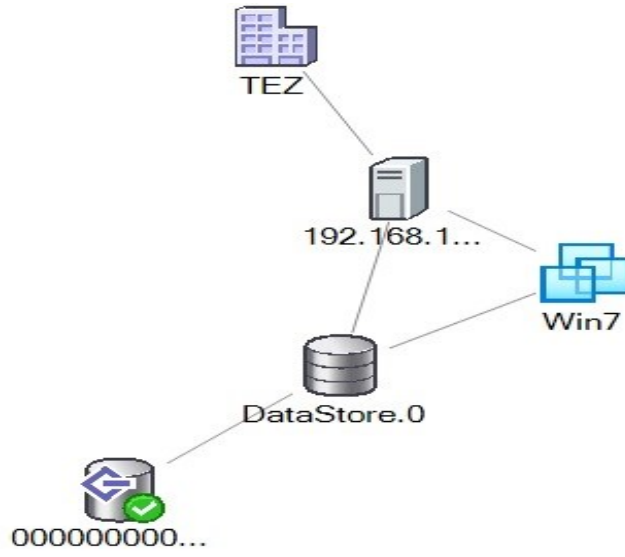
Sanal makineler bir ağ üzerinde yer almaktadır. Bu ağ üzerinden her bir sanal makina birbiriyle ve ESX konak sunucu ile haberleşebilir. Şekil 4.6 ve Şekil 4.7’de sanal sistemin ağ yapısı ile ilgili ağ şemaları gösterilmektedir.



Şekil 4.6 : Konak, sanal makina ve disklerin ilişkisini gösteren ağ şemaları.



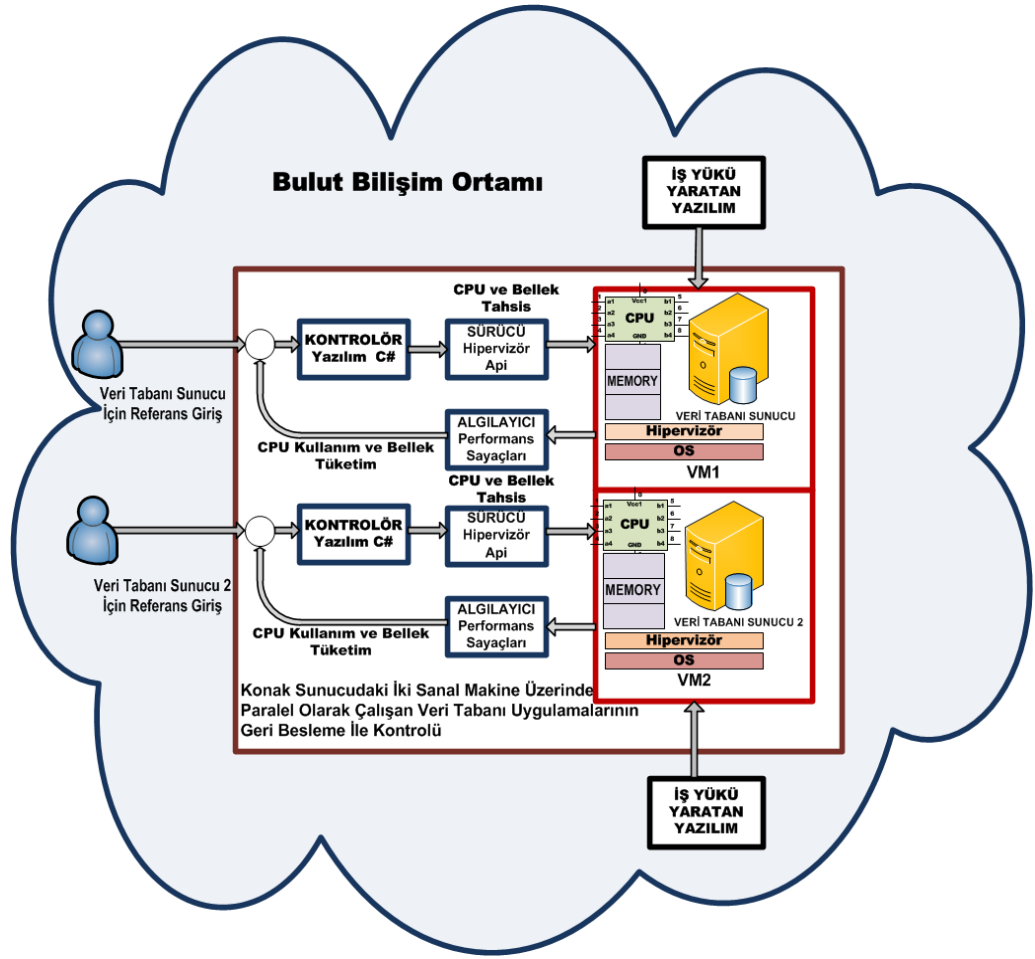
(b)



(c)

Şekil 4.7 : Konak, sanal makina ve disklerin ilişkisini gösteren detaylı ağ şemaları.

Konak ve sanal makinalardan oluşan test sistemi ayrıca bu sistemi kontrol eden sisteme ilişkin genel şema Şekil 4.8 ile gösterilmiştir.



Şekil 4.8 : Sanal makinalar ve bu makinaların kontrolüne ilişkin genel şema.

Bu tez çalışması kapsamında, veri merkezindeki veri tabanı sanal sunucu makinasının kontrolü için geri besleme çevrimi oluşturmak üzere algılayıcılar, sürücüler ve kontrolörler gerçekleştirilmiştir.

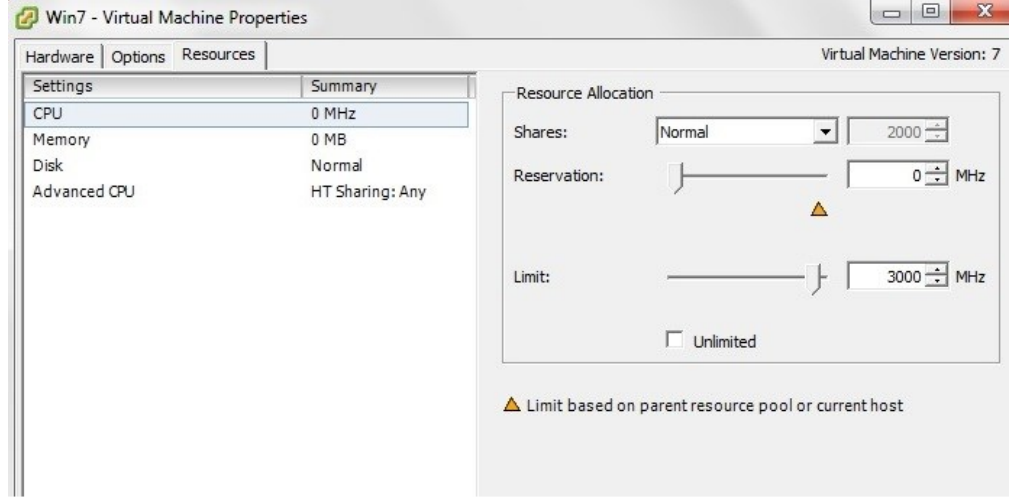
4.1.1 Algılayıcılar

Algılayıcılar, deney ortamında CPU kullanımı ve bellek tüketimi sistem çıkışlarını ölçmek için kullanılır. Sistem çıkışları ile ilgili geçmiş bilgiler de algılayıcılar vasıtasıyla elde edilebilir. VMware performans yöneticisi, sistemin performans sağlayıcıları tarafından yaratıldığı ve devam ettirildiği takdirde, sistem performansı ile ilgili farklı görünümelerde istatistiksel bilgi elde etmek için bir hizmet ara yüzü sağlar. Performans sağlayıcısı, performans metriklerini yaratan herhangi bir yönetilen objedir. Bir sağlayıcı konak sunucusu, sanal makinalar veya sistem kaynakları ile ilgili bilgiler içerebilir. Bu tez çalışmasında gerçekleştirilen algılayıcılar, CPU kullanımı ve bellek tüketimi ile ilgili istatistiksel bilgiye erişmek için VMware performans yöneticisine uygulama program ara yüz (API) çağrıları

yapan C# programlama diliyle yazılmış yazılım parçalarıdır [72]. Algılayıcılar, bellek için bellek tüketim (*Memory Consumed*) performans sayacına ilişkin veriyi mutlak değer olarak, kilobayt birimi cinsinden ve örnekleme zamanı olan 20 saniye içindeki ortalama değer olarak geri döndürür. CPU için ise CPU kullanım performans sayacını ilişkin veriyi (*CPU usagemhz*) mutlak değer olarak megahertz birimi cinsinden ve örnekleme zamanı olan 20 saniye içindeki ortalama değer olarak geri döndürür

4.1.2 Sürücüler

Test ortamında, bu tez çalışması kapsamında gerçekleştirilen sürücüler sanal makinalara CPU ve bellek tahsisinde kullanılan yazılımlardır. Sanal makinaların CPU ve bellek kaynakları ile ilgili '*limit*' parametresi kaynakların sürülmesi için kullanılır ve konak sunucusunun sanal makinaya ayırabileceği CPU ve bellek kaynakları için bir üst sınır belirler. Sanal makina, kullanılmayan başka kaynak olsa dahi kaynak kullanımında bu üst sınırı aşamaz. Bu özellik belli bir performans değerini garanti etmek için kullanılır. Limit parametresi Şekil 4.9 ile gösterilen ilgili yönetim konsolundan manuel olarak değiştirilebileceği gibi VMware API çağrıları ile de değiştirilebilir. Limit parametresinin API çağrıları ile değiştirilmesi şu şekilde yapılır: Sanal makinanın pek çok özelliğini içeren sunucu tarafındaki objesinin bir kopyası kullanıcı tarafında oluşturulur ve kullanıcı tarafında oluşturulan objenin özellikleri değiştirilir ve bu obje sunucu tarafında değiştirilmiş haliyle güncellenir. Deney ortamında, veri tabanı sanal sunucu makinasının CPU ve bellek tahsisi, CPU ve bellek kaynakları ile ilgili limit parametresini ayarlayan VMware API çağrıları ile yapılır [72].



Şekil 4.9 : Veri tabanı sanal makinası için CPU kaynak tahsisinin yapıldığı yönetim ara yüzü.

4.1.3 Kontrolör

CPU kullanımı ve bellek tüketimine ilişkin sistem çıkışları her bir geri besleme çevriminde algılayıcılar vasıtasıyla ölçülür. Sistem çıkışlarının algılayıcılar tarafından ölçümü sonrası, kontrolör bir sonraki geri besleme çevriminde her bir sistem çıkışı ile referans girişi arasındaki fark olarak tanımlanan hatayı dikkate alarak bellek ve CPU kaynak tahsisini dinamik olarak değiştirir yani sisteme uygun kaynak tahsisi yapılmış olur. Bu çalışmada CPU dinamiği ve bellek dinamiği SISO olarak da modellenemediği iki ayrı kontrolör tasarlanmıştır. Diğer taraftan sistemin etkileşimli MIMO modeli temel alınarak da bir kontrolör tasarlanmış ve iki kontrol yapısının performansları simülasyon düzeyinde karşılaştırılmıştır. Bu çalışma kapsamında başarımının yüksek olması nedeniyle çoklu SISO modellere odaklanılmış yani gerçek zamanlı uygulama CPU ve bellek için tasarlanan kontrolörler ile yapılmıştır. Kontrolörler simülatörde test edilmiş ve emülatör üzerindeki veri tabanı sanal sunucu makinasının geri besleme kontrolünde gerçek zamanlı olarak kullanılmıştır.

4.1.4 Veri tabanı sunucu iş yükü aracı - SQLStress

SQLstress iş yükü yaratıcısı ile veri tabanı sanal makinası üzerinde yapay iş yükü yaratılabilmekte ve iş yükünün büyüklüğü ayarlanabilmektedir [73]. Sanal veri tabanı sunucu makinelerden birine Win7 ismi verilmiştir. Win7 sanal veri tabanı sunucundaki DS isimli SQL Server veri tabanına iş yükü yaratmak üzere bağlantı sağlayan ön yüz Şekil 4.10 ile gösterilmiştir. Bağlantı sağlandıktan sonra ekrandan

girilen parametreler ile veri tabanında istenilen sayıda kullanıcı ve iş yükü yaratmak mümkün olabilmektedir.

Şekil 4.10 : Sanal veri tabanı sunucusuna bağlantı sağlamak üzere ön ekran aracılığıyla parametreler girilebilir.

Load	Operation	Weight	MSDTC
10%	Delete:	1	☐
50%	Insert:	5	☐
20%	Update:	2	☐
20%	Select:	2	☐
****	Round Trips:	0	☐
****	Execute SP:	0	☐
****	Execute WS:	0	☐

Şekil 4.11 : Sanal veri tabanı sunucusu üzerinde iş yükü yaratan ara birimin parametre tablosu.

Örnek olarak verilen Şekil 4.11 ile DS veri tabanında 800 kullanıcı yaratılıyor. İşlem seviyesi (transaction rate) saniyede 800 olarak girilmektedir. İş yükü içerisinde kayıt ekleme, kayıt düzeltme, kayıt silme ve seçme işlemlerine ağırlıklar tahsis edilmektedir. Bu ağırlıklara göre işlemler gerçekleştirilmektedir. Her bir kayıt ekleme işleminin veri tabanına tek bir kayıt ekleyeceği parametrik olarak ayarlanmaktadır. Deneyler sırasında farklı iş yükleri sisteme uygulanmış ve ilgili performans sayaçları ile sistemin işleyişi hakkında bilgi edinilmiştir. Değişen iş yükü Şekil 4.11 örneği görülen iş yükü aracı yardımıyla ayarlanmıştır. SQLStress ekranındaki iş yükü yaratma ile ilgili parametreler ve bu parametrelerin anlamları aşağıda açıklanmıştır.

Users/Database, Users/Workload: Veri tabanında yaratılacak sanal kullanıcı sayısını ifade eder. Her bir kullanıcı için bir bağlantı yaratılır. Bağlantı koptuğu zaman yeni bir bağlantı yaratılır.

Tnx/Second: Bir saniyede gerçekleşmesi istenen işlem sayısını ifade eder.

Tnx/User: Bir kullanıcı tarafından veri tabanında gerçekleşecek işlem sayısını ifade eder.

Rows/Insert: Veri tabanına her bir insert işlemi sırasında kaç kayıt insert edileceğini belirten parametredir.

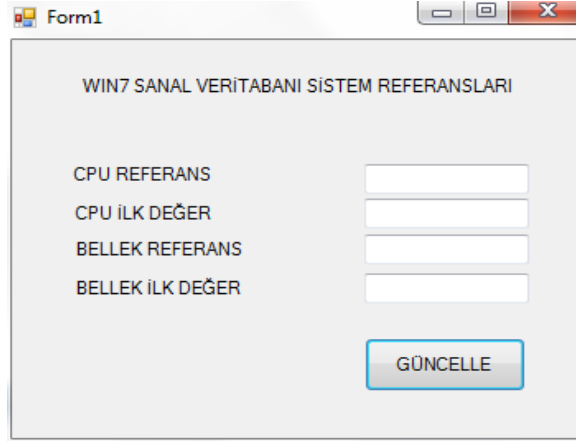
Bytes/Doc: Veri tabanına insert işlemlerinde insert edilen kayıt verisinin yanı sıra image kolonuna yazılacak bilginin bayt olarak ortalama büyüklüğünü ifade eder. Bu değer 0 ise bu kolona bilgi yazılmadığı anlamına gelir.

Veri tabanında iş yükü; kayıt ekleme, silme, sorgulama, güncelleme yaparak ya da stored program (SP) çalıştırarak yapılabilir. Yaratılan iş yükünde bu işlemlerin ağırlığı verilebilir. Böylelikle ilgili işlem tipine ilişkin iş yükü istenilen oranda yaratılabilir.

Web sunucusu üzerinde iş yükü [74] numaralı referanstaki Web uygulamaları iş yükü yönetici araç kullanılarak oluşturulabilir. Ayrıca, iş yükü yaratan diğer yönetici araçlar; IBM iş yükü yöneticisi [75], HP iş yükü yöneticisi [76] ve Dell iş yükü yönetici araçları [77] incelenmiştir.

4.1.5 Test ortamı sistem girişleri ara yüzü

CPU ve bellek referans girişleri kullanıcı tarafından Şekil 4.12 ile gösterilen bir program ara yüzü ile girilebilmekte ve iş yükünün büyüklüğü daha önce açıklandığı gibi SQLStress programı ile ayarlanabilmektedir. DS veri tabanına bağlanılarak Şekil 4.13'te gösterilen veri tabanı sorgulama ara yüzü aracılığıyla tablodaki kayıtların sayısı sorgulanabilmektedir. Böylelikle veri tabanı işlerinin tamamlanma hızı gözlemlenebilmektedir.



Form1

WIN7 SANAL VERİTABANI SİSTEM REFERANSLARI

CPU REFERANS

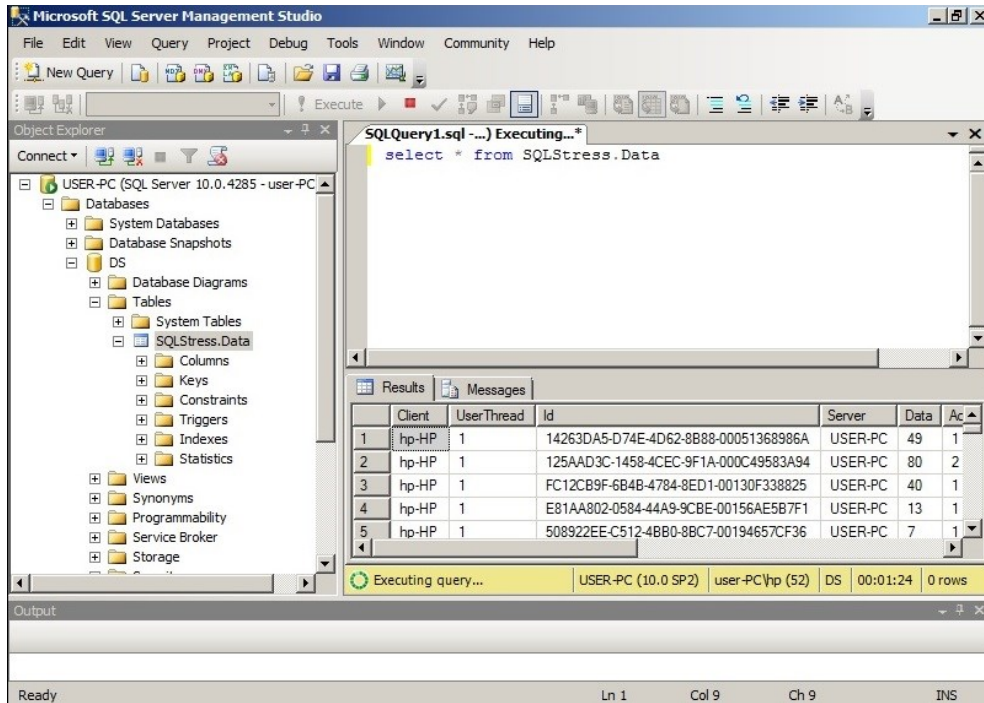
CPU İLK DEĞER

BELLEK REFERANS

BELLEK İLK DEĞER

GÜNCELLE

Şekil 4.12 : CPU ve bellek referanslarının belirlendiği kullanıcı ara yüzü.



Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Community Help

Object Explorer

Connect

SQLQuery1.sql (...): Executing...*

select * from SQLStress.Data

	Client	UserThread	Id	Server	Data	Ac
1	hp-HP	1	14263DA5-D74E-4D62-8B88-00051368986A	USER-PC	49	1
2	hp-HP	1	125AAD3C-1458-4CEC-9F1A-000C49583A94	USER-PC	80	2
3	hp-HP	1	FC12CB9F-6B4B-4784-8ED1-00130F338825	USER-PC	40	1
4	hp-HP	1	E81AA802-0584-44A9-9CBE-00156AE5B7F1	USER-PC	13	1
5	hp-HP	1	508922EE-C512-48B0-8BC7-00194657CF36	USER-PC	7	1

Results Messages

Executing query... USER-PC (10.0 SP2) user-PC\hp (52) DS 00:01:24 0 rows

Output

Ready Ln 1 Col 9 Ch 9 INS

Şekil 4.13 : Tamamlanan veri tabanı isteklerini gözlemleyebilmek için DS SQL sunucu veri tabanındaki kayıtlar sorgulanabileceği sorgulama ara yüz ekranı.

4.2 Sistem Tanıma

Bu tez çalışması kapsamında bulut ortamındaki sanal veri tabanı sunucusunun CPU kullanımı ve bellek tüketiminin kontrol edilmesi amaçlanmıştır. Bulut ortamı ve sanallaştırma konusunda mevcuttaki çalışmalara katkı sağlamak ve bu konudaki açık olan bir probleme kontrol teorisi ve tekniklerini kullanarak çözüm getirmek için güncel literatür incelenmiştir [78-94]. Sistemin kontrol edilebilmesi için önce sistemin matematik modelinin elde edilmesi gerekir. Veri tabanı sisteminin fiziksel bir yapı olarak ele alınması mümkün olmadığı için matematik modelinin elde edilmesinde deney ve gözleme dayanan sistem tanıma teknikleri kullanılmıştır. Sistemin sadece giriş-çıkış verilerini kullanan kara-kutu yaklaşımı bu süreçte modeli elde etmek için en uygun yaklaşımdır. Sistem modelini elde etmek için MATLAB SIT paket programı kullanılmıştır. Sanal veri tabanı sunucusunun modelini elde etmek amacıyla önce, sistem dinamiklerini yansıtan sistem giriş işareti seçimi için pek çok giriş işareti ile deney yapılmış ve sinüzoidal işaretin model elde etmek için yeterli olduğu kanısına varılmıştır. Verideki ortama değerler, sapmalar ve bozukluklar temizlenmiş ve var olan giriş çıkış verileri; model tahmini ve doğrulama yapmak için iki kısma ayrılmıştır. Farklı model yapıları ve çeşitli sistem tanıma teknikleri en uygun modeli elde etmek için denenmiş ve elde edilen modeller doğrulama verisi ile sınanmıştır.

Tüm deneylerde sistem örnekleme periyodu 20 saniye olarak seçilmiştir. Bu örnekleme periyodu sistem girişlerine karşılık sistemin çıkış zamanı düşünülecek olursa yeterli görülmüştür. Hypervizör, örnekleme periyodu süresince sistem çıkışının aldığı değerlerin ortalamasını alarak örnekleme zamanı sonunda bu ortalama değeri sistem çıkışı olarak atamıştır.

4.2.1 Giriş-çıkış değişkenleri ve sistem tanıma deneyi için giriş işareti seçimi

Modellemesi yapılacak dinamiklere ilişkin giriş ve çıkış değişkenleri şu şekilde tanımlanabilir: Sanal veri tabanı makinası üç giriş ve iki çıkışa sahiptir. CPU tahsis ve bellek tahsis sırasıyla C_a , M_a olarak isimlendirilmiştir. İş yükü bozucu giriş olarak tanımlanmış ve W olarak isimlendirilmiştir. Sanal sunucu makinasının çıkışları C_u , CPU kullanımı ve M_c ise bellek tüketimidir. Sistem üzerinde yapılan deneyler sonucu, sistemin C_a girişinin 500 Megahertz (MHz) and 3000 MHz arasında

değerler alabildiği, M_a girişinin ise 150 MB ile 2700 MB değerleri arasında değerler alabileceği görülmüştür. Sistem girişlerinin aldığı değerlerin minimum değerlerden daha küçük olması durumunda sistem çalışmadığından tanıma deneyleri esnasında girişler bu minimum değerlerden daha büyük seçilmiştir. Sistem modelinin elde edilebilmesine olanak veren birçok giriş işareti tanıma deneyinde kullanılmış ve sinüzoidal girişin sistemle ilgili verdiği bilginin -gerek dinamik açısından gerekse yapısal olarak- yeterli olduğu gözlenmiştir. Süreç sinüzoidal bir giriş işaretine karşılık sinüzoidal sistem çıkışı verdiği için matematik modelinin lineer olduğunun varsayılması uygun bulunmuştur.

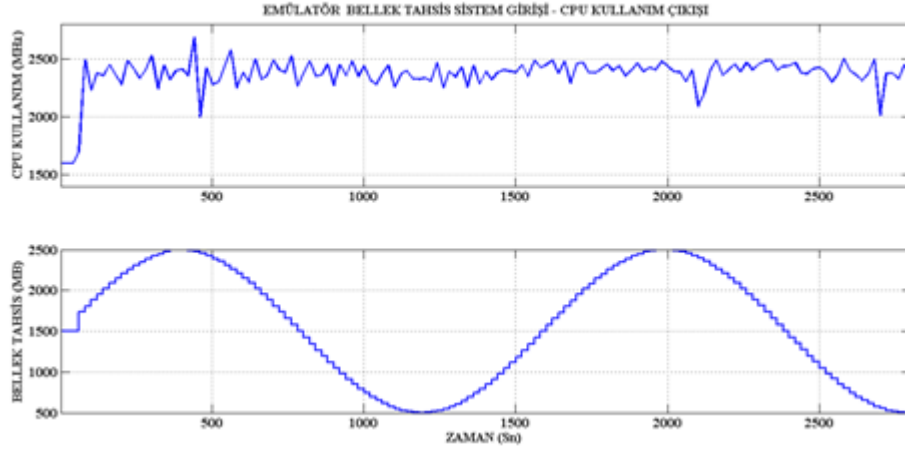
4.2.2 Sistem tanıma deneyleri, model yapısı ve model yapılarının karşılaştırılması

Literatürde yapılmış olan çalışmalar genellikle tek giriş tek çıkış (SISO) sistemler üzerineydi. Bu çalışmalardan farklı olarak üzerinde çalıştığımız sanal veri tabanı sunucu sistemi C_a , M_a , W sistem girişlerine ve C_u , M_c sistem çıkışlarına sahiptir. Bu sebeple sistemin MIMO modeli sistem tanıma teknikleri ile elde edilebilir. Bununla birlikte iki girişli sistemin bir girişi, diğer girişten etkilenmiyorsa ya da çok az etkileniyorsa sistemin modeli tek giriş-tek çıkışlı (SISO) olarak elde edilebilir. Veri tabanı sanal sunucu makinasının giriş-çıkış ilişkisini ortaya koymak için bir dizi deney yapılmıştır. Bu deneylerde girişlerden biri ayrık sinüzoidal bir işareti ile sürülürken diğer giriş işareti sabit tutulmuş ve sistem çıkışları gözlemlenmiştir.

$$C_a = 2500 \text{ MHz}$$

$$M_a = 1500 + 1000 \sin\left(\frac{2\pi t}{1600}\right) \text{ MB}, W = \text{sabit} \quad (4.1)$$

Bu deney sonucunda, (4.1) ilişkileri ile tanımlanan işaretler sisteme uygulandığı zaman, M_a sistem girişinin C_u sistem çıkışı üzerinde neredeyse hiç etkisinin olmadığı tespit edilmiştir ve bu durum Şekil 4.14 ile gösterilmiştir.

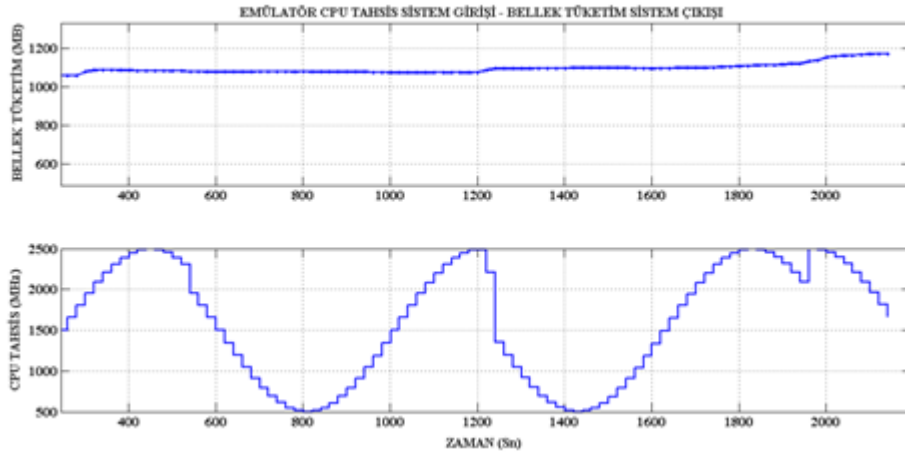


Şekil 4.14 : Bellek tahsis (M_a) – CPU kullanım (C_u).

$$C_a = 1500 + 1000 \sin\left(\frac{2\pi t}{800}\right) \text{ MHz}, W = \text{sabit}$$

$$M_a = 2500 \text{ MB} \quad (4.2)$$

Diğer taraftan, (4.2) ilişkileri ile tanımlanan işaretler sisteme uygulandığı zaman C_a sistem girişinin M_c sistem çıkışı üzerinde neredeyse hiç etkisinin olmadığı Şekil 4.15'te gösterilmiştir.



Şekil 4.15 : CPU tahsis (C_a) – Bellek tüketimi (M_c).

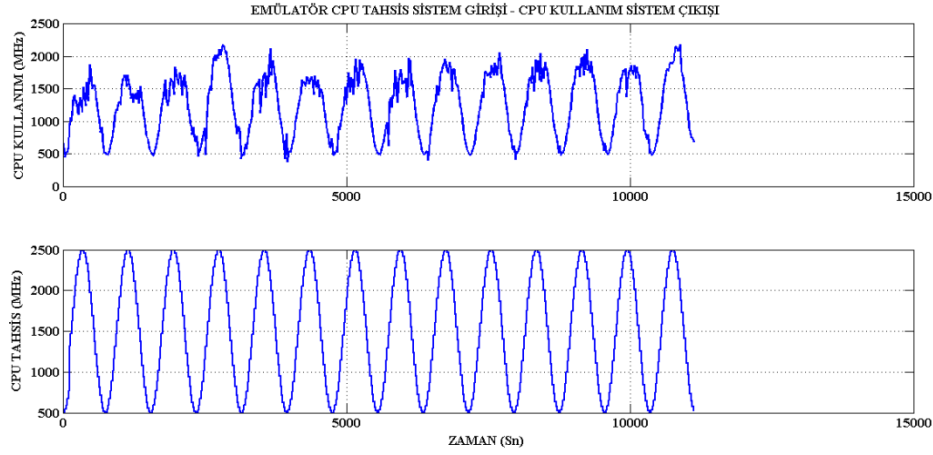
Yukarıdaki deneyler ışığında C_a sistem girişi ile M_a sistem girişi arasındaki ilişkinin çok az olduğu, bu sebeple sanal veri tabanı sunucu sistemi iki SISO sistem olarak modellenebilir olduğu sonucuna varılmıştır. İki SISO sistemden biri için C_a ve W sırasıyla sistem girişi ve bozucu girişi, C_u sistem çıkışıdır. Diğer SISO sistem için M_a ve W sırasıyla sistem girişi ve bozucu girişi, M_c sistem çıkışıdır.

MIMO ve çoklu SISO modelleri analiz etmek ve bu modellerin geçerliliğini karşılaştırmak üzere gerçekleştirilen bazı deneyler aşağıda açıklanmıştır. Sistemin matematiksel modelini elde etmek için sistem girişleri olan CPU ve bellek tahsisi sinüs şeklinde ve bozucu giriş olan iş yükü rastgele değerler alacak şekilde sanal sunucu makinasına (4.3) ifadeleri ile gösterildiği gibi uygulanır. Yaratılan iş yükü yoğun bir iş yüküne karşılık gelir. Elde edilen sonuçlar Şekil 4.16-21 arasındaki grafiklerle gösterilmiştir.

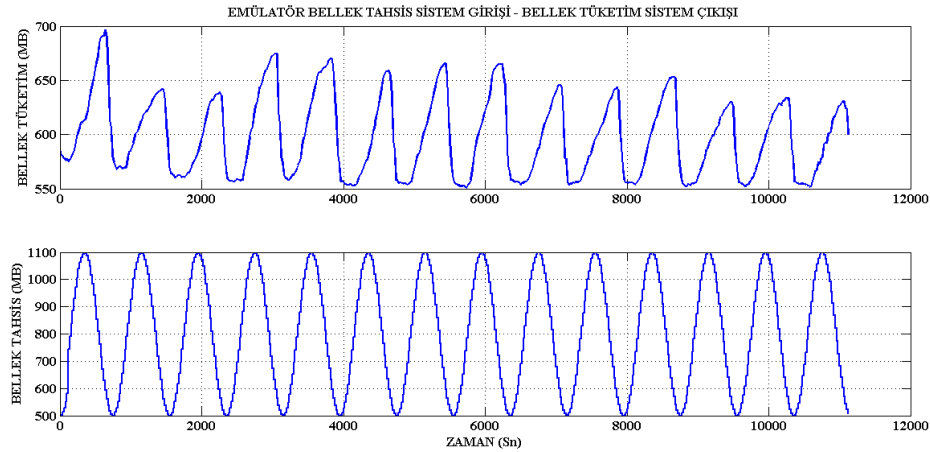
$$C_a = 1500 + 1000 \sin\left(\frac{2\pi t}{800}\right) \text{MHz}$$

$W = 100$ Kullanıcı, ortalama 100 kayıt/saniye rastgele iş yükü ve her bir işlem için 10000 bayt ek bilgi,

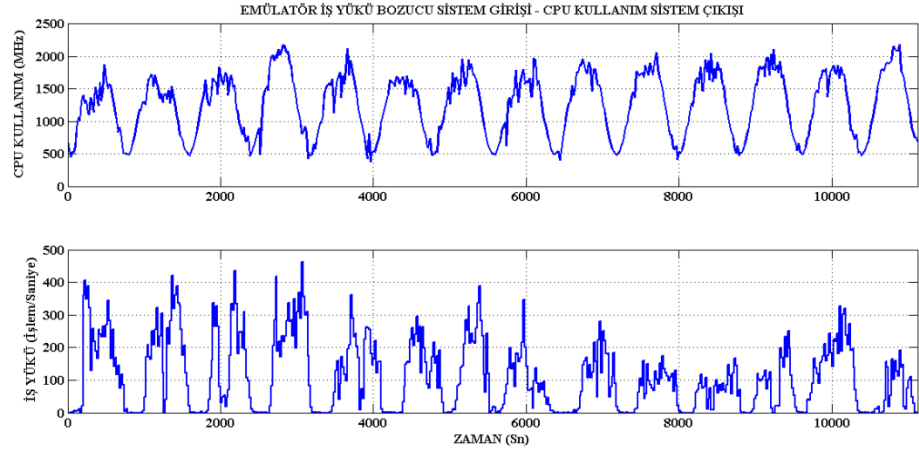
$$M_a = 800 + 300 \sin\left(\frac{2\pi t}{800}\right) \text{MB} \quad (4.3)$$



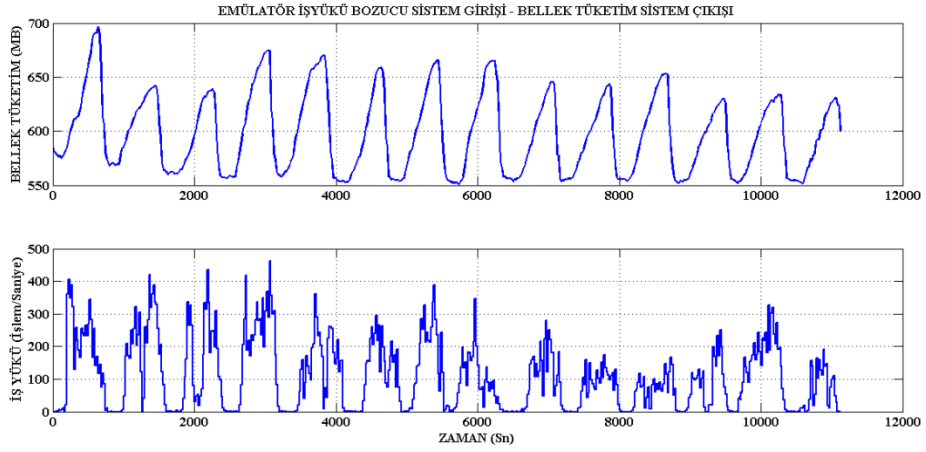
Şekil 4.16 : C_a sistem girişi ve C_u sistem çıkışının zamana göre değişimi.



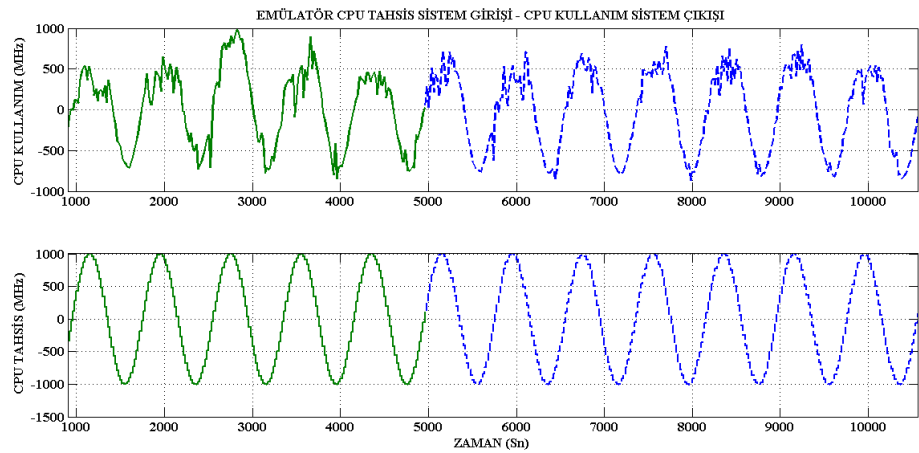
Şekil 4.17 : M_a sistem girişi ve M_c sistem çıkışının zamana göre değişimi.



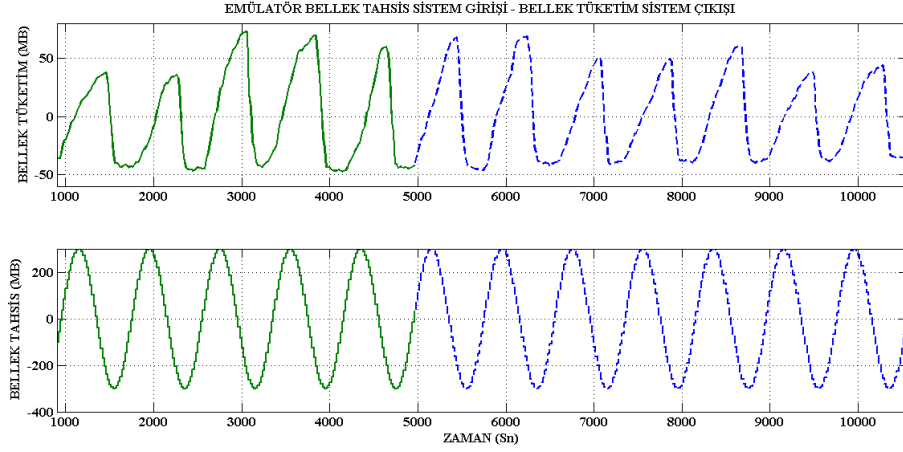
Şekil 4.18 : W iş yükü ve C_u sistem çıkışının zamana göre değişimi.



Şekil 4.19 : W iş yükü ve M_c sistem çıkışının zamana göre değişimi.



Şekil 4.20 : C_a sistem girişi ve C_u sistem çıkış verileri ortalama değer çıkartıldıktan sonra tahmin ve doğrulama kısımları olmak üzere iki kısma ayrılır; yeşil kısım tahmin verisi, mavi kısım doğrulama verisidir.



Şekil 4.21 : M_a sistem girişi ve M_c sistem çıkış verileri ortalama değer çıkartıldıktan sonra tahmin ve doğrulama kısımları olmak üzere iki kısma ayrılır; yeşil kısım tahmin verisi, mavi kısım doğrulama verisidir.

Modelin elde edilmesi için emülatör sistemde sürdürülen deneyler sonucu elde edilen veri, MATLAB ortamına aktarılmış ve sistem giriş ve çıkış verileri olarak MATLAB SIT üzerinde tanımlanmıştır. Verideki bozukluk ve sapmalar SIT vasıtasıyla veriden temizlenmiştir ve farklı modeller elde edilmiştir. Elde edilen modeller doğrulama verisi ile test edilmiş ve en fazla başarımlı gösteren model için sistem matrisleri olan A, B, C, D elde edilmiştir. Aşağıdaki MATLAB komutları sistem modeli elde etmede kullanılmıştır. Deneyler sonucu emülatörden elde edilen sistem verisini kullanarak çoklu SISO ve MIMO sistem modellerini elde etmek için MATLAB simülatörde sırasıyla Şekil 4.22 ve Şekil 4.23 ile gösterilen devre oluşturulmuştur. Ortalama büyüklüğü saniyede yüz işlem olan rastgele iş yükü sisteme etki etmiştir.

```
modeldata(:,5) = modeldata(:,5)/1024;
```

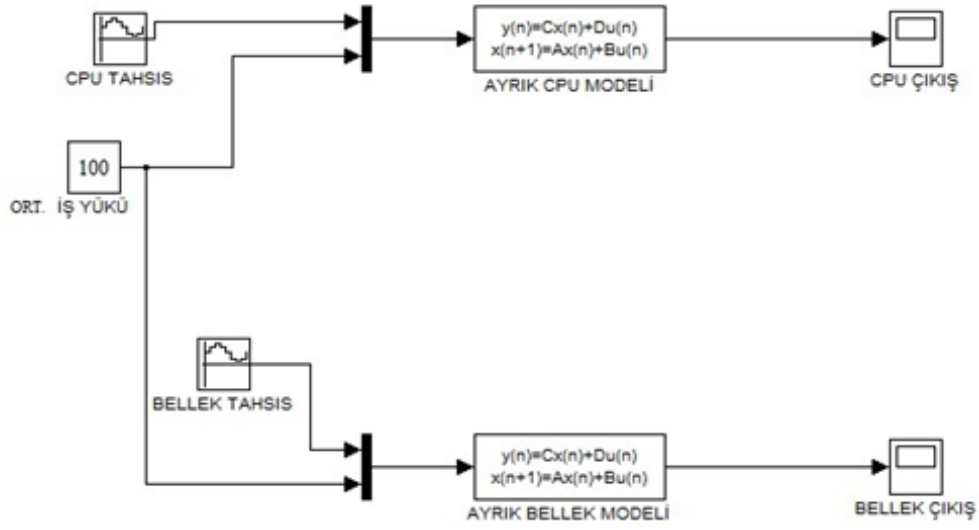
```
modeldata = detrend(modeldata);
```

```
sysgir=modeldata(:,1:3);
```

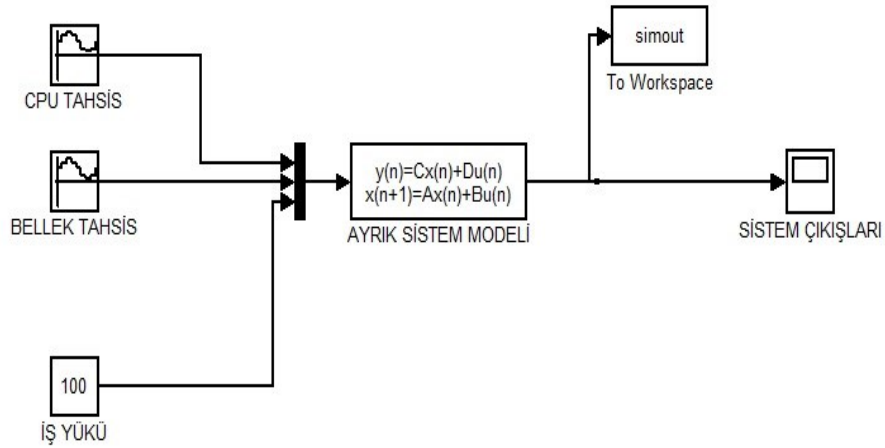
```
syscik = modeldata(:,4:5);
```

```
ident; mod = idss(arx110);
```

```
A = mod.A; B = mod.B; C = mod.C; D = mod.D;
```



Şekil 4.22 : SISO sistem simülasyonu için MATLAB SIMULINK modeli.



Şekil 4.23 : MIMO sistem simülasyonu için MATLAB SIMULINK modeli.

Sistem modelini elde etmek için farklı model yapıları ve yöntemleri denenmiştir. Denenen yöntemler içerisinde “Auto Regressive with External Input (ARX)” sistem tanıma tekniği Çizelge 4.1 ile gösterildiği gibi hem MIMO hem de çoklu SISO sistem için daha iyi sonuçlar vermiştir. Sistem modelini elde etmede Çizelge 4.1’de NLARX21, NLARX31 (Non-linear Auto Regressive with External Input) olarak isimlendirilen lineer olmayan sistem tanıma teknikleri de denenmiştir. Yüksek sistem mertebesi, daha iyi olmayan model başarımı ve lineer olmayan sistemleri kontrol etmedeki güçlük sebebiyle sanal veri tabanı sunucu sisteminin modellenmesinde lineer olmayan sistem modelleri tercih edilmemiştir.

Çizelge 4.1 : MIMO sistem ve çoklu SISO sistemler için ARX ve diğer sistem tanıma teknikleri ile elde edilen model uygunluğu çizelgede gösterilmektedir. R^2 model uyumunun başarımını göstermektedir.

MODEL	MIMO(CPU) R^2	MIMO(MEMORY) R^2	SISO(CPU) R^2	SISO(MEMORY) R^2
ARX110(AutoRegressive)	67,57	56,38	70,35	57,87
ARX111	67,15	51,72	69,92	59,28
ARX112	66,25	57,04	68,06	57,74
ARX113	62,29	55,10	62,58	56,17
ARX221	67,33	50,32	69,32	58,09
ARX222	66,85	50,22	69,27	57,66
ARX223	66,92	53,11	67,34	56,24
ARX224	67,20	57,18	66,11	55,57
ARX225	66,41	57,36	64,73	54,38
ARX2210	67,20	40,34	57,83	55,89
ARX1110	33,85	11,97	24,10	42,52
ARMX2222	67,48	46,71	62,97	57,97
NLARX21 (Non-Linear)	60,75	-4334	69,00	59,42
NLARX31	66,77	-4263	56,69	57,65
N4S3 (Durum Uzayı)	70,41	10,67	60,66	-1093
TF10 (Transfer Fonksiyon)	55,2	-147,7	69,65	57,84
TF21	65,82	58,06	69,28	59,72

Sanal veri tabanı sisteminin sistem girişleri olan CPU tahsisi ve bellek tahsisi sinüzoidal formda olduğu takdirde CPU kullanımı ve bellek tüketimi sistem çıkışları da sinüzoidal formda olmuştur. Bu sonuç veri tabanı sanal sunucu makinasının modelinin lineer bir model ile yaklaşık olarak ifade edilebileceğini göstermektedir.

4.3 Sistem Modeli

Sanal veri tabanı sunucu makinesinin kaynak kontrolü için gerekli giriş-çıkış değerleri için matematik modeli olarak birinci dereceden zamanda değişmeyen ARX modeli yeterli olmuştur. Model parametreleri a_k ve b_m Recursive Least-Squares (RLS) yöntemi ile tahmin edilmiştir. ARX yapısındaki doğrusal model (4.4) ilişkisi ile verilir,

$$y(t) = \sum_{k=1}^n a_k y(t-k) + \sum_{m=0}^{n-1} b_m u(t-m-d) \quad (4.4)$$

a_k , b_m model parametrelerdir, n model mertebesidir ve geçmişteki bilginin ne kadarının dikkate alındığını gösterir. d sisteme bir giriş işareti uygulanması ile bu girişin sistemde ilk etkisinin görülmesi arasında geçen gecikmeyi ifade eder. $y(t)$ $t \times T$ örnekleme zamanında C_u ve M_c sistem çıkışlarının, $u(t)$ ise $t \times T$ örnekleme zamanında M_a ve C_a sistem girişlerinin aldığı değerleri ifade eder, T ise örnekleme periyodudur. Sanal veri tabanı sisteminin giriş-çıkış modelini elde etmek için farklı genlik ve frekanstaki sinüzoidal giriş işaretleri sisteme uygulanmıştır.

Sanal veri tabanı sunucu sisteminin, kara-kutu sistem tanıma tekniklerini kullanarak MIMO model yapısında modellenebileceğini öne sürüyoruz. ARX220 yöntemi ile elde edilen iki kutup, iki sıfır, gecikmesiz MIMO modelin genel yapısı (4.5) ile ifade edilebilir.

MIMOARX110:

$$(C_a \rightarrow C_u), (M_a \rightarrow C_u), (W \rightarrow C_u), (C_a \rightarrow M_c), (M_a \rightarrow M_c), (W \rightarrow M_c)$$

Örneğin CPU tahsis sistem girişi ile CPU kullanım sistem çıkışı için transfer fonksiyonu en genel halde (4.5) ile gösterilebilir.

$$G_{C_a \rightarrow C_u}(z) = K_{cpu} \frac{az^2 + bz + c}{dz^2 + fz + g} \quad (4.5)$$

C_u çıkışına ilişkin SISO sistem modeli ARX110 auto regressive least square yöntemi ile sistem bir kutup, bir sıfır ve hiç gecikme olmayacak şekilde oluşturulmuştur. Sistem tanıma yöntemleri ile lineer, zamanda değişmeyen, birinci derece, tek giriş-tek çıkışlı, sanal veri tabanı sistemi, C_u sistem çıkışı için ve gecikmenin olmadığı durumda aşağıda gösterilen (4.6) fark denklemiyle formüle edilebilir

$$C_u(t) = \sum_{k=1}^l a_k(cpu) C_u(t-k) + \sum_{m=0}^0 b_m(cpu) C_a(t) \quad (4.6)$$

M_c sistem çıkışına ilişkin SISO sistem modeli, ARX111 auto regressive least square yöntemi ile sistem bir kutup, bir sıfır ve bir örnekleme zamanı gecikme olacak şekilde oluşturulmuştur. Sistem tanıma yöntemleri ile lineer, zamanda değişmeyen, birinci derece, tek giriş-tek çıkışlı, sanal veri tabanı sistemi, M_c sistem çıkışı için ve

bir örnekleme zamanı gecikme olduğu durumda aşağıda gösterilen (4.7) fark denklemiyle temsil edilebilir.

$$M_c(t) = \sum_{k=1}^I a_{k(mem)} M_c(t-k) + \sum_{m=0}^0 b_{m(mem)} M_a(t-1) \quad (4.7)$$

Deneyler, sanal veri tabanı sunucu sisteminin MIMO transfer fonksiyonu yanı sıra iki SISO sistem transfer fonksiyonu ile (4.8), (4.9), (4.10), (4.11) denklemleriyle gösterildiği gibi modellenilebileceğini göstermiştir:

SISO 1 : $(C_a, W \rightarrow C_u)$ ARX110 metodu kullanılarak

$$G_{Ca \rightarrow Cu}(z) = K_{cpu} \frac{z}{(z-b)} \quad (4.8)$$

$$G_{W \rightarrow Cu}(z) = K_W \frac{z}{(z-b)} \quad (4.9)$$

SISO 2 : $(M_a, W \rightarrow M_c)$ ARX111 metodu kullanılarak

$$G_{Ma \rightarrow Mc}(z) = K_{mem} \frac{1}{(z-c)} \quad (4.10)$$

$$G_{W \rightarrow Mc}(z) = K_W \frac{1}{(z-c)} \quad (4.11)$$

4.3.1 Çoklu-SISO modelin parametreleri

Her iki SISO model için sistem parametrelerinin boyutları şu şekilde gösterilebilir:

$A_{1 \times 1}$, $B_{1 \times 2}$, $C_{1 \times 1}$, $D_{1 \times 2}$ boyutunda matrislerdir.

CPU çıkışında gecikme olmadığı durumda SISO modelin sistem parametreleri aşağıdaki matrislerle gösterilebilir:

$$A_{\text{siso_CPU}} = [0,4286] \quad , \quad B_{\text{siso_CPU}} = [0,2425 \quad -0,0623] \quad ,$$

$$C_{\text{siso_CPU}} = [1] \quad , \quad D_{\text{siso_CPU}} = [0,56570 \quad -0,1453]$$

Bellek çıkışında bir örnekleme zamanı gecikme olduğu durumda SISO modelin sistem parametreleri aşağıdaki gibidir,

$$A_{\text{siso_mem}} = [0,9722] , B_{\text{siso_mem}} = [0,0207 \quad 0,0005] ,$$

$$C_{\text{siso_mem}} = [1] , D_{\text{siso_mem}} = [0 \quad 0].$$

CPU ve bellek çıkışında iki örnekleme zamanı gecikme görülmüştür. CPU ve bellek sistem çıkışlarında görülen iki örnekleme zamanı gecikme hesaba katıldığında sistem parametreleri aşağıdaki gibi gösterilebilir. $A_{2 \times 2}$, $B_{2 \times 2}$, $C_{1 \times 2}$, $D_{1 \times 2}$ boyutunda matrislerdir.

$$A_{\text{siso_CPU}} = \begin{bmatrix} 0 & 0 \\ 1 & 0,4286 \end{bmatrix} , B_{\text{siso_CPU}} = \begin{bmatrix} 0,5657 & -0,1453 \\ 0 & 0 \end{bmatrix} ,$$

$$C_{\text{siso_CPU}} = [0 \quad 1] , D_{\text{siso_CPU}} = [0 \quad 0]$$

$$A_{\text{siso_mem}} = \begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix} , B_{\text{siso_mem}} = \begin{bmatrix} 0,1656 & 0,004 \\ 0 & 0 \end{bmatrix} ,$$

$$C_{\text{siso_mem}} = [0 \quad 0,125] , D_{\text{siso_mem}} = [0 \quad 0].$$

4.3.2 CPU kullanımı sistem çıkışına ilişkin SISO sistemin transfer fonksiyonları, basamak girişi ve sistem kararlılığı

Gerçekleştirilen deneyler sonucunda sistemin CPU çıkışındaki gecikme süresinin iki örnekleme zamanı olduğu gözlenmiş ve bu gecikme terimi modele eklenmiştir. Gecikme olması sebebiyle ARX110 yöntemiyle bulunan modele ilişkin SISO sistemin CPU tahsis sistem girişinden (C_a), CPU kullanım sistem çıkışına (C_u) transfer fonksiyonu, ARX110 yöntemi ile bulunan ve iki örnekleme zamanı gecikmeli sistem modeline ilişkin transfer fonksiyonu,

$$T_{C_a \rightarrow C_u} = \frac{0.5657}{(z - 0,4286) z} \quad (4.12)$$

şeklinde elde edilir. Karşı düşen fark denklemi ise

$$C_u(k) = 0,4286 C_u(k - 1) + 0,5657 C_a(k - 2)$$

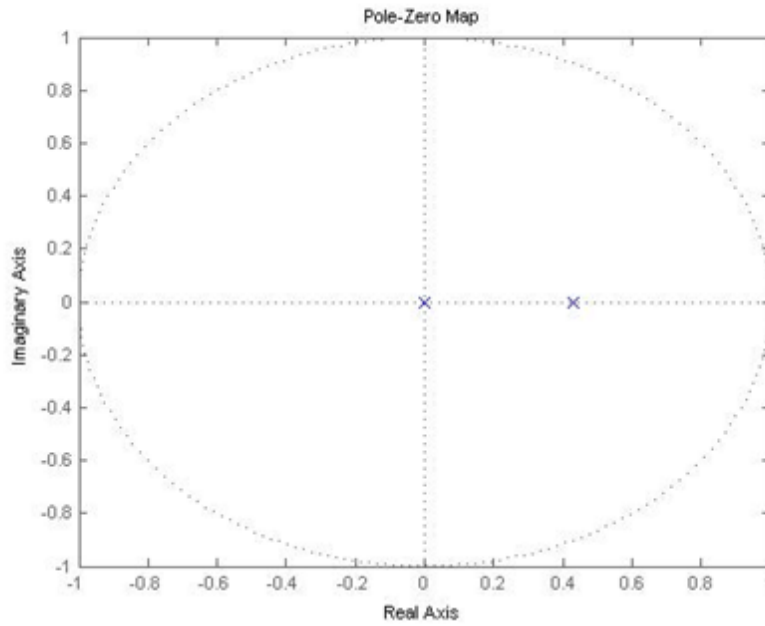
şeklinde olacaktır. SISO sistemin iş yükü -bozucu- sistem girişinden (W) CPU kullanım sistem çıkışına (C_u) transfer fonksiyonu, ARX110 yöntemi ile bulunan ve iki örnekleme zamanı gecikmeye denk olan modele ilişkin transfer fonksiyonu,

$$T_{W \rightarrow C_u} = \frac{-0,1453}{(z - 0,4286)z} \quad (4.13)$$

şeklinde elde edilir. Karşı düşen fark denklemi ise

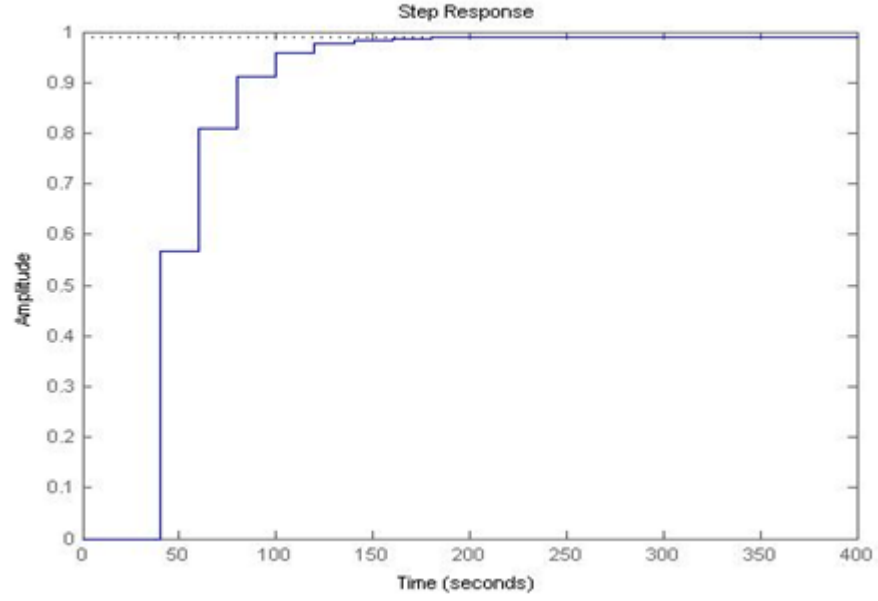
$$C_u(k) = 0,4286 C_u(k - 1) - 0,1453 W(k - 2)$$

şeklinde olacaktır. C_a sistem girişi - C_u sistem çıkışına sahip SISO sistemi için sıfır-kutup dağılımı Şekil 4.24 ile gösterilmiştir.



Şekil 4.24 : SISO sistem C_a sistem girişi - C_u sistem çıkışı için sıfır-kutup dağılımı.

SISO sistemin Şekil 4.24 ile gösterildiği gibi 0,4286 ‘da bir kutbu ve 0 ‘da diğer kutbu vardır. Transfer fonksiyonu (4.12) ile gösterilen açık çevrim sisteme ilişkin basamak cevabı Şekil 4.25 ile gösterilmiştir.



Şekil 4.25 : SISO sistem C_a sistem girişi - C_u sistem çıkışı için basamak cevabı.

4.3.3 Bellek kullanımı sistem çıkışına ilişkin SISO sistemin transfer fonksiyonları, basamak girişi ve sistem kararlılığı

Gerçekleştirilen deneyler sonucunda sistemin bellek çıkışında iki örnekleme zamanı gecikme olması sebebiyle ARX111 yöntemiyle bulunan modele ilişkin transfer fonksiyonu paydasına z çarpan olarak gelmiştir. SISO sistemin bellek tahsis sistem girişinden (M_a) bellek kullanım sistem çıkışına (M_c) transfer fonksiyonu iki örnekleme zamanı gecikme için (4.14) ile gösterilebilir.

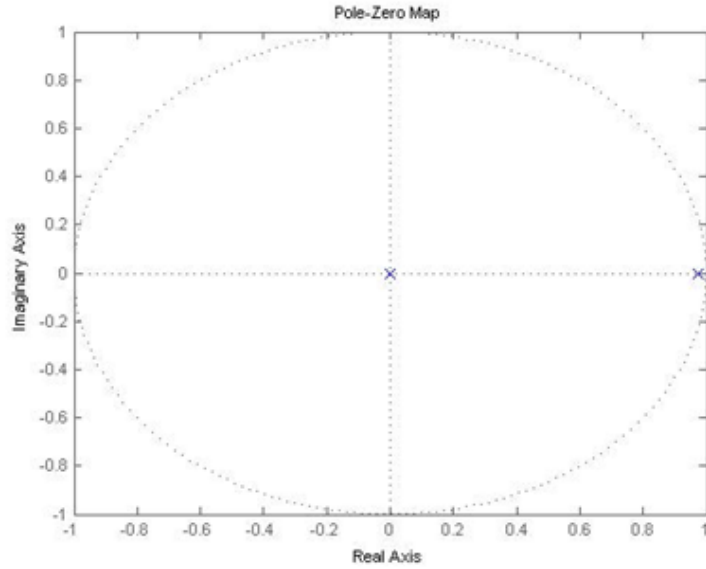
$$T_{M_a \rightarrow M_c} = \frac{0,0207}{(z - 0,9722)z} \quad (4.14)$$

Karşı düşen fark denklemi ise $M_c(k) = 0,9722 M_c(k - 1) + 0,0207 M_a(k - 2)$ şeklinde olacaktır. SISO sistemin iş yükü sistem bozucu sistem girişinden (W), bellek kullanım sistem çıkışına (M_c) transfer fonksiyonu ARX111 yöntemi ile bulunan bir örnekleme zamanı gecikmeli model ve ek olarak 1 örnekleme zamanı gecikme daha ilave edilerek iki örnekleme zamanı gecikme ile (4.15) ile elde edilir.

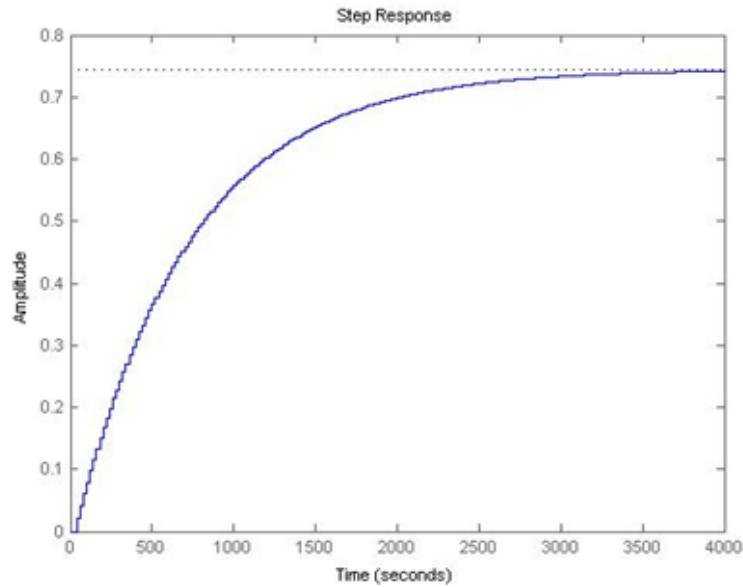
$$T_{W \rightarrow M_c} = \frac{0,0005}{(z - 0,9722)z} \quad (4.15)$$

Karşı düşen fark denklemi ise $M_c(k) = 0,9722 M_c(k - 1) + 0,0005 W(k - 2)$ şeklindedir.

SISO sistemin Şekil 4.26 ile gösterildiği gibi 0,9722 'de bir kutbu ve 0'da diğer bir kutbu vardır, sistemin sıfırı yoktur. (4.14) ile transfer fonksiyonu elde edilen sistemin basamak cevabı Şekil 4.27 ile gösterilmiştir.



Şekil4. 26 : SISO sistem M_a sistem giriş - M_c sistem çıkış kutup dağılımı.



Şekil 4.27 : SISO sistem M_a giriş - M_c çıkış sisteminin basamak cevabı.

Çoklu-SISO sistem aşağıda gösterilen matris denklemleri ile formüle edilebilir.

$$\begin{bmatrix} C_u(k) \\ M_c(k) \end{bmatrix} = \begin{bmatrix} 0,4286 & 0 \\ 0 & 0,9722 \end{bmatrix} \begin{bmatrix} C_u(k-1) \\ M_c(k-1) \end{bmatrix} + \begin{bmatrix} 0,5657 & 0 \\ 0 & 0,0207 \end{bmatrix} \begin{bmatrix} C_a(k-2) \\ M_a(k-2) \end{bmatrix} \\ + \begin{bmatrix} -0,1453 & 0 \\ 0 & 0,0005 \end{bmatrix} \begin{bmatrix} W(k-2) \\ W(k-2) \end{bmatrix}$$

4.3.4 MIMO modelinin parametreleri

Gecikmesiz durum için MIMO modele ilişkin sistem parametreleri $A_{2 \times 2}$, $B_{2 \times 3}$, $C_{2 \times 2}$, $D_{2 \times 3}$ matrisler olarak gösterilebilir ve MIMO modelin sistem parametreleri aşağıdaki gibi elde edilmiştir.

$$A_{mimo} = \begin{bmatrix} 0,6452 & 0,05443 \\ 0,007703 & 0,9456 \end{bmatrix}, \quad B_{mimo} = \begin{bmatrix} -3,741 & 12,96 & 0,06088 \\ 0,06088 & -0,1949 & -0,004366 \end{bmatrix} \\ , \quad C_{mimo} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_{mimo} = \begin{bmatrix} -5,808 & 20,12 & 0,09481 \\ 0,1142 & -0,37 & -0,005389 \end{bmatrix}$$

4.3.5 MIMO modelin transfer fonksiyonu gösterilimi

ARX220 yöntemi ile elde edilmiş ve iki kutup, iki sıfır, gecikmesiz MIMO sistem modeline ilişkin transfer fonksiyonları bu kısımda gösterilmektedir. MIMO sistemin CPU tahsis sistem girişinden (C_a) CPU kullanım sistem çıkışına (C_u) transfer fonksiyonu,

$$T_{C_a \rightarrow C_u} = \frac{-5,808 z^2 + 5,499 z}{z^2 - 1,591 z + 0,6097} \quad (4.16)$$

ve karşı düşen fark denklemi ise,

$$C_u(k) = 1,591 C_u(k-1) - 0,6097 C_u(k-2) - 5,808 C_a(k) + 5,499 C_a(k-1)$$

şeklinde elde edilir.

MIMO sistemin CPU tahsis sistem girişinden (C_a) bellek tüketim sistem çıkışına

(M_c) transfer fonksiyonu,

$$T_{C_a \rightarrow M_c} = \frac{0,1142 z^2 - 0,1184 z}{z^2 - 1,591 z + 0,6097} \quad (4.17)$$

ve karşı düşen fark denklemi ise,

$$M_c(k) = 1,591 M_c(k-1) - 0,6097 M_c(k-2) + 0,114 C_a(k) - 0,118 C_a(k-1)$$

şeklinde elde edilir.

MIMO sistemin bellek tahsis sistem girişinden (M_a) CPU kullanım sistem çıkışına (C_u) transfer fonksiyonu,

$$T_{M_a \rightarrow C_u} = \frac{20,12 z^2 - 19,05 z}{z^2 - 1,591 z + 0,6097} \quad (4.18)$$

ve karşı düşen fark denklemi ise,

$$C_u(k) = 1,591 C_u(k-1) - 0,6097 C_u(k-2) + 20,12 M_a(k) - 19,05 M_a(k-1)$$

şeklinde elde edilir. MIMO sistemin bellek tahsis sistem girişinden (M_a) bellek tüketim sistem çıkışına (M_c) transfer fonksiyonu,

$$T_{M_a \rightarrow M_c} = \frac{-0,37 z^2 + 0,3937 z}{z^2 - 1,591 z + 0,6097} \quad (4.19)$$

ve karşı düşen fark denklemi ise,

$$M_c(k) = 1,591 M_c(k-1) - 0,6097 M_c(k-2) - 0,37 M_a(k) + 0,3937 M_a(k-1)$$

şeklinde elde edilir. MIMO sistemin iş yükü bozucu sistem girişinden (W) CPU kullanım sistem çıkışına (C_u) transfer fonksiyonu,

$$T_{W \rightarrow C_u} = \frac{0,09481 z^2 - 0,08995 z}{z^2 - 1,591 z + 0,6097} \quad (4.20)$$

ve fark denklemi aşağıdaki gibi elde edilir.

$$C_u(k) = 1,591 C_u(k-1) - 0,6097 C_u(k-2) + 0,09481 W(k) - 0,08995 W(k-1)$$

MIMO sistemin iş yükü bozucu sistem girişinden (W) bellek tüketim sistem çıkışına (M_c) transfer fonksiyonu,

$$T_{W \rightarrow M_c} = \frac{-0,005389 z^2 + 0,004207 z}{z^2 - 1,591 z + 0,6097} \quad (4.21)$$

ve fark denklemi aşağıdaki gibi elde edilir.

$$M_c(k) = 1,591 M_c(k-1) - 0,6097 M_c(k-2) - 0,005389 W(k) + 0,004207 W(k-1).$$

MIMO sistem aşağıdaki gibi çok giriş çok çıkışlı fark denklemleri olarak da ifade edilebilir.

$$\begin{aligned} \begin{bmatrix} C_u(k) \\ M_c(k) \end{bmatrix} = & \begin{bmatrix} 1,591 & -0,6097 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} C_u(k-1) \\ C_u(k-2) \end{bmatrix} \\ & + \begin{bmatrix} 0 & 0 \\ 1,591 & -0,6097 \end{bmatrix} \begin{bmatrix} M_c(k-1) \\ M_c(k-2) \end{bmatrix} \\ & + \begin{bmatrix} -5,808 & 5,499 \\ 0,1142 & -0,1184 \end{bmatrix} \begin{bmatrix} C_a(k) \\ C_a(k-1) \end{bmatrix} \\ & + \begin{bmatrix} 20,12 & -19,05 \\ -0,37 & 0,3937 \end{bmatrix} \begin{bmatrix} M_a(k) \\ M_a(k-1) \end{bmatrix} \\ & + \begin{bmatrix} 0,09481 & -0,08995 \\ -0,005389 & 0,004207 \end{bmatrix} \begin{bmatrix} W(k) \\ W(k-1) \end{bmatrix} \end{aligned}$$

MIMO sistemde iki gecikme olduğu durumda sistemin transfer fonksiyonunun paydasında z^2 yer alır. Sıfır noktasındaki sıfır, sıfır noktasındaki tek kutbun etkisini ortadan kaldırır ve MIMO sistemin mertebesi sistem çıkışlarında iki örnekleme zamanı gecikme olduğu durumda üç olur. Neticede iki örnekleme zamanı gecikmeli CPU ve bellek SISO sistemlerin mertebesi iki, iki örnekleme zamanı gecikmeli MIMO sistemin derecesi üç olur. Kontrolör tasarımında, kontrol etmedeki kolaylık ve sistem mertebesinin daha az olması sebebiyle SISO sistem modelleri kullanılacaktır.

4.4 Veri Tabanı Sunucu Sistem Modeline İlişkin Elde Edilen Sonuçlar

İş yükü veri tabanı sistemi üzerinde kayıt ekleme, silme, değiştirme yaparak veya veri tabanından kayıt okuma yaparak yaratılmıştır. Bu dört farklı işlem için belirli ağırlıklar verilerek deneyler yapılmıştır. Örneğin 10 işlemten 7 adedi kayıt ekleme 3 tanesi kayıt değiştirme olarak belirlenebilmiştir. İş yükü arttıkça CPU kullanımı ve bellek tüketimi artış göstermiştir.

CPU tahsis birimi megahertz (MHz), bellek kaynak tahsisi birimi megabayttır (MB). İş yükü birimi bir saniyede veri tabanına gelen transaction sayısı (tnx/sec) ya da ağ üzerinde veri tabanına birim zamanda gelen veri miktarı saniyedeki kilo bit (Kbps: Kilobit per second) ile ölçülmüştür.

CPU ve bellek kaynak tahsisi 20 saniye aralıklarla VMware apilerini çağıran C# programı vasıtasıyla otomatik hale getirilmiştir. Sistem dinamiğini tam olarak ortaya koymak ve doğru modeli elde etmek için CPU ve bellek kaynak tahsisi farklı frekans ve genlikte sinüs eğrileri olarak değiştirilerek sistem çıkışları olan CPU ve bellek kullanımı gözlemlenmiştir. Yüzü aşkın deneyde elde edilen sonuçlara göre sistem dinamiğini yansıtan en uygun modeller belirlenmeye çalışılmıştır.

Farklı sistem tanıma yöntemleri denenmiş fakat ARX sistem tanıma yöntemi sistem modeli elde etmede iyi sonuçlar verdiğinden ve basit model yapısı sebebiyle tercih edilmiştir. Lineer olmayan sistem modellerin başarımı lineer modellerden daha iyi olmadığı için, sistem modeli olarak lineer olmayan bir model yerine lineer model seçilmiştir.

CPU çıkışı hızlı bir dinamik göstermektedir. Giriş değiştiğinde CPU çıkışındaki değişim de hızlı olmaktadır. Hâlbuki bellek dinamiği yavaştır. Sistem çıkışı olan bellek tüketimi değişimlere diğer sistem çıkışı olan CPU kullanımına oranla daha yavaş yanıt vermektedir.

MIMO sistem için rastgele değişen ve yeterli büyüklükteki iş yükü altında veri tabanı sanal sunucu makinası model CPU çıkışı ile gerçek çıkışın başarımı ortalama olarak %70-80'dir. MIMO sistem için rastgele değişen ve yeterli büyüklükteki iş yükü altında, veri tabanı sanal sunucu makinası model bellek çıkışı ile gerçek çıkışın başarımı ortalama olarak %55-60 seviyesindedir. Çoklu SISO model başarımlarının MIMO model başarımı kadar iyi olduğu gözlemlenmiştir.

Tez çalışması kapsamında gerçekleştirilen deneyler, MIMO ve çoklu SISO modelleri gerçek sistemi temsil etmekte oldukça iyi bir başarımlar göstermişlerdir. MIMO sistem ve çoklu SISO sistem modelleri arasında model derecesinin düşük olması sebebiyle çoklu SISO sistem modellerine odaklanılmıştır ve bu modellerin sanal veri tabanı sunucusunun geri besleme kontrolünde kullanılabilir olduğuna karar verilmiştir.

5. KONTROLÖR TASARIMI: PI KONTROLÖR

PI kontrolör, basamak bozucu için sürekli hal hatasını yok eden ve değişen koşullara karşı dayanıklı ve yaygın olarak kullanılan bir kontrolördür. Bu nedenle, sistemin kontrolü için önce PI türü bir kontrolör tasarlanmasının uygun olacağı düşünülmüştür. PI tipi bir kontrolör (5.1) ile gösterilen kontrol kuralı ile ifade edilir,

$$u(k) = K_P \cdot e(k) + K_I \sum_{j=1}^{k-1} e(j) \quad (5.1)$$

Burada, $u(k)$ kontrol işaretidir $r(k)$ çıkış büyüklüklerinin ulaşması istenen referans değerleri olmak üzere $e(k) = r(k) - y(k)$ hata işaretidir. PI kontrolör (5.2) ile gösterildiği gibi K_P ve K_I oran kazancı ve integral kazancı olmak üzere iki parametreye sahiptir. K_P oransal kazanç, K_I integral kazancıdır. PI Kontrolörün z-tanım bölgesinde transfer fonksiyonu;

$$U(z) = K_P + \frac{K_I z}{(z-1)} E(z) = (K_P + K_I) \frac{(z - \frac{K_P}{K_P + K_I})}{(z-1)} E(z) \quad (5.2)$$

şeklinde elde edilir. CPU çıkışına ilişkin SISO sistem bu kontrolör kullanılarak birim geri besleme ile kapatılırsa, kapalı çevrim transfer fonksiyonu K_P ve K_I kazanç parametrelerine bağlı olarak en genel halde (5.3) ile gösterilebilir.

$$T(z) = \frac{Y(z)}{R(z)} = \frac{G_c(z) G(z)}{1 + G_c(z) G(z)} = \frac{(K_P + K_I) \frac{(z - \frac{K_P}{K_P + K_I})}{(z-1)} (\frac{0,5657}{(z-0,4286)z})}{1 + [(K_P + K_I) \frac{(z - \frac{K_P}{K_P + K_I})}{(z-1)} (\frac{0,5657}{(z-0,4286)z})]} = \frac{0,5657 (K_P + K_I) \left(z - \frac{K_P}{K_P + K_I} \right)}{z(z-1)(z-0,4286) + [0,5657 (K_P + K_I) \left(z - \frac{K_P}{K_P + K_I} \right)]} \quad (5.3)$$

$$T(1) = \frac{0,5657 (K_P + K_I) \left(1 - \frac{K_P}{K_P + K_I}\right)}{0 + [0,5657 (K_P + K_I) \left(1 - \frac{K_P}{K_P + K_I}\right)]} = 1 \text{ olduğundan birim basamak girişi için}$$

sistemin sürekli hal hatası K_P , K_I -PI kontrolör kazanç- parametrelerinden bağımsız olarak sıfır olacaktır. Aynı sonuç bellek çıkışına ilişkin SISO sistem için de geçerlidir. SISO sistem için $e(k)$, $u(k)$, K_P , K_I büyüklükleri tek boyutludur. MIMO sistem için bu büyüklükler matris olarak ifade edilir. PI kontrolör tasarlamak için K_P ve K_I oran ve integral kazançları istenilen kontrol şartlarını sağlayacak biçimde seçilmelidir. Kapalı çevrimli sistemin sağlaması gereken şartlar; sürekli hal hatasının olmaması, kısa bir yerleşme zamanı, aşımın mümkün olduğunca az ya da hiç olmaması olarak düşünülebilir.

5.1 CPU Kullanımı sistemine ilişkin SISO model temelli Kontrolör Tasarımı

MIMO sistemin derecesi daha yüksek olduğundan daha basit bir sistem olan çoklu SISO sistem için kontrolör tasarımı yapılacaktır. CPU tahsis sistem girişi ve CPU kullanım sistem çıkışı olan SISO sistem için sistem transfer fonksiyonu (5.4) ile ifade edilebilir.

$$G(z) = K_{cpu} \frac{z}{(z - b)} = \frac{0,5657 z}{(z - 0,4286)} \quad (5.4)$$

SISO sistemin CPU tahsis girişinde bir değişiklik yapıldığı zaman sistem çıkışında iki örnekleme zamanı gecikme görülmüştür. Bu durumda sistemin transfer fonksiyonu (5.5) ile formüle edilebilir.

$$G(z)/z^2 = \frac{0,5657 z}{(z-0,4286) z^2} = \frac{0,5657}{(z-0,4286) z} \quad (5.5)$$

Basamak girişine karşı sistem cevabı kritik sönümlü olacak şekilde, örnekleme zamanı 20 saniye için PI kontrolörünün K_P , K_I kazançları 0,1885 ve 0,2534 seçilmiştir. Sonuç olarak tasarlanan kontrolörün transfer fonksiyonu $G_c(z)$,

$$G_c(z) = \frac{0,4419 (z - 0,4286)}{(z - 1)} \quad (5.6)$$

ve karşı düşen fark denklemi ise,

$$u(k) = u(k-1) + 0,4419 e(k) - 0,1885 e(k-1) \quad (5.7)$$

şeklinde elde edilir. Kapalı çevrim transfer fonksiyonu $T(z)$, $R(z)$ çıkış , $Y(z)$ giriş olmak üzere (5.8) ile verilir

$$\begin{aligned} T(z) &= \frac{Y(z)}{R(z)} = \frac{G_c(z) G(z)}{1 + G_c(z) G(z)} \\ &= \frac{\left(\frac{0,4419 (z - 0,4286)}{(z - 1)} \right) \left(\frac{0,5657 z}{(z - 0,4286) z^2} \right)}{1 + \left(\frac{0,4419 (z - 0,4286)}{(z - 1)} \right) \left(\frac{0,5657 z}{(z - 0,4286) z^2} \right)} \\ &= \frac{0,25}{(z - 0,5)^2} \end{aligned} \quad (5.8)$$

Karakteristik denklem : $1 + G_c(z) G(z) = 0$

$$1 + \left(\frac{0,4419 (z - 0,4286)}{(z - 1)} \right) \left(\frac{0,5657 z}{(z - 0,4286) z^2} \right) = 0, (z - 0,5)^2 = 0 \quad (5.9)$$

$z_{1,2}$ 0,5 'de (5.9) ile gösterilen karakteristik denklemin iki kökü vardır, bu kökler kapalı çevrim sistemin kutuplarının z tanım bölgesinde bulunduğu yerleri gösterir. Kapalı çevrim sistemin kutupları ya da karakteristik denklemin kökleri z tanım bölgesinde birim çemberin içerisinde olduğundan sistem kararlıdır. Sistem kararlı olduğu **son değer teoremi** gereği sistemin sürekli halde cevabı hesaplanabilir [40].

$$\lim_{k \rightarrow \infty} x(k) = \lim_{z \rightarrow 1} [(1 - z^{-1}) X(z)] \quad , \text{ Birim basamak girişi } R(z) = \frac{z}{(z - 1)}$$

Birim basamak girişi için sistemi sürekli hal sürekli hal cevabı:

$$\begin{aligned} \lim_{z \rightarrow 1} [(1 - z^{-1}) Y(z)] \\ &= \lim_{z \rightarrow 1} [(1 - z^{-1}) T(z) R(z)] \\ &= \lim_{z \rightarrow 1} \left[(1 - z^{-1}) \frac{0,25}{(z - 0,5)^2} \frac{z}{(z - 1)} \right] = 1 \end{aligned}$$

$$y(\infty) = 1, e_{ss} = r - y(\infty) = 1 - 1 = 0$$

Büyükluğu A olarak değişen farklı basamak girişleri için $R(z) = \frac{A z}{(z - 1)}$

$$\begin{aligned}
\lim_{z \rightarrow 1} [(1 - z^{-1}) Y(z)] &= \lim_{z \rightarrow 1} [(1 - z^{-1}) T(z) R(z)] \\
&= \lim_{z \rightarrow 1} \left[(1 - z^{-1}) \frac{0,25}{(z - 0,5)^2} \frac{A \cdot z}{(z - 1)} \right] = A
\end{aligned}$$

$y(\infty) = 1$, $e_{ss} = r - y(\infty) = A - A = 0$ PI kontrolör ile kontrol edilen kapalı çevrim sistem çıkışında farklı büyüklükteki basamak girişleri için sürekli hal hatası oluşmaz.

5.2 Bellek Kullanım sistemine ilişkin SISO model temelli Kontrolör Tasarımı

Bellek tahsis sistem girişi ve bellek tüketim sistem çıkışı SISO sistemi için transfer fonksiyonu iki örnekleme zamanı gecikme için (5.10) ile gösterilmiştir.

$$G(z)/z = K_{mem} \frac{1}{(z - c)z} = \frac{0,0207}{(z - 0,9722)z} \quad (5.10)$$

Basamak girişine karşı sistem cevabı kritik sönümlü olacak şekilde, örnekleme zamanı 20 saniye için PI kontrolörünün K_p ve K_I kazançları 1,361 ve 0,039, seçilmiştir. Sonuç olarak tasarlanan kontrolörün transfer fonksiyonu $G_c(z)$,

$$G_c(z) = \frac{1,4 (z - 0,9722)}{(z - 1)} \quad (5.11)$$

ve karşı düşen fark denklemi ise,

$$u(k) = u(k-1) + 1,4 e(k) - 1,361 e(k-1) \quad (5.12)$$

şeklinde elde edilir. Kapalı çevrim transfer fonksiyonu $T(z)$, $R(z)$ çıkış, $Y(z)$ giriş olmak üzere (5.13) ile gösterilebilir.

$$\begin{aligned}
T(z) = \frac{Y(z)}{R(z)} &= \frac{G_c(z) G(z)}{1 + G_c(z) G(z)} \\
&= \frac{\left(\frac{1,4 (z - 0,9722)}{(z - 1)} \right) \left(\frac{0,0207}{(z - 0,9722)z} \right)}{1 + \left(\frac{1,4 (z - 0,9722)}{(z - 1)} \right) \left(\frac{0,0207}{(z - 0,9722)z} \right)} \\
&= \frac{0,0289}{(z^2 - z + 0,0289)}
\end{aligned} \quad (5.13)$$

Karakteristik denklem : $1 + G_c(z) G(z) = 0$

$$1 + \left(\frac{1,4 (z-0,9722)}{(z-1)} \right) \left(\frac{0,0207}{(z-0,9722) z} \right) = 0 \quad (5.14)$$

$z_{1,2}$, 0,9702 ve 0,0298 ‘de (5.14) ile gösterilen karakteristik denklemin iki kökü vardır, bu kökler kapalı çevrim sistemin kutuplarının z tanım bölgesinde bulunduğu yerleri gösterir. Kapalı çevrim sistemin kutupları ya da karakteristik denklemin kökleri z tanım bölgesinde birim çemberin içerisinde olduğundan sistem kararlıdır. Sistem kararlı olduğu **son değer teoremi** gereği sistemin sürekli halde cevabı hesaplanabilir.

$$\lim_{k \rightarrow \infty} x(k) = \lim_{z \rightarrow 1} [(1 - z^{-1}) X(z)] \quad , \text{ Birim basamak girişi } R(z) = \frac{z}{(z-1)}$$

Birim basamak girişi için sistemi sürekli hal sürekli hal cevabı:

$$\begin{aligned} \lim_{z \rightarrow 1} [(1 - z^{-1}) Y(z)] &= \lim_{z \rightarrow 1} [(1 - z^{-1}) T(z) R(z)] \\ &= \lim_{z \rightarrow 1} \left[(1 - z^{-1}) \frac{0,0289}{(z^2 - z + 0,0289)} \frac{z}{(z-1)} \right] = 1 \end{aligned}$$

$$y(\infty) = 1, \quad e_{ss} = r - y(\infty) = 1 - 1 = 0$$

Büyükliği A olarak değişen farklı basamak girişleri için $R(z) = \frac{A z}{(z-1)}$

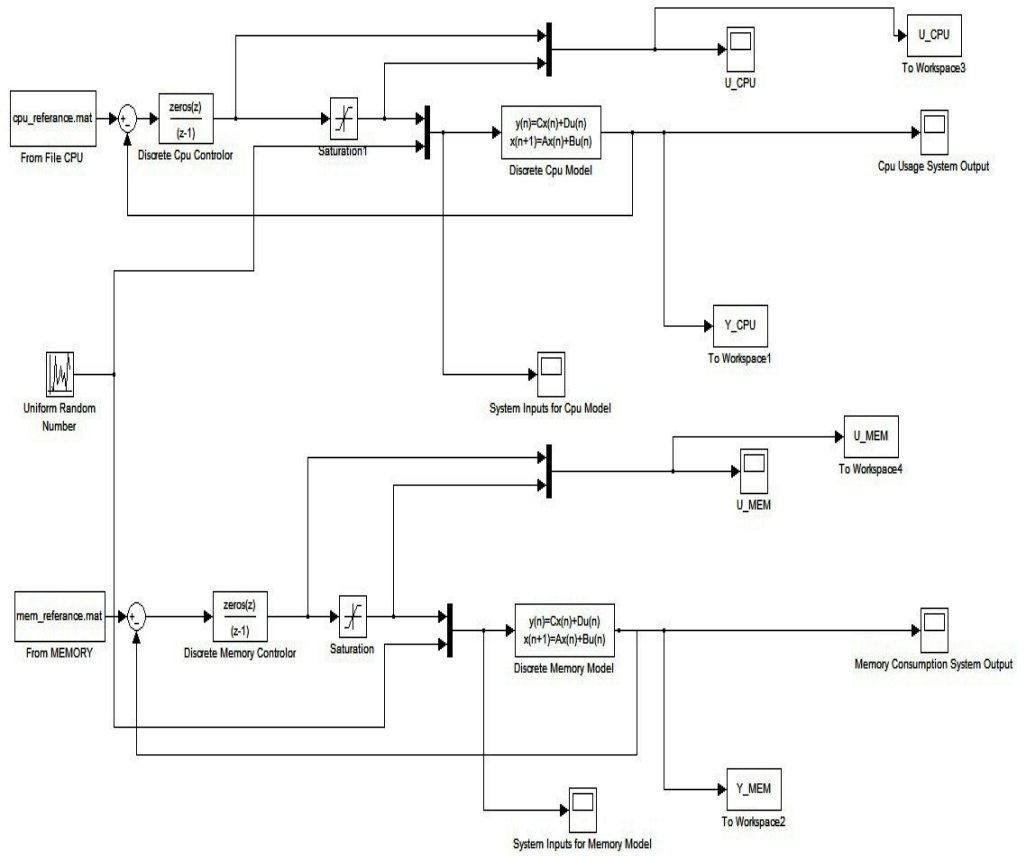
$$\begin{aligned} \lim_{z \rightarrow 1} [(1 - z^{-1}) Y(z)] &= \lim_{z \rightarrow 1} [(1 - z^{-1}) T(z) R(z)] \\ &= \lim_{z \rightarrow 1} \left[(1 - z^{-1}) \frac{0,0289}{(z^2 - z + 0,0289)} \frac{A \cdot z}{(z-1)} \right] = A \end{aligned}$$

$y(\infty) = 1, \quad e_{ss} = r - y(\infty) = A - A = 0$ PI kontrolör ile kontrol edilen kapalı çevrim sistem çıkışında farklı büyüklükteki basamak girişleri için sürekli hal hatası oluşmaz.

5.3 Test Ortamının Gerçeklemesi, PI Kontrolör İle Yönetilen Sistem Deney Sonuçları

Deneyler için emülatör sistem ve emülatörü simüle eden simülatör sistem oluşturulmuştur. Simülatör, Şekil 5.1 ile gösterildiği MATLAB simülink ortamında,

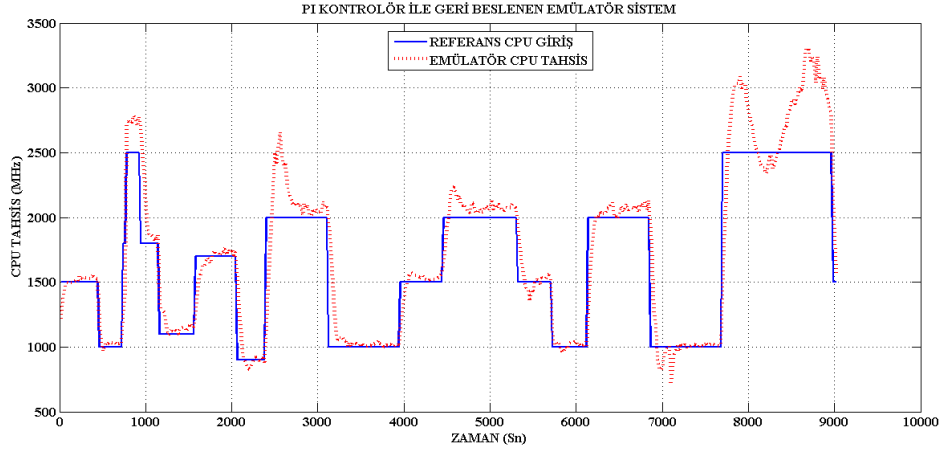
emülatör ise VMware sanallaştırma ortamında oluşturulmuştur. Sentetik iş yükü yaratıcı araç vasıtasıyla veri tabanı sanal sunucusu (DBVM) üzerinde iş yükü yaratılmıştır. Emülatörde bir veri merkezi kurulmuştur. Bir sunucu ve bu sunucu üzerinde iki sanal makina veri merkezinde oluşturulmuştur. Sanal makinalar veri tabanı sunucusu olarak işlev görmektedir. Her bir DBVM üzerine SQL Server 2008 R2 kurulmuş ve Win7 makinası üzerinde 'DS', WinXP makinası üzerinde 'DS2' veri tabanı oluşturulmuştur. Tablo, izlek, fonksiyon ve indeks gibi bazı veri tabanı objeleri her iki veri tabanı üzerinde oluşturulmuştur. Ayrıca kayıt ekleme, güncelleme, silme, sorgulama ve izlek çalıştırma işlemleri için veri tabanı kullanıcıları yaratılmış ve bu kullanıcılara gerekli yetkiler verilmiştir. Kullanıcılardan gelen istekler DBVM üzerinde iş yükü yaratır. DBVM CPU kullanımı ve bellek tüketimi iş yükünün büyüklüğüne göre değişmektedir. İş yükünün büyüklüğü, iş yükü ile ilgili kullanıcı ara yüzündeki parametreler değiştirilerek ayarlanabilmektedir. CPU kullanımı ve bellek tüketimi sistem çıkış değişkenlerinin şu anki ve geçmiş değerlerini ölçmek üzere algılayıcılar tasarlanmıştır. Sistem değişkenlerine ilişkin performans verisini, Hipervizörün performans yöneticisinden elde eden yazılım fonksiyonları bu tez kapsamında gerçekleştirilmiştir. Geliştirmiş olduğumuz yazılım izlekleri, hipervizör API çağrıları yaparak CPU kullanımı ve bellek tüketimi sistem çıkış değişkenlerine ilişkin performans metriklerine erişir. Hipervizör API çağrıları ile sanal makinalara CPU ve bellek kaynak tahsisi yapan sürücüler de bu tez çalışması kapsamında tasarlanmıştır.



Şekil 5.1 : Sanal veri tabanı sunucusu çoklu SISO sistemlerinin geri besleme ile kontrolüne ilişkin simulink şeması.

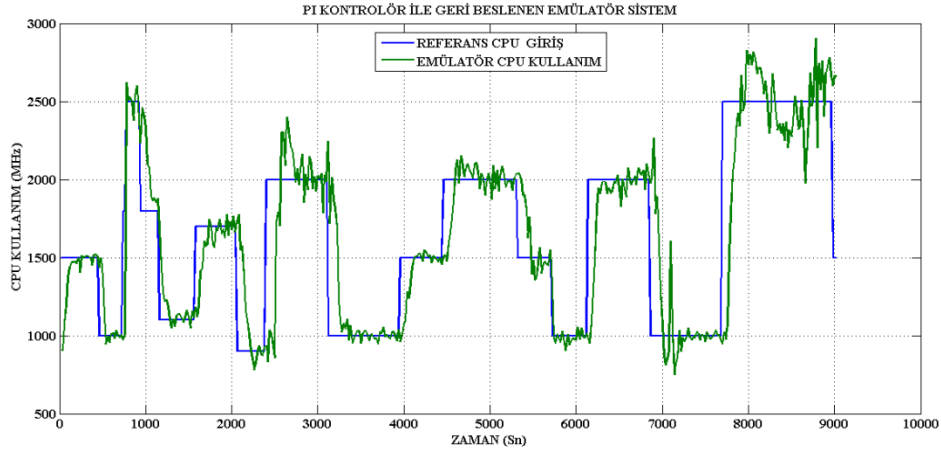
Emülatörden elde edilen referans değerleri ve sistem giriş-çıkış verisi, MATLAB simülatörde analiz edilmek ve doğrulanmak üzere MATLAB çalışma ortamına aktarılmıştır. İlk olarak MATLAB SIT kullanılarak çoklu SISO modelleri elde edilmiş ve bu modeller MATLAB simülink ortamına aktarılmıştır. İkinci olarak, kararlılık-doğruluk-yerleşme zamanı-aşım gibi kıstasları sağlayan PI kontrolörler simülatörde tasarlanmıştır. Üçüncü bir iş adımı olarak MATLAB simülinkte Şekil 5.1 ile gösterildiği gibi sanal veri tabanı sistemi, kontrolör ve diğer kontrol elemanları geri besleme çevrimini oluşturacak şekilde birbirlerine bağlanmıştır. Son olarak simülatör sistem, emülatöre uygulanan aynı büyüklük ve frekanstaki CPU kullanım referans girişleri, bellek tüketim referans girişleri ve yine aynı büyüklükte zamanda değişen iş yükü ile çalıştırılmıştır. Hem emülatör hem de simülatörden elde edilen sistem giriş-çıkış verisi aynı grafik üzerinde karşılaştırılmıştır. Emülatörde PI kontrolör, yazılım olarak gerçekleştirilmiştir. Ayrıca emülatör ortamında CPU ve bellek referans değerleri gerçek zamanlı olarak kullanıcı ara yüzü vasıtasıyla belirlenmiştir.

Hem emülatör hem de simülatörde deneyler gerçekleştirilmiştir. Her iki ortamdan elde edilen ilgili veriler aynı ya da farklı grafiklerle karşılaştırmalı olarak gösterilmiştir. Emülatörden elde edilen referans giriş, CPU tahsis verisi Şekil 5.2 ile gösterilmiştir.



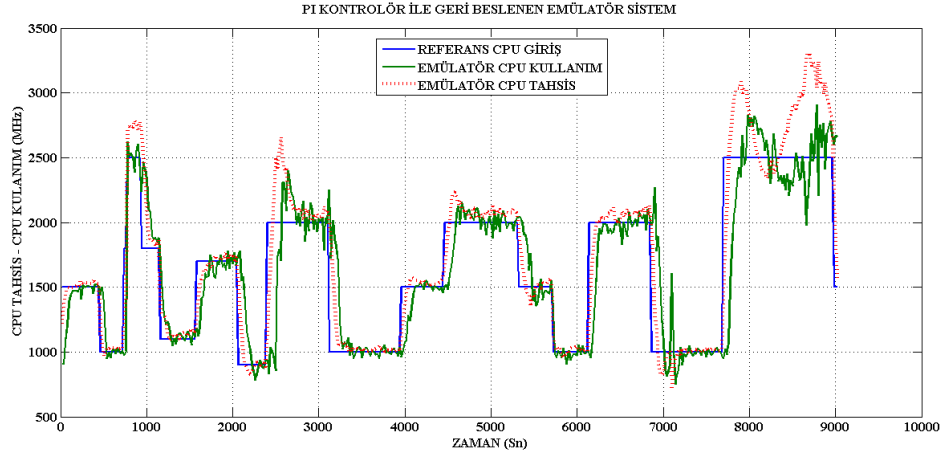
Şekil 5.2 : CPU SISO sistem için emülatör ortamında referans değer ve CPU tahsis gösterimi.

Emülatörden elde edilen referans giriş, CPU kullanım verisi Şekil 5.3 ile grafikte gösterilmiştir.



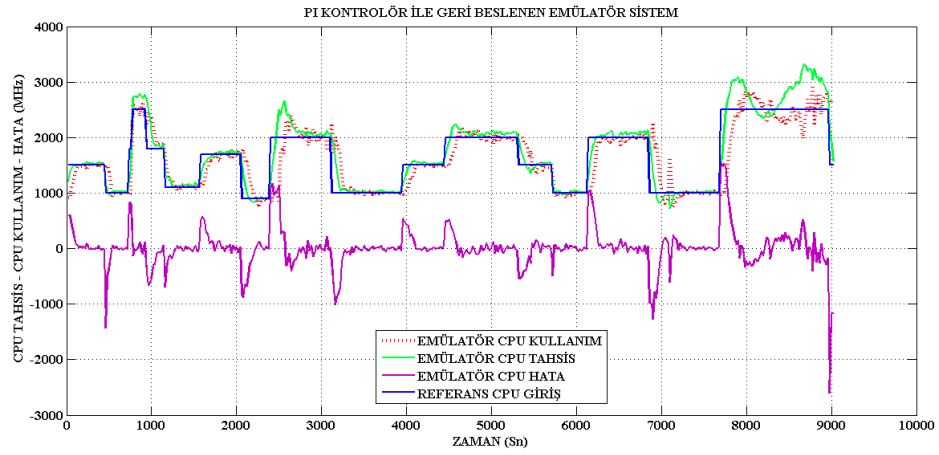
Şekil 5.3 : CPU SISO sistem için emülatör ortamında referans değer ve CPU kullanım gösterimi.

Emülatörden elde edilen referans giriş, CPU tahsis ve CPU kullanım verisi Şekil 5.4 ile grafik olarak gösterilmiştir.



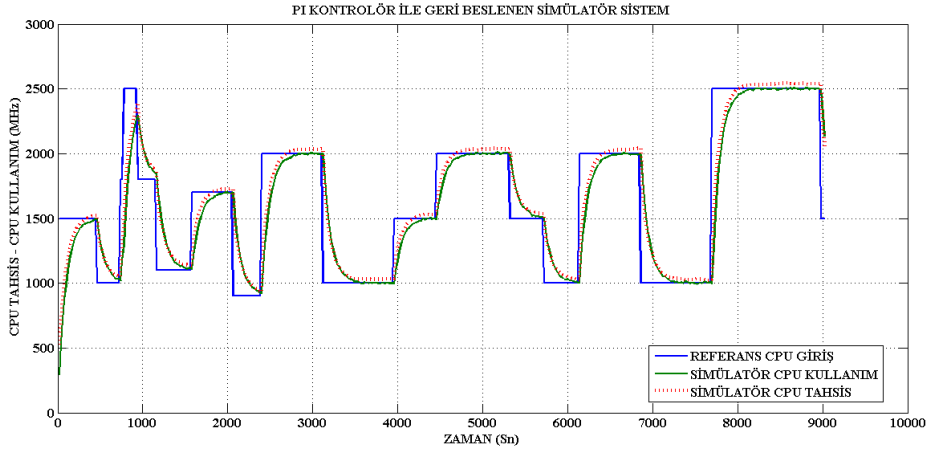
Şekil 5.4 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis, CPU kullanım gösterimi.

Emülatörden elde edilen referans giriş, CPU tahsis, CPU kullanım ve oluşan hata işareti Şekil 5.5’deki grafikte verilmiştir.



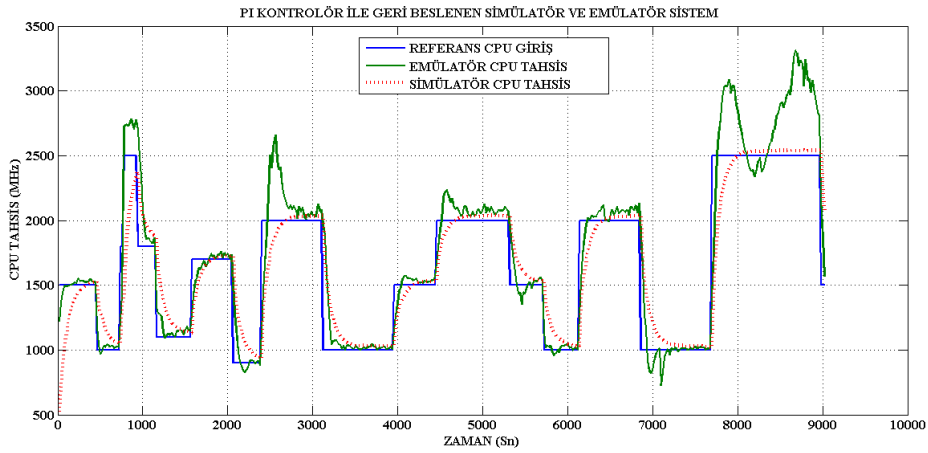
Şekil 5.5 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis, CPU kullanım ve referans ile sistem çıkışı olan CPU kullanımı arasındaki farktan kaynaklanan hata bilgilerinin gösterimi.

Simülatörden elde edilen referans giriş, CPU tahsis, CPU kullanım verisi Şekil 5.6 ile gösterilmiştir.



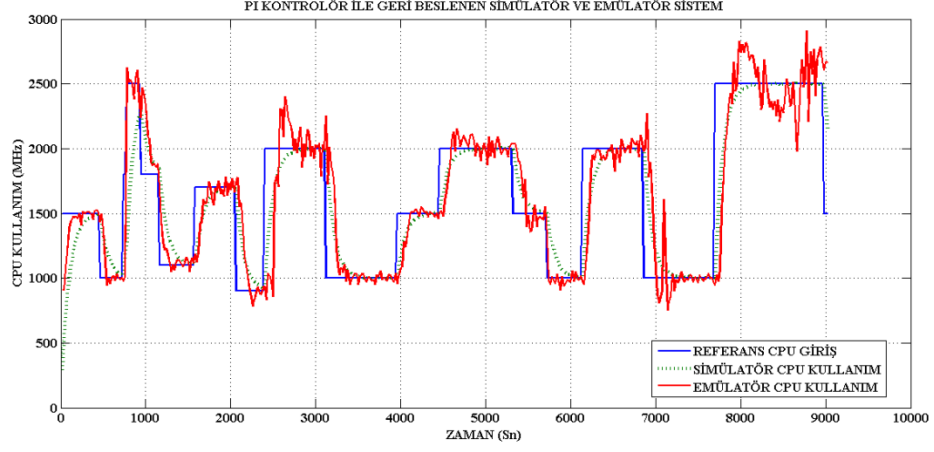
Şekil 5.6 : CPU SISO sistem için simülâtör ortamında referans değêr, CPU tahsis, CPU kullanım gösterimi.

Simülâtörden ve emülâtörden elde edilen referans giriř, CPU tahsis verisi Şekil 5.7’deki grafik ile gösterilmiřtir.



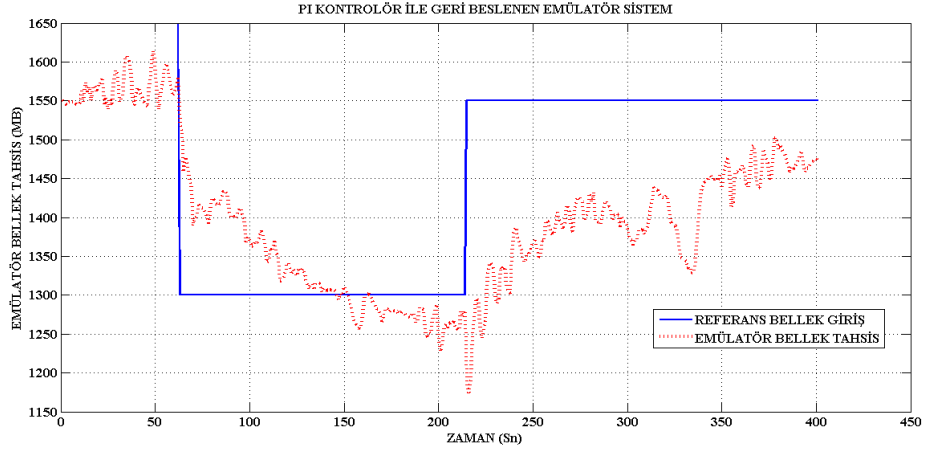
Şekil 5.7 : CPU SISO sistem için simülâtör ve emülâtör ortamları için referans değêr, CPU tahsis gösterimi.

Simülâtörden ve emülâtörden elde edilen referans giriř, CPU kullanım verisi Şekil 5.8’deki grafik ile gösterilmiřtir.



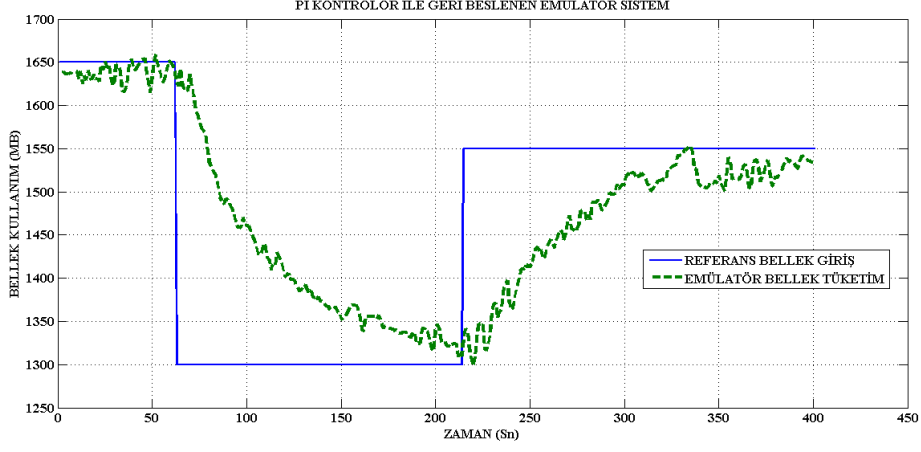
Şekil 5.8 : CPU SISO sistem için simülör ve emülör ortamlarından elde edilen referans değeri, CPU kullanım verisi gösterimi.

Emülörden elde edilen referans girişı ve bellek tahsis verisi -kontrol girişı- Şekil 5.9 ile gösterilmiştir.



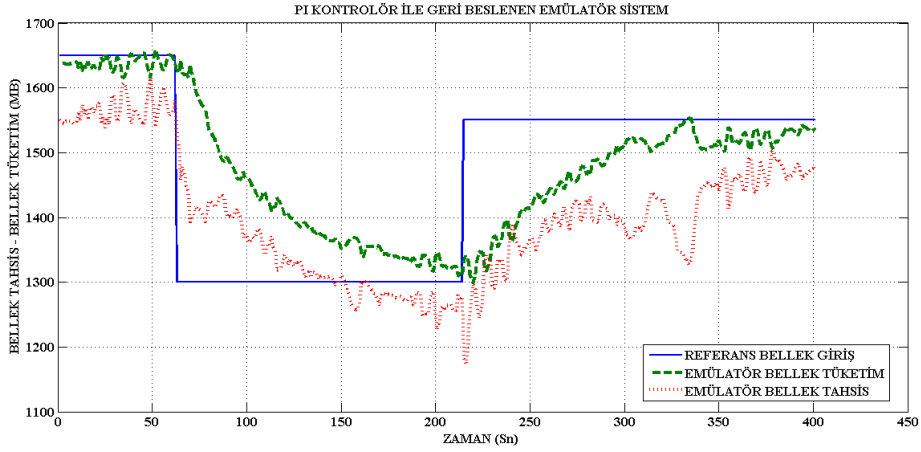
Şekil 5.9 : Bellek SISO sistem için emülör ortamında referans değeri, bellek tahsis -kontrol girişı- gösterimi.

Emülörden elde edilen referans girişı, bellek tüketim verisi -sistem çıkışı- Şekil 5.10 ile gösterilmiştir.



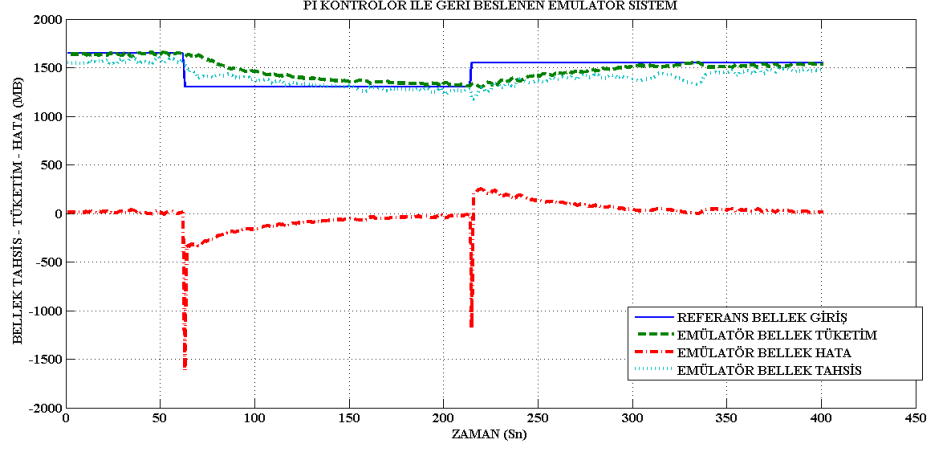
Şekil 5.10 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tüketim -sistem çıkışı- gösterimi.

Emülatörden elde edilen referans giriş, bellek tahsis ve bellek tüketim verisi Şekil 5.11 ile gösterilmiştir.



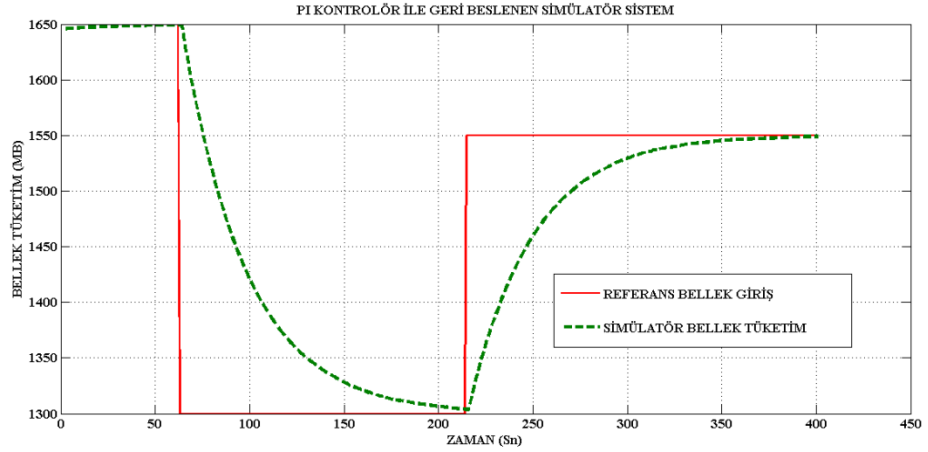
Şekil 5.11 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis, bellek tüketim gösterimi.

Emülatörden elde edilen referans giriş, bellek tahsis, bellek tüketim ve hata verisi Şekil 5.12 ile gösterilmiştir.



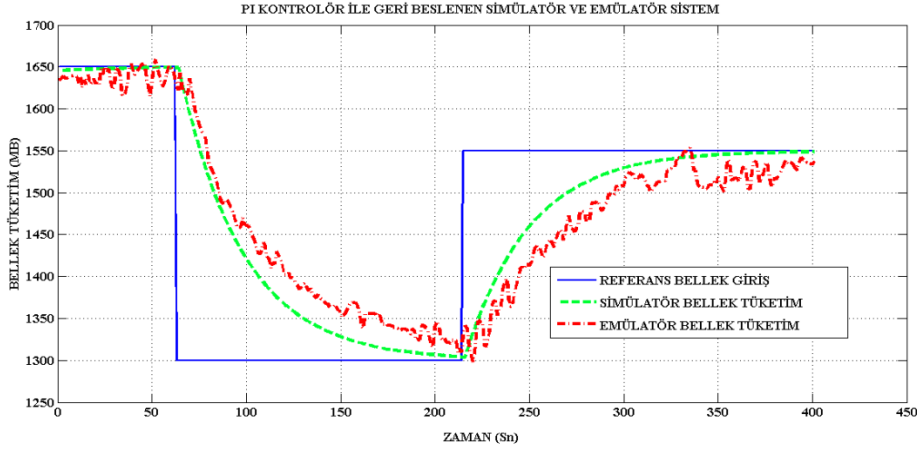
Şekil 5.12 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis, bellek tüketim ve referans ile sistem çıkışı olan bellek kullanımı arasındaki hata bilgilerinin gösterimi.

Simülatörden elde edilen referans giriş, bellek tüketim verisi aşağıdaki Şekil 5.13 ile gösterilmiştir.



Şekil 5.13 : Bellek SISO sistem için simülatör ortamında referans değer, bellek tüketim gösterimi.

Simülatörden ve emülatörden elde edilen referans giriş, bellek tüketim verisi Şekil 5.14'teki grafikte gösterilmiştir.



Şekil 5.14 : Bellek SISO sistem için emülatör ve simülatör ortamında referans değer, bellek tüketim gösterimi.

Geri besleme sisteminde referans girişler QoS gösteren metriklerdir. Referans girişler iş yükü dikkate alınarak sisteme uygulanabilir. Deney sonuçları kontrol teorisine dayalı yaklaşımın, değişen iş yükü özellikle yoğun iş yükü altında çoklu kaynak kullanımının yönetimi problemine dayanıklı bir çözüm getirdiğini göstermiştir. CPU kullanımı ve bellek tüketimi sistem çıkışları PI kontrolörler ile kontrol edilebilmiş ve sistem çıkışları referans girişleri izlemiştir. Böylelikle servis kalite garantisi sağlanmıştır.

i. örneklemede ölçülen sistem çıkışı X_{obs} ve modelden elde edilen çıkış X_{model} ise karesel hataların ortalaması (RMSE: Root Mean Square Error)

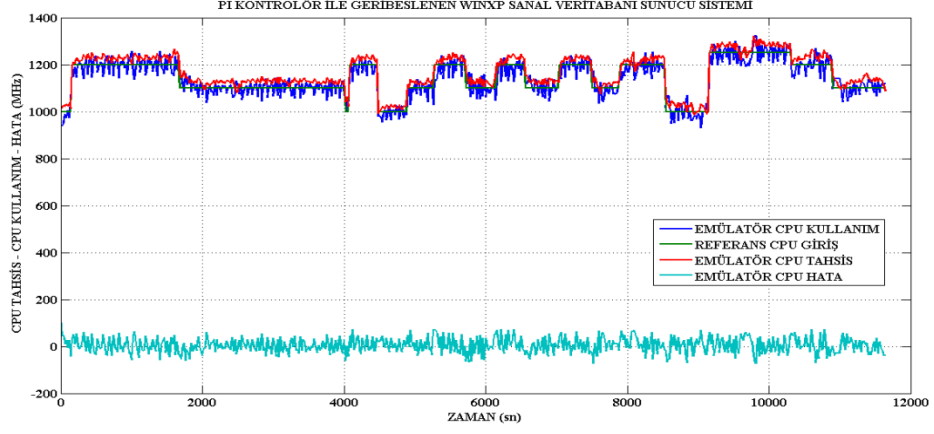
$$RMSE = \sqrt{\frac{\sum_{i=1}^n (X_{obs,i} - X_{model,i})^2}{n}}$$

olarak hesaplanabilir. RMSE değeri, kontrolörlerin başarımlarının sayısal olarak karşılaştırılmasında kullanılmıştır. Ayrıca RMSE değeri ile geri beslemeli sistem ile geri beslemesiz sistemin performansları karşılaştırılmıştır.

5.4 Win7 ve WinXP Veri tabanı Sanal Sunucuları Paralel Çalışma Deneyleri

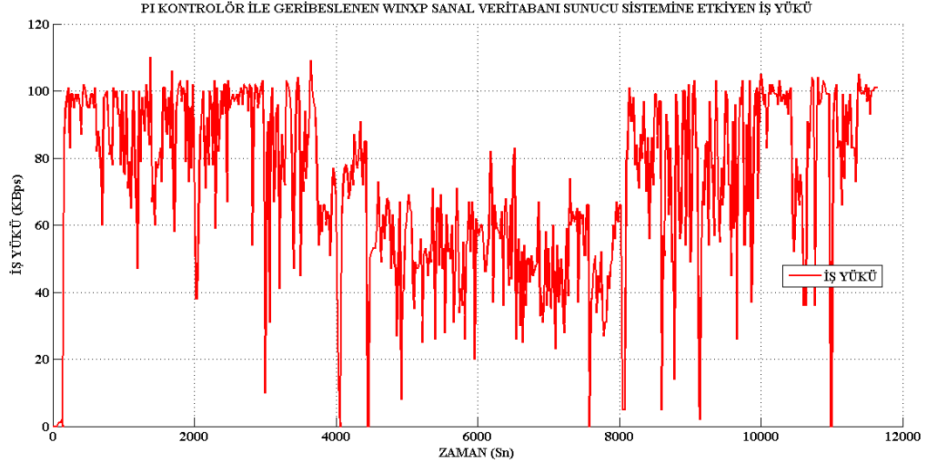
Deney 1: WinXP ve Win7 CPU Sonuçları

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları aşağıdaki Şekil 5.15 ile gösterilmiştir.



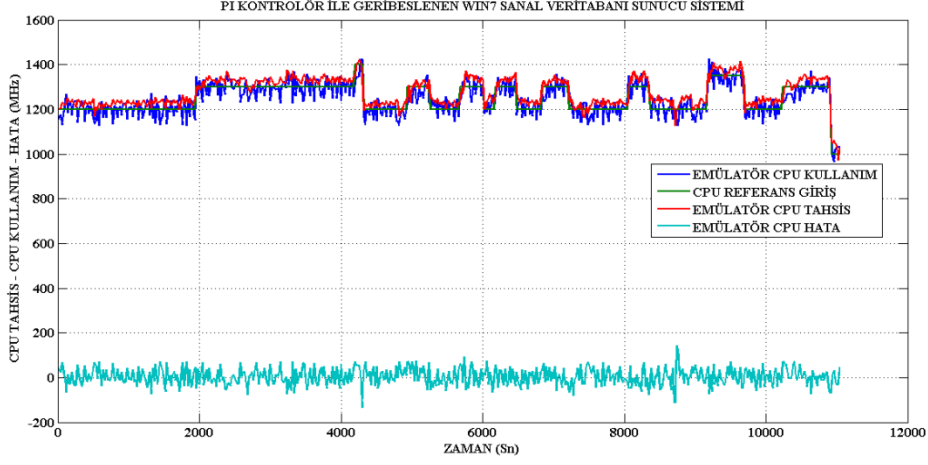
Şekil 5.15 : WinXP sanal makinası emülatör CPU sonuçları.

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına etkiyen iş yükü Şekil 5.16 ile gösterilmiştir.



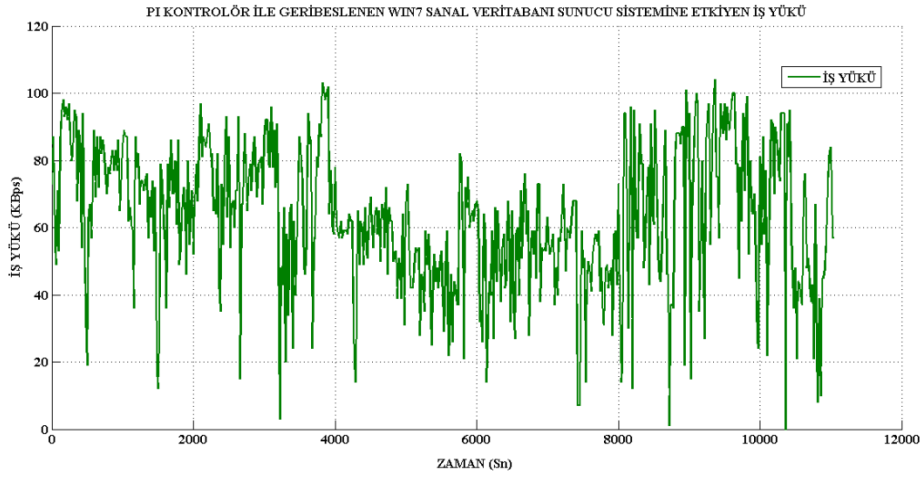
Şekil 5.16 : WinXP sanal makinasına etkiyen iş yükü.

Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına ilişkin CPU sonuçları Şekil 5.17 ile gösterilmiştir.



Şekil 5.17 : Win7 sanal makinası emülatör CPU sonuçları.

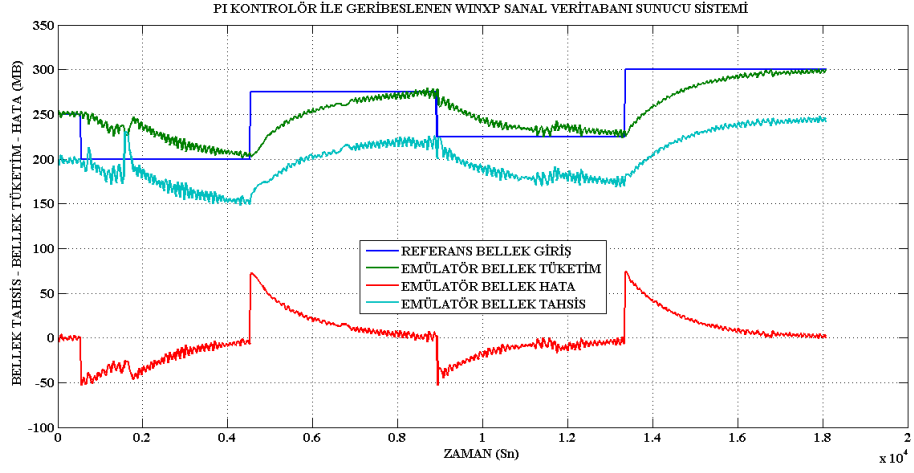
Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına etkiyen iş yükü Şekil 5.18 ile gösterilmiştir.



Şekil 5.18 : Win7 sanal makinasına etkiyen iş yükü.

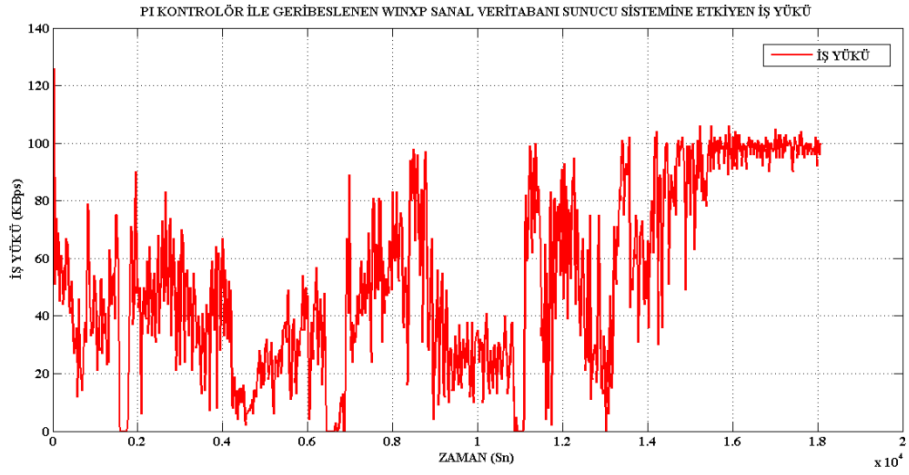
Deney 2: WinXP ve Win7 bellek sonuçları

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına ilişkin bellek sonuçları Şekil 5.19 ile gösterilmiştir.



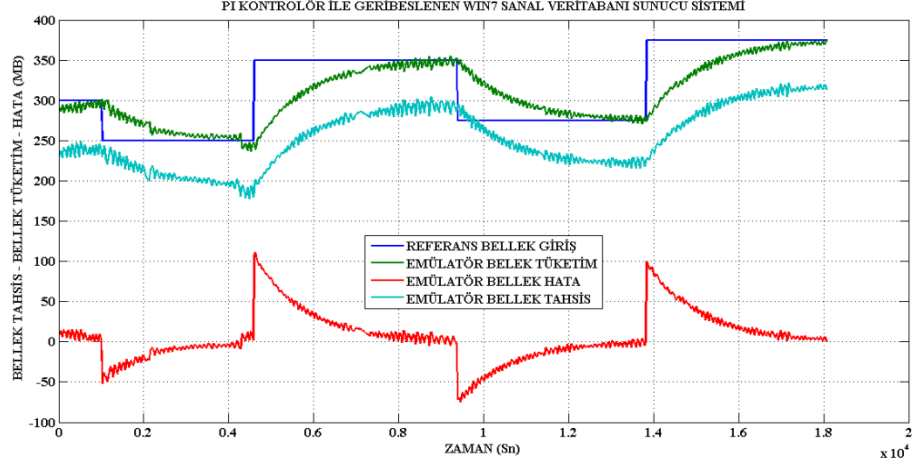
Şekil 5.19 : WinXP sanal makinası emülatör bellek sonuçları.

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına etkiyen iş yükü Şekil 5.20 ile gösterilmiştir.



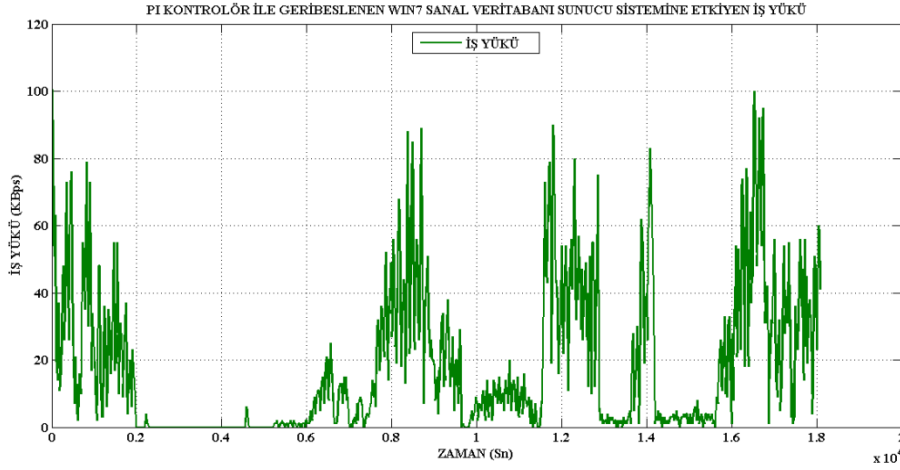
Şekil 5.20 : WinXP sanal makinasına etkiyen iş yükü.

Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına ilişkin bellek sonuçları Şekil 5.21’de gösterilmiştir.



Şekil 5.21 : Win7 sanal makinası emülatör bellek sonuçları.

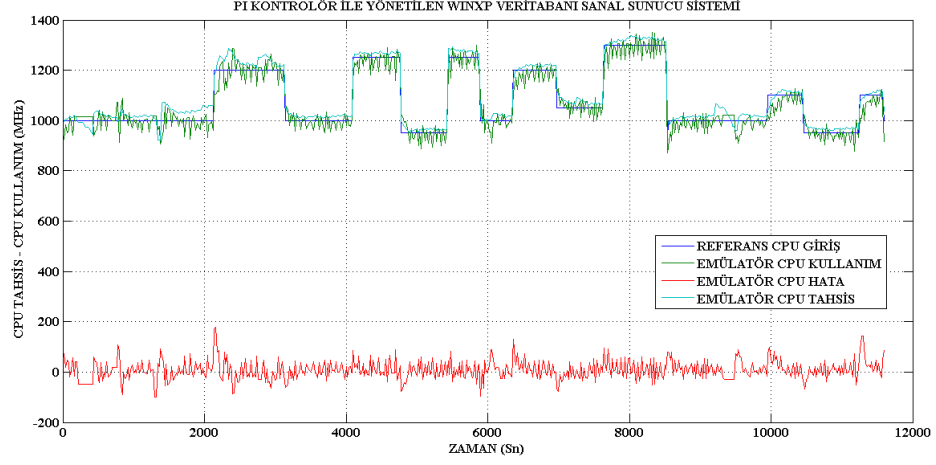
Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına etkiyen iş yükü Şekil 5.22 ile gösterilmiştir.



Şekil 5.22 : Win7 sanal makinasına etkiyen iş yükü.

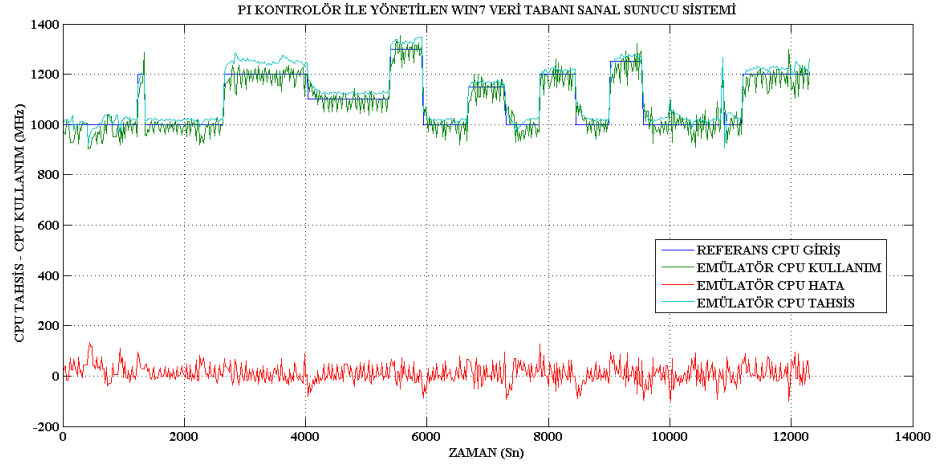
Deney 3: WinXP ve Win7 CPU sonuçları

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları Şekil 5.23 ile gösterilmiştir. Bu deney için RMSE WinXP **35,5580** olarak elde edilmiştir.



Şekil 5.23 : WinXP sanal makinası emülatör CPU sonuçları.

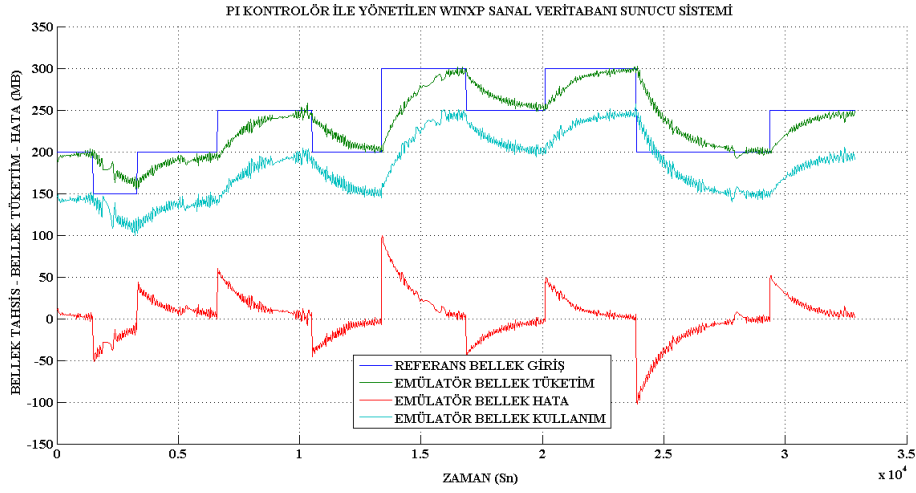
Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına ilişkin CPU sonuçları Şekil 5.24 ile gösterilmiştir. Bu deney için RMSE Win7 **38,5679** olarak elde edilmiştir.



Şekil 5.24 : Win7 sanal makinası emülatör CPU sonuçları.

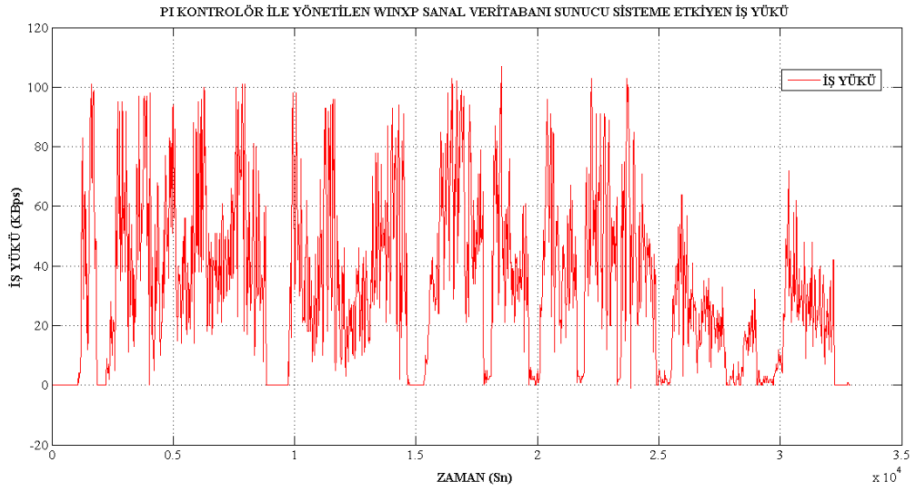
Deney 4: WinXP ve Win7 bellek sonuçları

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına ilişkin bellek sonuçları Şekil 5.25 ile gösterilmiştir. Bu deney için RMSE WinXP **25,5955** olarak elde edilmiştir.



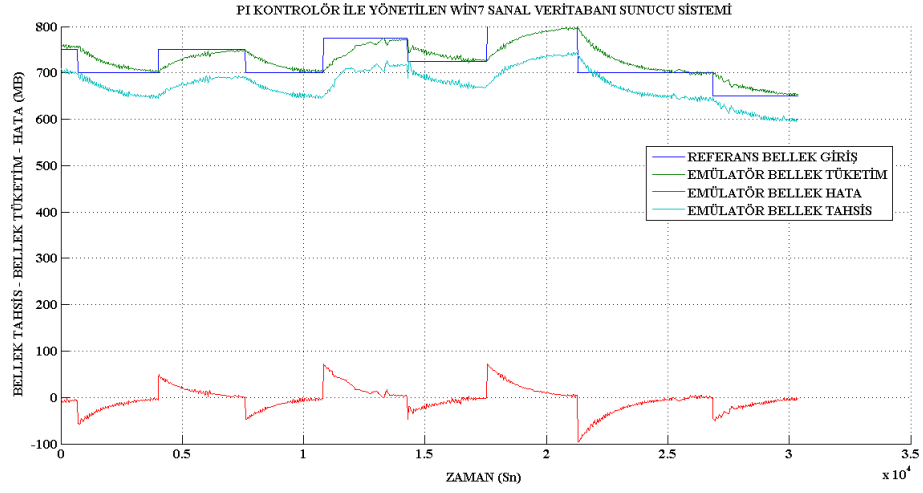
Şekil 5.25 : PI kontrolör ile yönetilen WinXP makinasına ilişkin bellek sonuçları.

Paralel çalışma durumunda PI kontrolör ile yönetilen WinXP makinasına etkiyen iş yükü Şekil 5.26 ile gösterilmiştir.



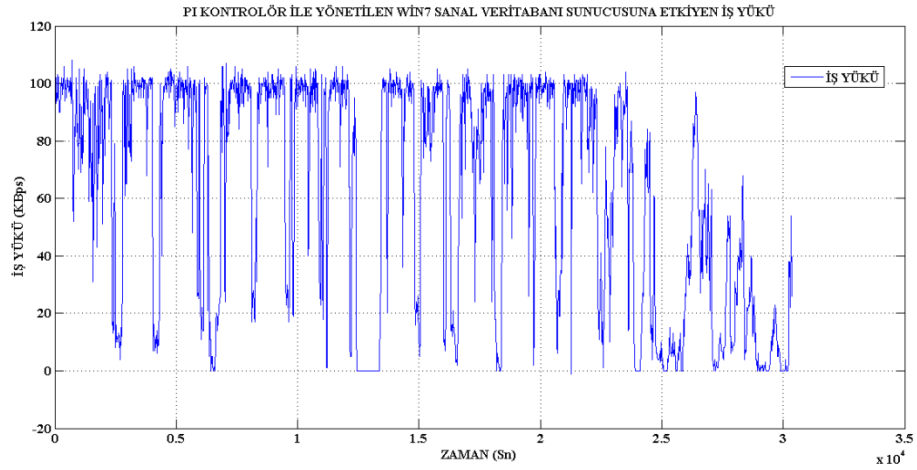
Şekil 5.26 : WinXP sanal makinasına etkiyen iş yükü.

Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına ilişkin bellek sonuçları Şekil 5.27 ile gösterilmiştir. Bu deney için RMSE Win7 **25,6602** olarak elde edilmiştir.



Şekil 5.27 : PI kontrolör ile yönetilen Win7 makinasına ilişkin bellek sonuçları.

Paralel çalışma durumunda PI kontrolör ile yönetilen Win7 makinasına etkiyen iş yükü Şekil 5.28 ile gösterilmiştir.



Şekil 5.28 : Win7 sanal makinasına etkiyen iş yükü.

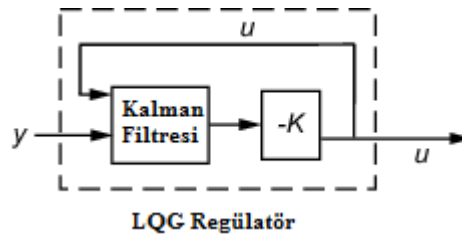
6. KONTROLÖR TASARIMI: INTEGRAL ETKİLİ LQG SERVO KONTROLÖR

Özellikle bozucu etkilerin ve ölçüm gürültülerinin olduğu sistemlerde sürekli hal hatasını mümkün olduğunca azaltmak için ve sistem performansı diğer bir deyişle sistem çıkışının referansı takip etmedeki başarımını yüksek tutarken bunu mümkün olan en küçük enerji ile diğer bir deyişle büyüklüğü sınırlı kontrol işareti ile başarmak için uygun yöntemlerden biri integral etkili Linear-Quadratic Gaussian (LQG) servo kontrolör ile sistemi kontrol etmektir. Kalman filtresi ile sistemin geçmiş giriş ve çıkış işaretleriyle sistemin sonraki durum ve çıkışları kestirilmiştir. Sistemin kestirilen durumları integratörden gelen durum da dikkate alınarak uygun değer durum kazancı ile geri beslenerek diğer bir deyişle sistem durum değişkenlerinin geri beslenmesiyle yönetilmiştir. $u = -Kx$ geri besleme kontrol kuralı ile sistem yönetilmiştir. Aşağıda integral etkili servo kontrolör tasarımı için izlenen yol anlatılmaktadır.

İntegral etkili LQG servo kontrolör için

- 1-) Sistem durumlarının kestirimi için Kalman filtresi oluşturuldu.
- 2-) Optimal durum geri besleme kazancı (LQ) hesaplandı.
- 3-) LQG düzenleyici, Kalman filtresi ve LQ-en uygun kazancının birbirine uygun bir şekilde bağlanmasıyla oluşturuldu.

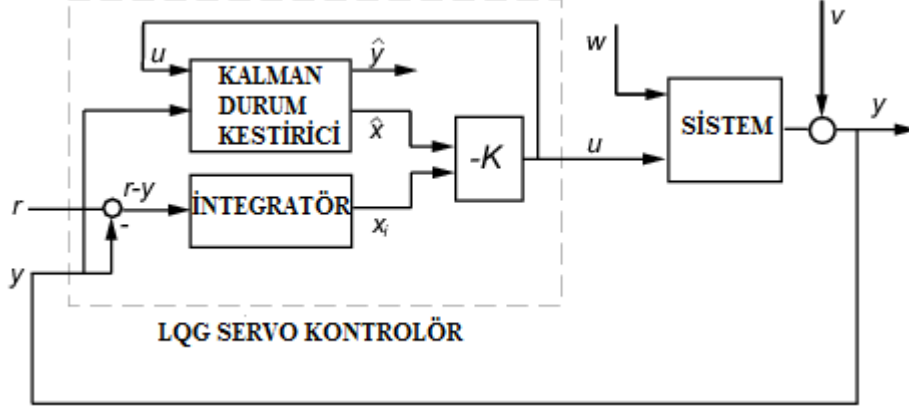
LQG düzenleyici aşağıdaki Şekil 6.1 ile gösterilmektedir.



Şekil 6.1 : LQG düzenleyici.

Sürekli hal hatasını en aza indirmek için integral etkili LQG düzenleyici ile veri tabanı sanal sunucusu kontrol edilmek istenmiştir. Bu sebeple integratör LQG düzenleyiciye dâhil edilmiştir. Tasarlanan servo kontrolörde sistem çıkışı olan y

'nin, referans giriş olan r işaretini takip etmesi istenir. Ayrıca servo kontrolör bozucu etkilerin (w) ve ölçüm hataları v sisteme etkisini engeller. İntegral etkili LQG kontrolör yaygın ismiyle integral etkili LQG servo kontrolör ile yönetilen geri besleme kontrol sistemi Şekil 6.2 ile gösterilmiştir.



Şekil 6.2 : İntegral etkili LQG kontrolör ile yönetilen geri beslemeli sistem.

Kontrolör tasarımında izlenen adımlar detaylı olarak aşağıdaki kısımlarda bahsedilmiştir.

6.1 Kalman Filtresi Tasarımı

CPU tahsis - CPU kullanım SISO sistemi ve bellek tahsis - bellek kullanım SISO sistemi durum uzayında en genel halde (6.1) denklemleri ile ifade edilebilir. x sistem durumlarını, y sistem çıkışlarını, u kontrol işaretlerini, w bozucu giriş işaretlerini ifade eder. A, B, C, D, G, H sistem kazanç matrisleridir.

$$\begin{aligned} x[n+1] &= Ax[n] + Bu[n] + Gw[n] \\ y[n+1] &= Cx[n] + Du[n] + Hw[n] \end{aligned} \quad (6.1)$$

Kalman filtresi, u kontrol işareti ve y çıkış işaretini kullanarak sistem çıkışı $\hat{y}$ ve durumları $\hat{x}$ kestirir. Ayırık Kalman filtresi durum denklemi (6.2) formülü ile ifade edilebilir.

$$\hat{x}(n+1) = A\hat{x}[n] + Bu[n] + L(y[n] - C\hat{x}[n] - Du[n]) \quad (6.2)$$

P , Riccati denkleminin çözümüdür. Kalman kazancı L , ayırık Riccati denkleminin çözümü olan P ve diğer parametreler hesaba katılarak aşağıdaki denklemler ile hesaplanabilir [40].

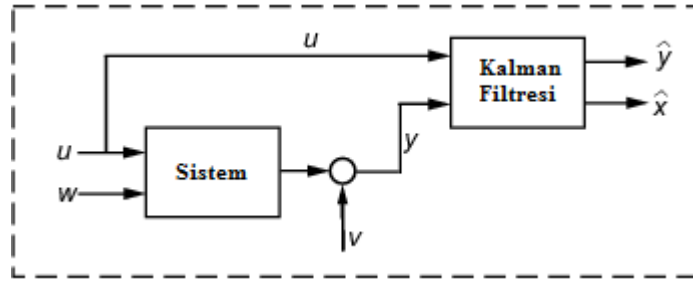
$$L = (APC^T + \bar{N})(CPC^T + \bar{R})^{-1}$$

$$\bar{R} = R + HN + N^T H^T + HQH^T$$

$$\bar{N} = G(QH^T + N)$$

$[kest, L, P] = kalman(sys, Qn, Rn, Nn);$

Yukarıdaki MATLAB komutuyla Kalman sistemine ilişkin durum uzayı parametreleri elde edilmiştir. ‘**sys**’ ile ifade edilen sistem $A, [B \ G], C, [D \ H]$ ile ifade edilen bozucu etkisi altında elde edilmiş durum uzayı modelidir. Veri tabanı sunucusu kontrol işareti olan u ve çıkış işareti olan y ile sistemin bir sonraki çıkış $\hat{y}$ ve durumları $\hat{x}$ kestirilmektedir. Kalman filtresine ilişkin grafik Şekil 6.3’de görülmektedir.



Şekil 6.3 : Kalman Filtresi.

6.1.1 CPU tahsis – CPU kullanım SISO sistemine ilişkin Kalman filtresi durum parametreleri ve Kalman kazancı

Sanal ortamdaki veri tabanı sunucu üzerinde oluşan iş isteklerinin yarattığı iş yükü sistem üzerinde bozucu etki yaratır. CPU tahsis-CPU kullanım SISO sistemi için iş yükü bozucu giriştir, CPU tahsis kontrol işareti, CPU kullanım çıkış büyüklüğüdür. İki örnekleme zamanı gecikmeli, ayrık zamanlı, zamanla değişmeyen CPU tahsis-CPU kullanım SISO sistemi durum uzayında *bozucu etki altında* aşağıdaki sistem parametreleri ile ifade edilebilir.

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0,4286 \end{bmatrix}, B = \begin{bmatrix} 0,5657 \\ 0 \end{bmatrix}, G = \begin{bmatrix} -0,1453 \\ 0 \end{bmatrix}, C = [0 \ 1], D = [0], H = [0]$$

Qn, Rn, Nn tasarım parametreleri belirlendikten sonra **sys** ile ifade edilen CPU SISO sistemi için kalman filtresi sistemi elde edilir. Kalman filtresi ‘**Kest**’ ile isimlendirilmiştir. Aşağıdaki Çizelge 6.1’de CPU tahsis - CPU kullanım SISO

sistemi durum uzayı modeli ile farklı Qn , Rn ve Nn değerleri ile elde edilmiş Kalman filtresi durum parametreleri ve Kalman kazancı, L gösterilmektedir.

Çizelge 6.1 : CPU tahsis-CPU kullanım SISO sistemi için Kalman durum parametreleri ve Kalman kazançları.

Qn	Rn	Nn	Kest.A	Kest.B	Kest.C	Kest.D	L
0,25	2,5	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,4275 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & 0 \\ 0 & 0,0011 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,9974 \\ 1 & 0 \\ 0 & 0,9974 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0026 \\ 0 & 0 \\ 0 & 0,0026 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,0011 \end{bmatrix}$
0,25	2	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,4272 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & 0 \\ 0 & 0,0014 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,9968 \\ 1 & 0 \\ 0 & 0,9968 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0032 \\ 0 & 0 \\ 0 & 0,0032 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,0014 \end{bmatrix}$
0,5	2	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,4259 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & 0 \\ 0 & 0,0027 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,9936 \\ 1 & 0 \\ 0 & 0,9936 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0064 \\ 0 & 0 \\ 0 & 0,0064 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,0027 \end{bmatrix}$
1	1	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,4179 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & 0 \\ 0 & 0,0107 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,9749 \\ 1 & 0 \\ 0 & 0,9749 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0251 \\ 0 & 0 \\ 0 & 0,0251 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,0107 \end{bmatrix}$
1	1	1	$\begin{bmatrix} 0 & 0,1453 \\ 1 & 0,4286 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & -0,1453 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -0,1453 \\ 0 \end{bmatrix}$
0,25	0,06	0,12	$\begin{bmatrix} 0 & 0,2889 \\ 1 & 0,4250 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & -0,2889 \\ 0 & 0,0036 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,994 \\ 1 & 0 \\ 0 & 0,994 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0060 \\ 0 & 0,0010 \\ 0 & 0,0060 \end{bmatrix}$	$\begin{bmatrix} -0,2889 \\ 0,0036 \end{bmatrix}$
0,25	0,25	0,25	$\begin{bmatrix} 0 & 0,1453 \\ 1 & 0,4286 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & -0,1453 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -0,1453 \\ 0,0 \end{bmatrix}$

Veri tabanı sanal sunucu sistemi için CPU tahsis - CPU kullanım sistemine ilişki Kalman filtresi tasarım parametreleri $Qn = [0,25]$; $Rn = [2,5]$; $Nn = [0]$; olarak seçilmiştir.

6.1.2 Bellek tahsis - bellek kullanım SISO sistemine ilişkin Kalman filtresi durum parametreleri ve Kalman kazancı

Bellek tahsis- bellek kullanım SISO sistemi için iş yükü bozucu giriştir, bellek tahsis kontrol işareti, bellek kullanım çıkış büyüklüğüdür. İki örnekleme zamanı gecikmeli, ayrık zamanlı, zamanla değişmeyen bellek tahsis-bellek kullanım SISO sistemi durum uzayında **bozucu etki altında** aşağıdaki sistem parametreleri ile ifade edilebilir.

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix}, B = \begin{bmatrix} 0,1656 \\ 0 \end{bmatrix}, G = \begin{bmatrix} 0,004 \\ 0 \end{bmatrix}, C = [0 \ 0,125], D = [0], H = [0]$$

Qn, Rn, Nn tasarım parametreleri belirlendikten sonra sys ile ifade edilen bellek SISO sistemi için kalman filtresi sistemi elde edilir. Kalman filtresi ‘**Kest2**’ ile isimlendirilmiştir. Çizelge 6.2’de bellek tahsis-bellek kullanım SISO sistemi durum uzayı modeli ile farklı Qn, Rn ve Nn değerleri ile elde edilmiş Kalman filtresi durum parametreleri ve Kalman kazancı, $L2$ gösterilmektedir.

Çizelge 6.2 : Bellek tahsis-bellek kullanım SISO sistemi için tasarım parametreleri ve Kalman kazançları.

Qn	Rn	Nn	Kest2.A	Kest2.B	Kest2.C	Kest2.D	L2
0,25	0,25	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,00004 \end{bmatrix}$
0,25	2,5	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,000004 \end{bmatrix}$
1	1	0	$\begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,00004 \end{bmatrix}$
1	1	1	$\begin{bmatrix} 0 & -0,0005 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0,004 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0,004 \\ 0 \end{bmatrix}$
5	1	1	$\begin{bmatrix} 0 & -0,0005 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0,004 \\ 0 & 0,0001 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,00001 \\ 0 & 0 \\ 0 & 0,00014 \end{bmatrix}$	$\begin{bmatrix} 0,004 \\ 0,0001 \end{bmatrix}$
0,5	1	5	$\begin{bmatrix} 0 & -0,0025 \\ 1 & 0,9723 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0,02 \\ 0 & -0,0008 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0001 \\ 0 & 0,0010 \\ 0 & 0,0008 \end{bmatrix}$	$\begin{bmatrix} 0,02 \\ -0,0008 \end{bmatrix}$
0,25	0,25	0,25	$\begin{bmatrix} 0 & -0,0005 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0,004 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0,004 \\ 0 \end{bmatrix}$

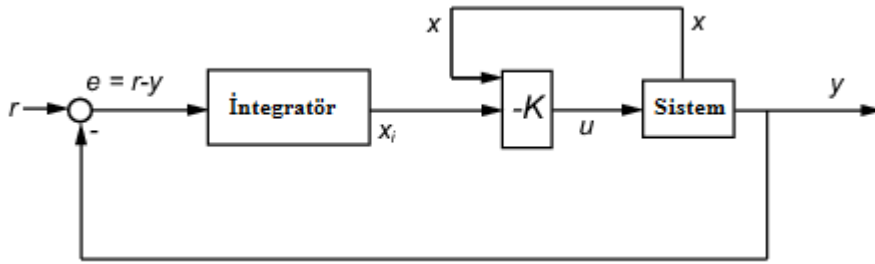
Veri tabanı sanal sunucu sistemi için bellek tahsis-bellek kullanım sistemine ilişki Kalman filtresi tasarım parametreleri $Qn=[0,25]$; $Rn=[0,25]$; $Nn=[0]$; olarak seçilmiştir.

6.2 İntegral Özellikli LQG Kazancı

Ayrık sys sistemi durum uzayında (6.3) denklemleri ile ifade edilebilir.

$$\begin{aligned} x[n+1] &= Ax[n] + Bu[n] \\ y[n+1] &= Cx[n] + Du[n] \end{aligned} \quad (6.3)$$

$u = K[x, x_i]$ durum geri besleme kontrolüdür. x_i integratör çıkışıdır. Bu kontrol kuralı çıkışın (y), r referans girişi takip etmesini sağlar. Linear-Quadratic İntegral (LQI), ayrık zaman uzayında x_i integratör çıkışını $x_i[n+1] = x_i[n] + T_s(r[n] - y[n])$ ileri Euler formülü ile hesaplar. T_s örnekleme zamanını ifade eder. T_s örnekleme zamanı tüm deneylerde 20 saniye olarak alınmıştır. LQI Şekil 6.4 ile gösterilen grafikte referans izleme çevriminde uygun değer durum geri besleme kontrol kuralını hesaplar.



Şekil 6.4 : LQI.

$[K, S, e] = LQI(sys, Q, R, N);$

Yukarıda gösterilen MATLAB komutu vasıtasıyla sistemin durum uzayı modeli ve Q, R, N ağırlık matrisleri hesaplamaya katılarak uygun değer kazanç matrisi K hesaplanır. Optimum K ile ifade edilen kontrol kuralı $u = Kz = K[x, x_i]$, (6.4) denkleminde $J(u)$ ile gösterilen maliyet fonksiyonunu minimize eder. S , Riccati denkleminin çözümüdür, e ise kapalı çevrim sistemin öz değerleridir.

$$J(u) = \sum_{n=0}^{\infty} [z^T Q z + u^T R u + 2z^T N] \quad (6.4)$$

6.2.1 LQG kazancını oluşturmak üzere CPU tahsis-CPU kullanım SISO sistemi için simülatörde gerçekleştirilen deneyler ve sonuçları

CPU tahsis-CPU kullanım SISO sistemi için durum uzayı sistem parametreleri

$$A = \begin{bmatrix} 0,4286 & 0 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = [0 \ 0,5657], \quad D = [0]$$

İki gecikmeli ayrık zamanlı zamanda değişmeyen CPU tahsis-CPU kullanım SISO sistemi transfer fonksiyonu (6.5) ile gösterilmektedir.

$$G(z) = \frac{0,5657}{(z - 0,4286) z} \quad (6.5)$$

CPU tahsis-CPU kullanım SISO sistemi için uygun değer kontrol kuralını elde etmek üzere A, B, C, D durum uzayı sistem modeli ve farklı Q, R, N katsayıları ile deneyler yapılmıştır. Deney sonuçları Çizelge 6.3'te gösterilmiştir.

Çizelge 6.3 : CPU tahsis-CPU kullanım SISO sistemine ilişkin LQ en uygun geri besleme kazancı.

Q	R	N	K	S	e
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	0,9 9	0	$[1,39 \ 0,97 \ -0,08]$	$\begin{bmatrix} 134,08 & 131,49 & -11,62 \\ 131,49 & 260,08 & -22,89 \\ -11,6223 & -22,89 & 3,02 \end{bmatrix}$	$\begin{bmatrix} 0,01 + 0,05i \\ 0,01 - 0,05i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0,01 & 0 & 0 \\ 0 & 0,01 & 0 \\ 0 & 0 & 0,01 \end{bmatrix}$	0,9 9	0	$[0,91 \ 0,60 \ -0,05]$	$\begin{bmatrix} 2,52 & 2,12 & -0,18 \\ 2,12 & 3,15 & -0,27 \\ -0,18 & -0,27 & 0,03 \end{bmatrix}$	$\begin{bmatrix} 0,25 + 0,23i \\ 0,25 - 0,23i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{bmatrix}$	0,9 9	0	$[0,17 \ 0,10 \ -0,009]$	$\begin{bmatrix} 0,19 & 0,12 & -0,01 \\ 0,12 & 0,09 & -0,008 \\ -0,01 & -0,008 & 0,0008 \end{bmatrix}$	$\begin{bmatrix} 0,8152 \\ 0,4382 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$	0,9 9	0	$[1,41 \ 0,98 \ -0,08]$	$\begin{bmatrix} 0,001^* & & \\ 1,32 & 1,30 & -0,11 \\ 1,30 & 2,59 & -0,22 \\ -0,11 & -0,22 & 0,03 \end{bmatrix}$	$\begin{bmatrix} 0,0083 + 0,01i \\ 0,0083 - 0,01i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix}$	0,9 9	0	$[1,41 \ 0,98 \ -0,08]$	$\begin{bmatrix} 0,00001^* & & \\ 1,31 & 1,29 & -0,11 \\ 1,29 & 2,58 & -0,22 \\ -0,11 & -0,22 & 0,03 \end{bmatrix}$	$\begin{bmatrix} 0,0002 \\ 0,0150 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{bmatrix}$	0,0 1	0	$[0,90 \ 0,60 \ -0,05]$	$\begin{bmatrix} 0,02 & 0,02 & -0,001 \\ 0,02 & 0,03 & -0,002 \\ -0,002 & -0,002 & 0,0003 \end{bmatrix}$	$\begin{bmatrix} 0,25 + 0,23i \\ 0,25 - 0,23i \\ 0 \end{bmatrix}$

Deney -) Kalman filtresi için tasarım parametreleri:

$$Qn = 0,25, Rn = 2,5, Nn = 0$$

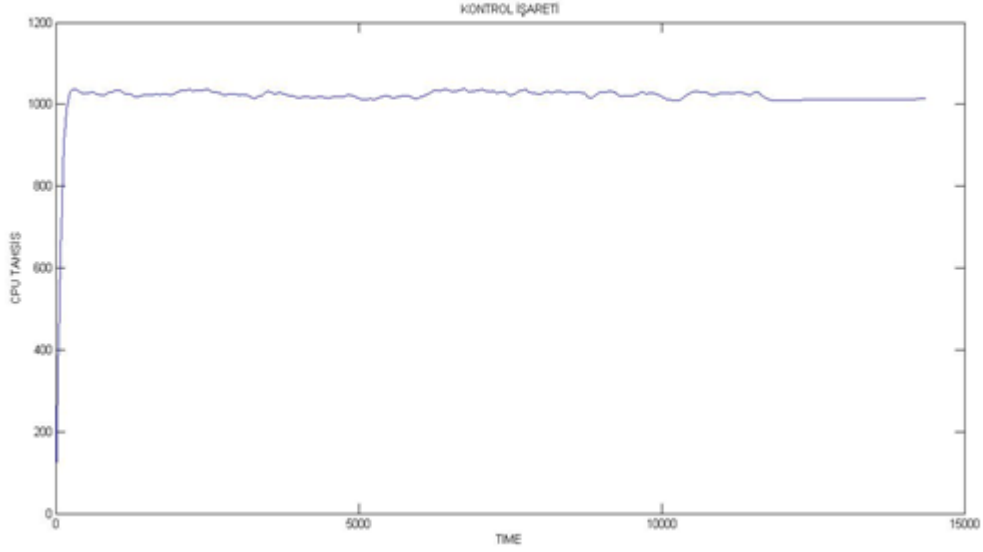
LQI parametreleri:

$$Q = \begin{bmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{bmatrix}$$

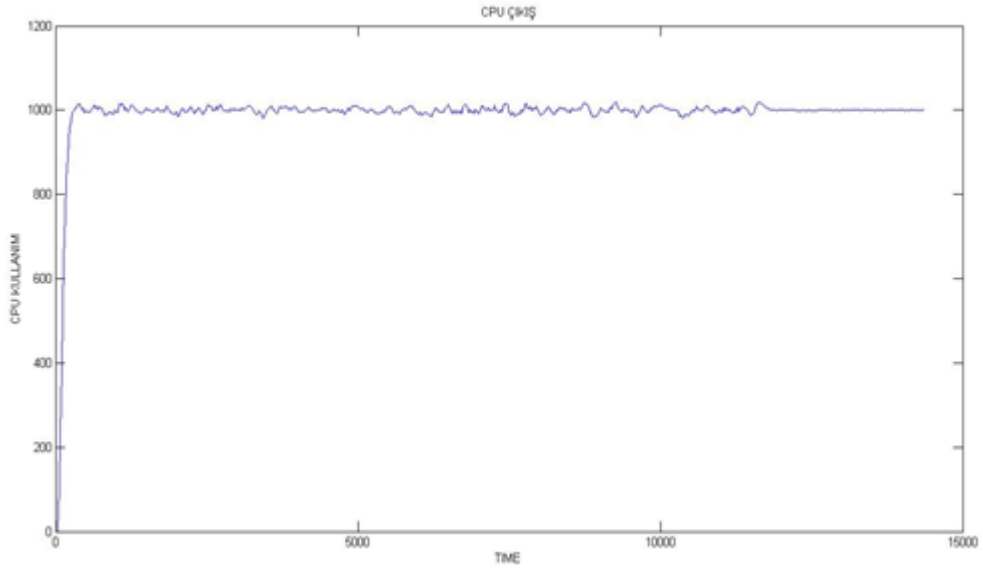
$$R = [0,99]$$

$$N = [0]$$

Emülatör sistemde K kazancı, Q, R, N parametrelerine farklı değerler verilmesiyle elde edilmiş en uygun değer değer olarak seçilmiştir. Yukarıdaki parametrelerle tasarlanan sistem için CPU tahsis kontrol işareti ve CPU kullanım çıkış işareti Şekil 6.5 ve Şekil 6.6 ile gösterilmektedir. Bu Q, R, N parametre değerleri, farklı parametre değerleri ile yapılan deneyler içerisinde en uygun değer sonucu veren parametrelerdir.



Şekil 6.5 : Simülatör CPU tahsis kontrol işareti.



Şekil 6.6 : Simülatör CPU kullanım çıkış işareti.

CPU tahsis- CPU kullanım SISO sistemine ilişkin olarak *LQI için tasarım parametreleri*:

$$Q = \begin{bmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{bmatrix}, R = [0,99], N = [0] \text{ olarak seçilmiştir.}$$

Kalman filtresi için tasarım parametreleri:

$$Qn = [0,25]; \quad Rn = 2,5; \quad Nn = 0; \quad \text{olarak seçilmiştir.}$$

Örnekleme zamanı $T_s = 20$ saniyedir. $K = [0,1752 \quad 0,1038 \quad -0,0092]$ olarak elde edilmiştir. CPU SISO sistemine ilişkin elde edilen Kalman sistem parametreleri Çizelge 6.4'te gösterilen değerler olarak elde edilmiştir.

Çizelge 6.4 : CPU SISO sistemi Kalman filtresi durum uzayı parametreleri.

kest.A	kest.B	kest.C	kest.D	L
$\begin{bmatrix} 0 & 0 \\ 1 & 0,4275 \end{bmatrix}$	$\begin{bmatrix} 0,5657 & 0 \\ 0 & 0,0011 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,9974 \\ 1 & 0 \\ 0 & 0,9974 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,0026 \\ 0 & 0 \\ 0 & 0,0026 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,0011 \end{bmatrix}$

6.2.2 LQG kazancını oluşturmak üzere bellek tahsis-bellek kullanım SISO sistemi için simülatörde gerçekleştirilen deneyler ve sonuçları

Bellek tahsis-bellek kullanım SISO sistemi için durum uzayı sistem parametreleri

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix}, \quad B = \begin{bmatrix} 0,1656 \\ 0 \end{bmatrix}, \quad C = [0 \quad 0,125], \quad D = [0]$$

İki gecikmeli ayrık zamanlı zamanda değişmeyen bellek tahsis-bellek kullanım SISO sistemi transfer fonksiyonu (6.6) formülü ile gösterilmektedir.

$$G(z) = \frac{0,0207}{(z - 0,9722)z} \quad (6.6)$$

Bellek tahsis-bellek kullanım SISO sistemi için uygun değer kontrol kuralını elde etmek üzere A, B, C, D durum uzayı sistem modeli ve farklı Q, R, N katsayıları ile deneyler yapılmıştır. Deney sonuçları Çizelge 6.5'te gösterilmiştir.

Çizelge 6.5 : Bellek tahsis-bellek kullanım SISO sistemine ilişkin LQ en uygun geri besleme kazançları ve karşı düşen öz değerler.

Q	R	N	K	S	e
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	1	0	[5,0748 6,4973 -0,62]	$\begin{bmatrix} 56,74 & 78,33 & -9,65 \\ 78,33 & 121,57 & -17,76 \\ -9,65 & -17,76 & 4,35 \end{bmatrix}$	$\begin{bmatrix} 0,56 + 0,26i \\ 0,56 - 0,26i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0,01 & 0 & 0 \\ 0 & 0,01 & 0 \\ 0 & 0 & 0,01 \end{bmatrix}$	1	0	[1,58 1,756 -0,086]	$\begin{bmatrix} 11,02 & 12,43 & -0,68 \\ 12,43 & 14,27 & -0,86 \\ -0,68 & -0,86 & 0,08 \end{bmatrix}$	$\begin{bmatrix} 0,85 + 0,12i \\ 0,85 - 0,12i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{bmatrix}$	1	0	[0,40 0,42 -0,0097]	$\begin{bmatrix} 2,55 & 2,63 & -0,06 \\ 2,63 & 2,73 & -0,066 \\ -0,06 & -0,066 & 0,0025 \end{bmatrix}$	$\begin{bmatrix} 0,95 + 0,04i \\ 0,95 - 0,04i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$	1	0	[7,82 10,7 -1,24]	$\begin{bmatrix} 200,1 & 306,41 & -48,63 \\ 306,41 & 586,99 & -111,6 \\ -48,63 & -111,6 & 35,74 \end{bmatrix}$	$\begin{bmatrix} 0,33 + 0,27i \\ 0,33 - 0,27i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0,0001 & 0 & 0 \\ 0 & 0,0001 & 0 \\ 0 & 0 & 0,0001 \end{bmatrix}$	0,01	0	[1,58 1,75 -0,087]	$\begin{bmatrix} 0,11 & 0,12 & -0,006 \\ 0,12 & 0,14 & -0,008 \\ -0,006 & -0,008 & 0,0009 \end{bmatrix}$	$\begin{bmatrix} 0,85 + 0,12i \\ 0,85 - 0,12i \\ 0 \end{bmatrix}$
$\begin{bmatrix} 10e-6 & 0 & 0 \\ 0 & 10e-6 & 0 \\ 0 & 0 & 10e-6 \end{bmatrix}$	0,5	0	[0,98 0,099 -0,0014]	$\begin{bmatrix} 0,3 & 0,3026 & -0,0043 \\ 0,3026 & 0,3052 & -0,0044 \\ -0,0043 & -0,0044 & 0,0001 \end{bmatrix}$	$\begin{bmatrix} 0,97 + 0,009i \\ 0,97 - 0,009i \\ 0 \end{bmatrix}$

Deney -) Kalman filtresi için tasarım parametreleri:

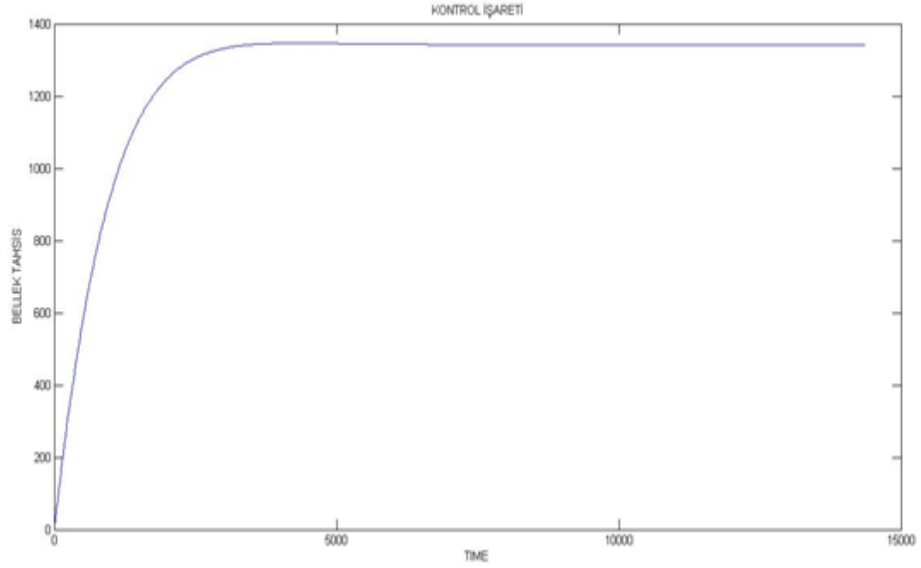
$$Q_n = [0,25]; \quad R_n = [0,25]; \quad N_n = [0];$$

LQI için tasarım parametreleri:

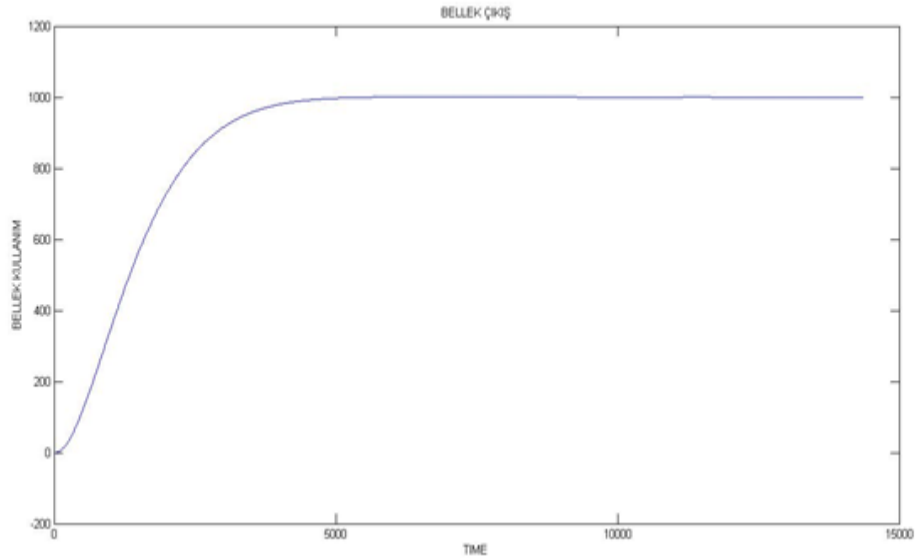
$$Q = \begin{bmatrix} 0,000001 & 0 & 0 \\ 0 & 0,000001 & 0 \\ 0 & 0 & 0,000001 \end{bmatrix}$$

$$R = [0,5], \quad N = [0]$$

Emülatör sistemde K kazancı, Q, R, N parametrelerine farklı değerler verilmesiyle elde edilmiş en uygun değer değer olarak seçilmiştir. Yukarıdaki parametrelerle tasarlanan sistem için bellek tahsis kontrol işareti ve bellek kullanım çıkış işareti Şekil 6.7 ve Şekil 6.8 ile gösterilmektedir. Bu Q, R, N parametre değerleri, farklı parametre değerleri ile yapılan deneyler içerisinde en uygun değer sonucu veren parametrelerdir.



Şekil 6.7 : Simülör bellek tahsis kontrol işareti.



Şekil 6.8 : Simülör bellek kullanım çıkış işareti.

Bellek tahsis-bellek kullanım SISO sistemine ilişkin olarak

Kalman filtresi için tasarım parametreleri:

$Qn = [0,25]$; $Rn = [0,25]$; $Nn = [0]$; olarak seçilmiştir.

LQI için tasarım parametreleri:

$$Q = \begin{bmatrix} 0,000001 & 0 & 0 \\ 0 & 0,000001 & 0 \\ 0 & 0 & 0,000001 \end{bmatrix},$$

$R = [0,5]$, $N = [0]$ olarak seçilmiştir.

Örnekleme zamanı $T_s = 20$ saniyedir.

$K = [0,98 \ 0,099 \ -0,0014]$ olarak elde edilmiştir.

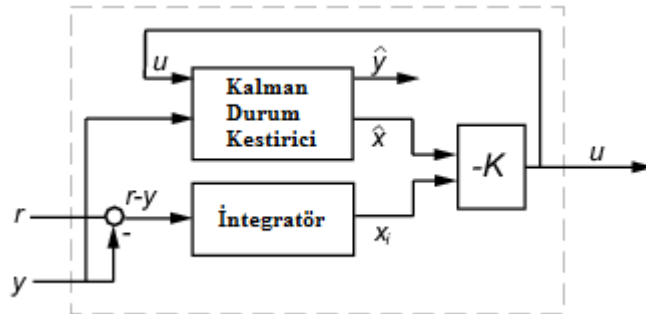
Bellek SISO sistemine ilişkin elde edilen Kalman filtresinin sistem parametreleri Çizelge 6.6'da gösterilmiştir.

Çizelge 6.6 : Bellek SISO sistemi Kalman filtresi durum uzayı parametreleri.

kest2.A	kest2.B	kest2.C	kest2.D	L2
$\begin{bmatrix} 0 & 0 \\ 1 & 0,9722 \end{bmatrix}$	$\begin{bmatrix} 0,1656 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0,125 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0,00004 \end{bmatrix}$

6.3 İntegral Etkili LQG Servo Kontrolörün Oluşturulması

Kalman kestiricisi **kest** ve durum geri besleme kazancı **K**'nın birbirine bağlanmasıyla oluşturulan LQG servo kontrolör **C**, Şekil 6.9 ile gösterilmiştir. Kontrolör **C** 'nin girişleri u ve y 'dir. Bu girişler $u = -K[\hat{x}, x_i]$ komutunu üretir. $\hat{x}$ sanal veri tabanı sisteminin durumlarının Kalman filtresi tarafından kestirilmiş değerleridir. x_i integratör çıkışıdır. Kazanç matrisi olan K , x_i nin boyutunu belirler x_i, y ve r aynı boyuttadır.



Şekil 6.9 : İntegral etkili servo kontrolör.

Kontrolör aşağıdaki MATLAB komutu ile elde edilebilir. Kalman kestiricisi **kest**, durum geri besleme kazancı **K** ile gösterilmiştir.

$$C = \text{lgtrack}(\text{kest}, K);$$

Kontrolör durum uzayı denklemleri (6.7) denklemleri ile gösterilmiştir.

$$\begin{bmatrix} \dot{\hat{x}} \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} A - BK_x - LC + LDK_x & -BK_i + LDK_i \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ x_i \end{bmatrix} + \begin{bmatrix} -L \\ I \end{bmatrix} e \quad (6.7)$$

$$u = [-K_x \ K_i] \begin{bmatrix} \hat{x} \\ x_i \end{bmatrix}$$

6.3.1 LQG servo kontrolörün CPU için oluşturulması

CPU SISO sistemine ilişkin LQG servo kontrolör transfer fonksiyonu $G_c(z)$ (6.8) denklemi ile ifade edilir.

$$C = \text{lgtrack}(kest, K);$$

$$G_c = \text{tf}(C);$$

$$G_c(z) = \frac{0,1835z^2 - 0,07845z}{z^3 - 1,328z^2 + 0,3446z - 0,0162} R(z) + \frac{-0,0002676z^3 - 0,1832z^2 + 0,07845z}{z^3 - 1,328z^2 + 0,3446z - 0,0162} Y(z) \quad (6.8)$$

$$u(k) = 1,328u(k-1) - 0,3446u(k-2) + 0,0162u(k-3) + 0,1835r(k-1) - 0,07845r(k-2) - 0,0002676y(k) - 0,1832y(k-1) + 0,7845y(k-2) \quad (6.9)$$

CPU kontrolör fark denklemi 6.9 ile ifade edilir. $u(k)$, k örnekleme zamanındaki CPU tahsis kontrol işareti $y(k)$ veri tabanı sunucusu CPU kullanım sistem çıkışı, $r(k)$ CPU kullanımı için referans giriştir. İki örnekleme zamanı gecikmeli ayrık zamanlı zamanda değişmeyen CPU tahsis-CPU kullanım SISO sistemi transfer fonksiyonu (6.10) denklemiyle ifade edilmiştir.

$$G(z) = \frac{0,5657 z}{(z - 0,4286) z^2} = \frac{0,5657}{(z - 0,4286) z} \quad (6.10)$$

MATLAB komutu ile kapalı çevrim sistem ayrık zamanda (6.11)'de gösterilen $T(z)$ transfer fonksiyonu ile gösterilebilir.

$$T = \text{feedback}(C^*_{sys}, 1);$$

$$T(z) = \frac{Y(z)}{R(z)} = \frac{G_c(z) G(z)}{1 + G_c(z) G(z)}$$

$$T(z) = \frac{0,0001514 z^3 + 0,1037 z^2 - 0,04438 z}{z^5 - 1,757z^4 + 0,9141z^3 - 0,06024z^2 - 0,03743z} \quad (6.11)$$

Kapalı çevrim transfer fonksiyonun paydası karakteristik denklemi ifade eder. Karakteristik denklemin köklerinin z -tanım bölgesindeki yerleri, sistemin kararlılığı konusunda bilgi verir.

Karakteristik denklem : $z^5 - 1,757z^4 + 0,9141z^3 - 0,06024z^2 - 0,03743z$.

Karakteristik denklemin kökleri

$Kökler = roots([1 -1,757 0,9141 -0,06024 -0,03743 0]);$

$$z_1 = 0$$

$$z_2 = 0,7412 + 0,1519i$$

$$z_3 = 0,7412 - 0,1519i$$

$$z_4 = 0,4275$$

$z_5 = -0,1529$ birim çemberin içerisinde olduğundan veri tabanı sunucusu CPU SISO sistemi kararlıdır. Sistem kararlı olduğundan son değer teoremi ile sürekli hal cevabı elde edilebilir.

Son değer teoremi

$$\lim_{k \rightarrow \infty} x(k) = \lim_{z \rightarrow 1} [(1 - z^{-1}) X(z)] \text{ olarak ifade edilir.}$$

Birim basamak CPU tahsis girişi

$$R(z) = \frac{z}{(z - 1)}$$

olarak gösterilir.

Birim basamak CPU tahsis için CPU kullanım sistem cevabı

$$\lim_{z \rightarrow 1} [(1 - z^{-1}) Y(z)] = \lim_{z \rightarrow 1} [(1 - z^{-1}) T(z) R(z)]$$

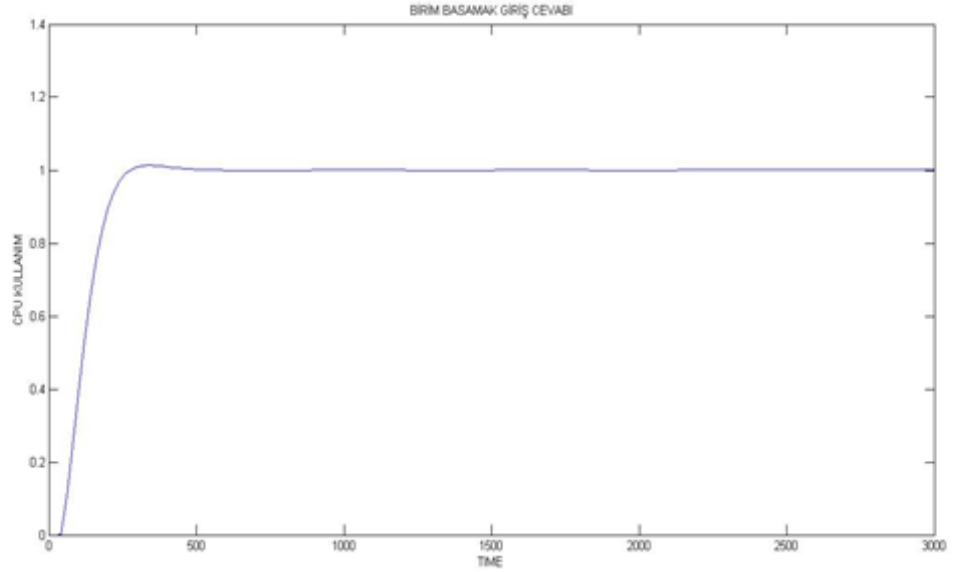
$$\lim_{z \rightarrow 1} \left[(1 - z^{-1}) \frac{0,0001514 z^3 + 0,1037 z^2 - 0,04438 z}{z^5 - 1,757z^4 + 0,9141z^3 - 0,06024z^2 - 0,03743z} \frac{z}{(z-1)} \right] = 1 \text{ olarak elde edilir.}$$

CPU SISO modeline ilişkin sürekli hal hatası

$y(\infty) = 1$, $e_{ss} = r - y(\infty) = 1 - 1 = 0$ olarak elde edilmiştir. CPU SISO sistemi için kapalı çevrimde sistemin birim basamak cevabı girişi için sürekli hal hatası oluşmamıştır. On beş örnekleme zamanı sonunda birim basamak girişine karşılık CPU kullanım sistem çıkışı aşağıdaki Şekil 6.10 ile gösterilmiştir.

$$[y,t] = \text{step}(\text{feedback}(C * \text{sys}, 1), 300);$$

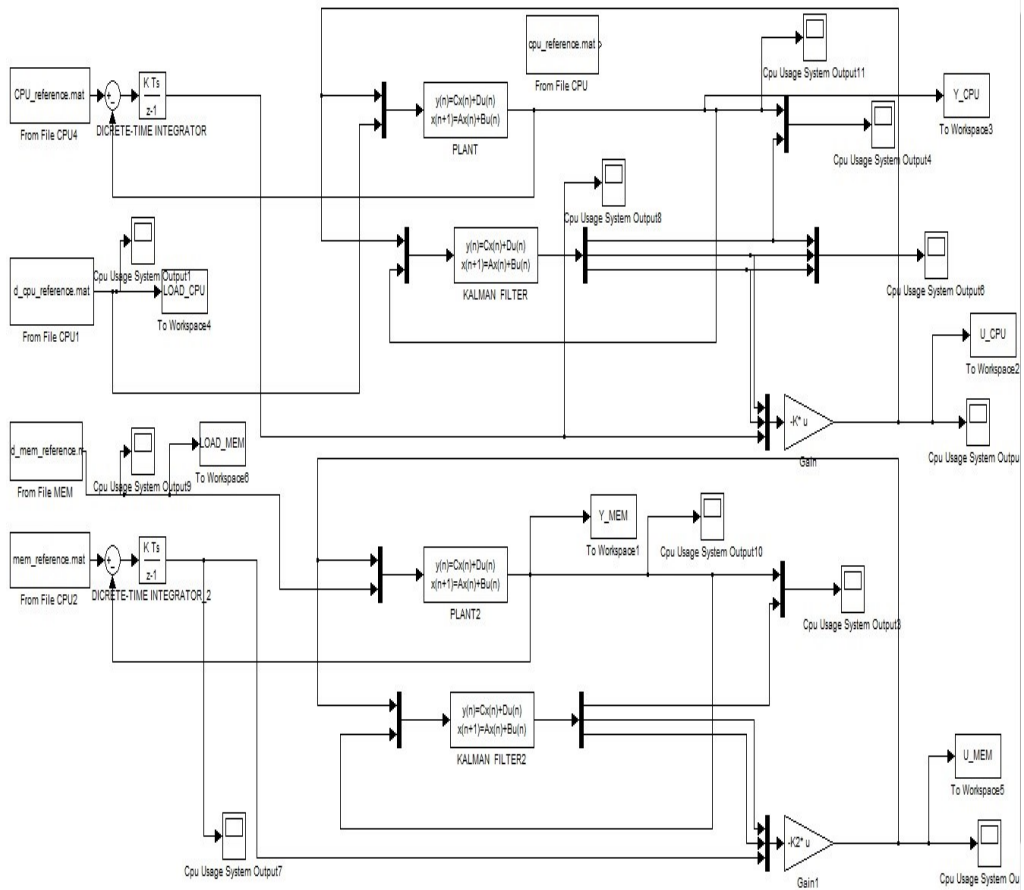
$$\text{plot}(t,y);$$



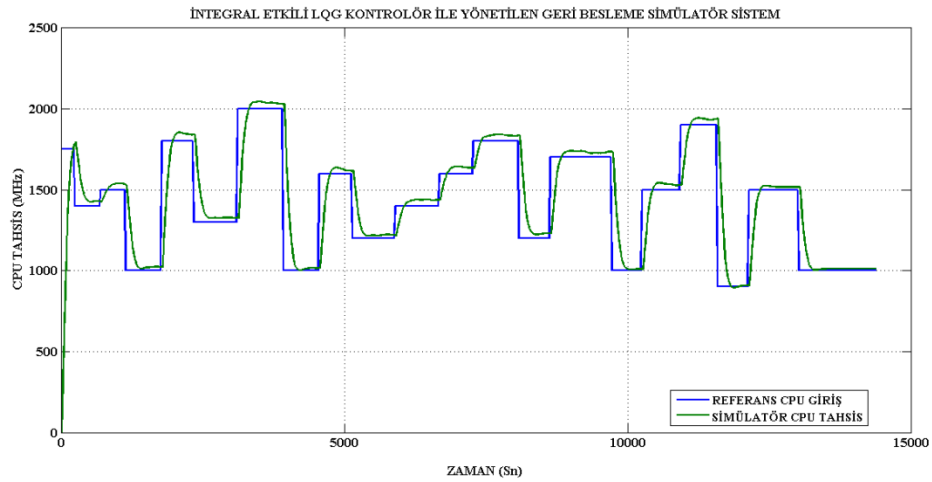
Şekil 6.10 : Simülatör veri tabanı sunucusu CPU modeline ilişkin birim basamak girişi cevabı.

6.3.2 CPU tahsis-CPU kullanım SISO sistemine ilişkin simülatör ve emülatörden elde edilen deney sonuçları

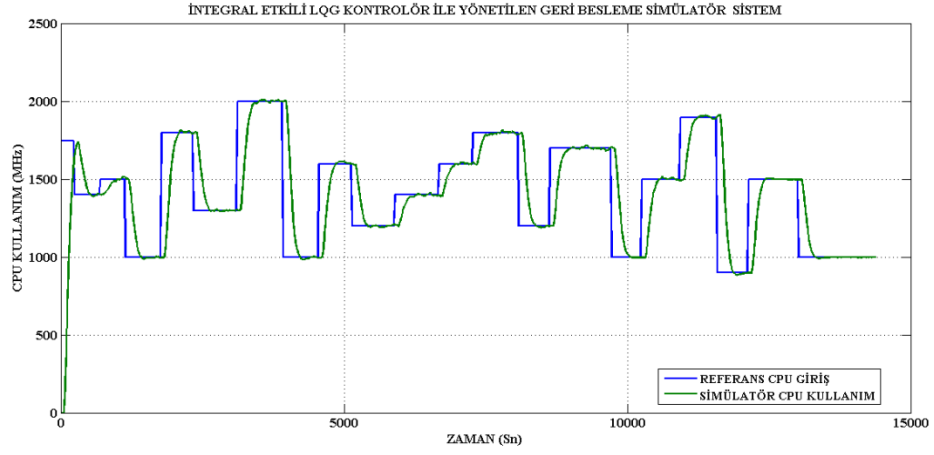
İlk olarak kontrolör sistemi simülinkte tasarladık. CPU tahsis-CPU kullanım sistemi ve bellek tahsis-bellek kullanım sistemlerine ilişkin simülink devresi Şekil 6.11’de gösterilmiştir. Simülinkte tasarladığımız kontrolör ile gerçek sistemi emulatörde çalıştırdık. Emülatörde, çalışma esnasında gerçek zamanlı olarak referans girişler, bozucu girişler, kontrol işaretleri ve sistem çıkışlarının aldığı değerler bir dosyaya kaydedildi. Bir sonraki aşamada gerçek sistemin çalıştığı referans giriş ve bozucu giriş işaretleri simülatördeki sisteme uygulanarak sonuçlar elde edildi ve kaydedildi. En son aşamada emülatörden elde edilen gerçek sistem giriş-çıkış değerleriyle, simülatörden elde edilen giriş-çıkış değerleri grafikler üzerinde karşılaştırıldı. CPU tahsis-CPU kullanım SISO sistemine ilişkin elde edilen deney sonuçları Şekil 6.12-21 şekil numaraları arasındaki grafiklerle gösterilmiştir.



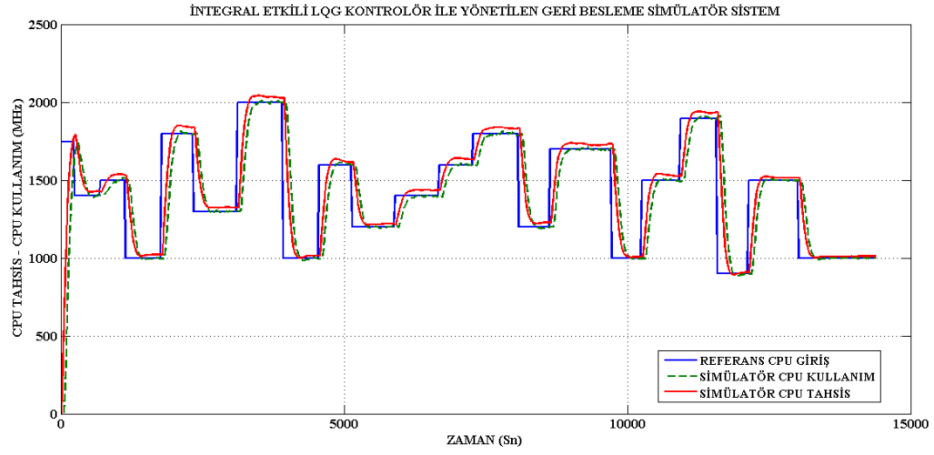
Şekil 6.11 : Sanal ortamdaki veri tabanı sisteminin LQG servo kontrolör ile geri besleme kontrolüne ilişkin simülasyon.



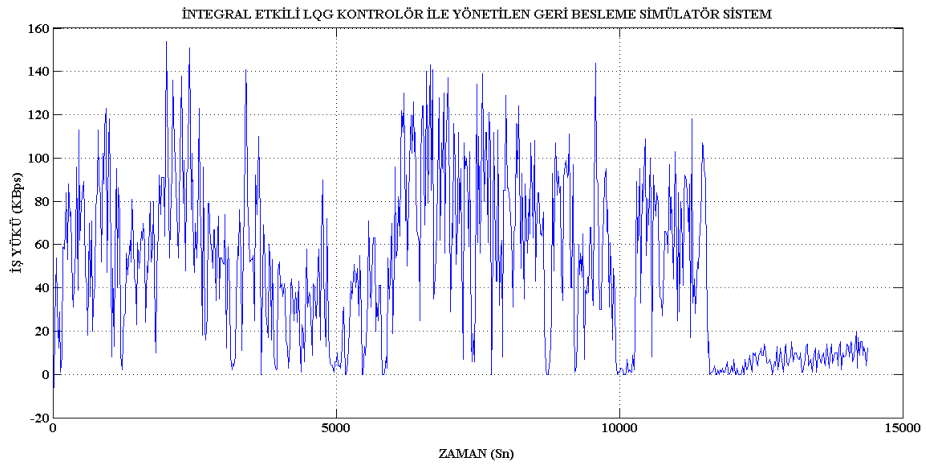
Şekil 6.12 : CPU SISO sistem için simülator ortamında referans değer ve CPU tahsis bilgilerinin gösterimi.



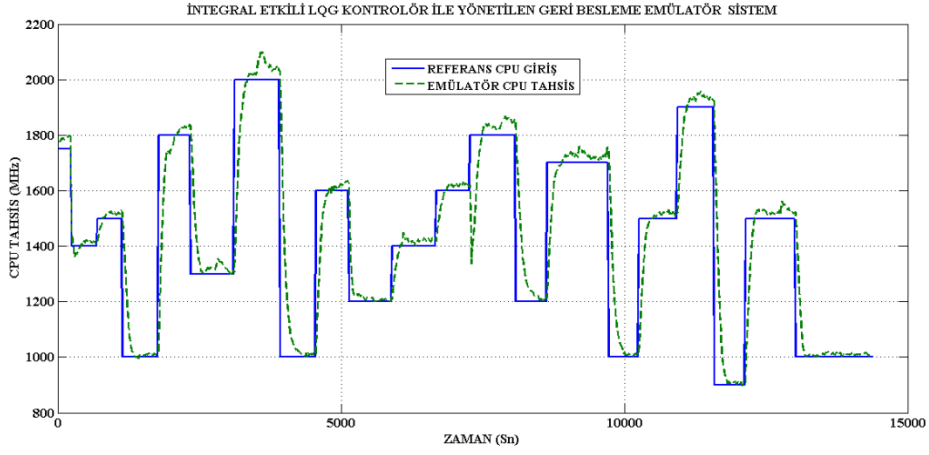
Şekil 6.13 : CPU SISO sistem için simülör ortamında referans deęer ve CPU kullanım bilgilerinin gösterimi.



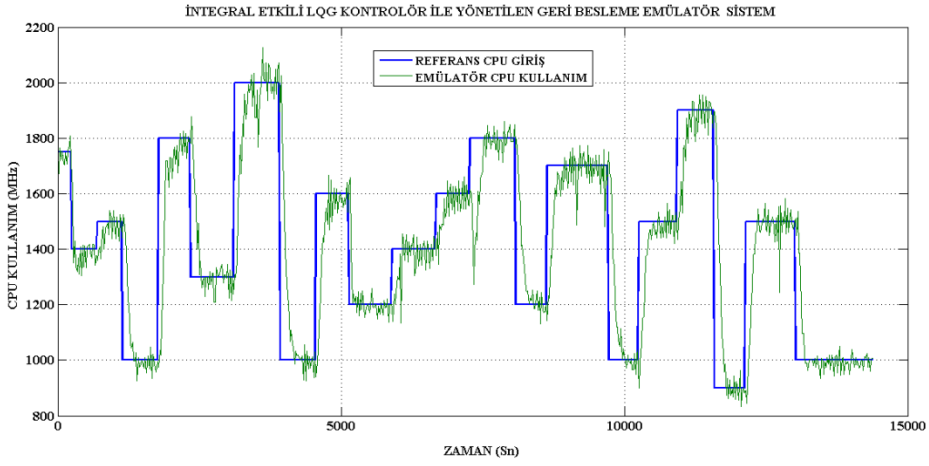
Şekil 6.14 : CPU SISO sistem için simülör ortamında referans deęer CPU tahsis ve CPU kullanım bilgilerinin gösterimi.



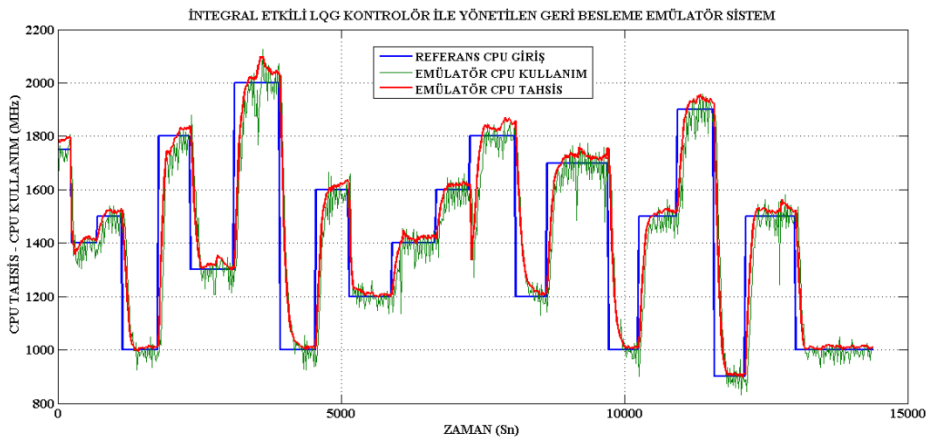
Şekil 6.15 : Simülör sanal ortamdaki veri tabanı sunucusuna etkiyen iş yükü dağılımı.



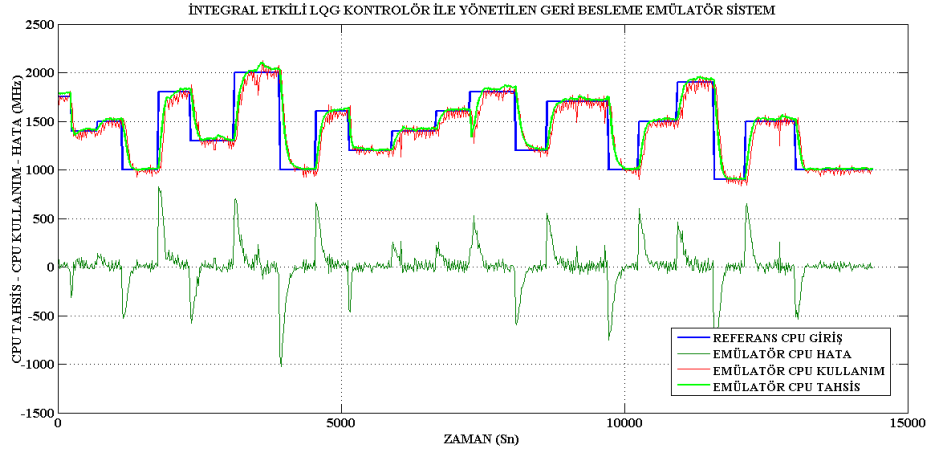
Şekil 6.16 : CPU SISO sistem için emülatör ortamında referans değer ve CPU tahsis gösterimi.



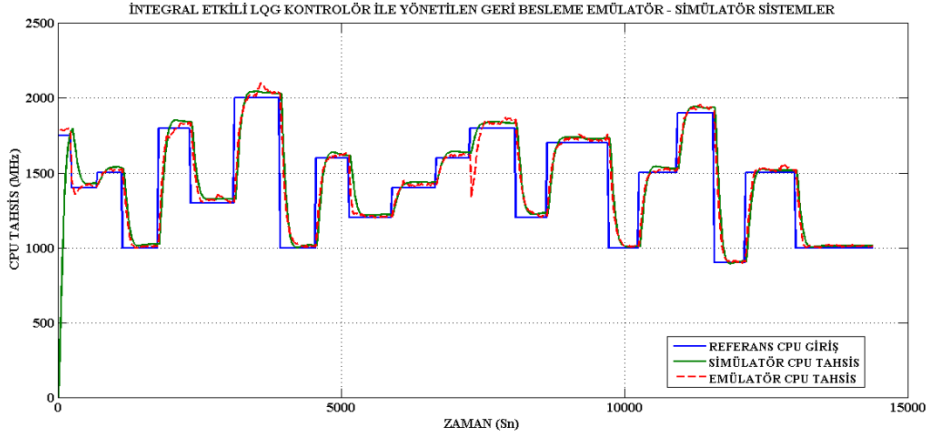
Şekil 6.17 : CPU SISO sistem için emülatör ortamında referans değer ve CPU kullanım gösterimi.



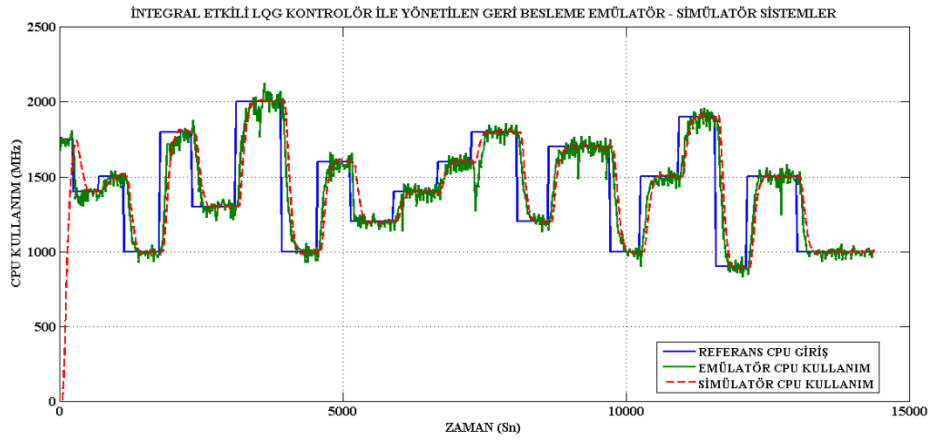
Şekil 6.18 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis ve CPU kullanım gösterimi.



Şekil 6.19 : CPU SISO sistem için emülatör ortamında referans değer, CPU tahsis, CPU kullanım ve referans ile sistem çıkışı olan CPU kullanımı arasındaki hata bilgilerinin gösterimi.



Şekil 6.20 : CPU SISO sistem için simülatör ve emülatör ortamları için referans değer, CPU tahsis gösterimi.



Şekil 6.21 : CPU SISO sistem için simülatör ve emülatör ortamları için referans değer, CPU kullanım gösterimi.

6.3.3 LQG servo kontrolörün bellek için oluşturulması

Bellek SISO sistemine ilişkin LQG servo kontrolör transfer fonksiyonu $G_c(z)$ (6.12) denklemi ile ifade edilir.

$$C = \text{lqgtrack}(kest, K);$$

$$G_c = \text{tf}(C);$$

$$G_c(z) = \frac{0,02805z^2 - 0,02727z}{z^3 - 1,956z^2 + 0,9564z - 0,0005807} R(z) + \frac{-0,02827z^2 + 0,02727z}{z^3 - 1,956z^2 + 0,9564z - 0,0005807} Y(z) \quad (6.12)$$

Bellek için kontrolör fark denklemi (6.13) ile ifade edilebilir.

$$u(k) = 1,956u(k-1) - 0,9564u(k-2) + 0,0005807u(k-3) + 0,02805r(k-1) - 0,02727r(k-2) - 0,02805y(k-1) + 0,02727y(k-2) \quad (6.13)$$

$u(k)$, k örnekleme zamanındaki bellek tahsis kontrol işareti, $y(k)$ veri tabanı sunucusu bellek kullanım sistem çıkışı, $r(k)$ bellek kullanımı için referans giriştir. İki örnekleme zamanı gecikmeli, ayrık, zamanda değişmeyen bellek tahsis-bellek kullanım SISO sistemi transfer fonksiyonu (6.14) ile formüle edilmiştir.

$$G(z) = \frac{0,0207}{(z - 0,9722)z} \quad (6.14)$$

MATLAB komutu ile kapalı çevrim sistem ayrık zamanda aşağıdaki $T(z)$ transfer fonksiyonu ile gösterilebilir.

$$T = \text{feedback}(C * \text{sys}, I);$$

$$T(z) = \frac{Y(z)}{R(z)} = \frac{G_c(z) G(z)}{1 + G_c(z) G(z)}$$

$$T(z) = \frac{7,503e-08 z^3 + 0,0005806 z^2 - 0,0005646 z + 2,355e-19}{z^5 - 2,928 z^4 + 2,858 z^3 - 0,9299 z^2 - 1,249e-16 z + 2,355e-19}$$

$T(z)$ yaklaşık olarak (6.15) ile ifade edilebilir.

$$T(z) = \frac{0,000586 z^2 - 0,0005646 z}{z^5 - 2,928z^4 + 2,858z^3 - 0,9299z^2} \quad (6.15)$$

Kapalı çevrim transfer fonksiyonun paydası karakteristik denklemi ifade eder. Karakteristik denklemin köklerinin z-tanım bölgesindeki yerleri, sistemin kararlılığı konusunda bilgi verir.

$$\text{Karakteristik denklem : } z^5 - 2,928z^4 + 2,858z^3 - 0,9299z^2$$

Karakteristik denklemin kökleri

$$\text{Kökler} = \text{roots}([1 -2,928 \ 2,858 -0,9299 \ 0 \ 0]);$$

$$z_1 = 0$$

$$z_2 = 0$$

$$z_3 = 0,9965 + 0,0391i$$

$$z_4 = 0,9965 - 0,0391i$$

$z_5 = 0,9351$ birim çemberin içerisinde olduğundan veri tabanı sunucusu bellek SISO sistemi kararlıdır. Sistem kararlı olduğundan son değer teoremi ile sürekli hal cevabı elde edilebilir.

Son değer teoremi

$$\lim_{k \rightarrow \infty} x(k) = \lim_{z \rightarrow 1} [(1 - z^{-1}) X(z)] \text{ olarak elde edilir.}$$

Birim basamak CPU tahsis girişi

$$R(z) = \frac{z}{(z - 1)}$$

olarak gösterilir.

Birim basamak CPU tahsis için CPU kullanım sistem cevabı

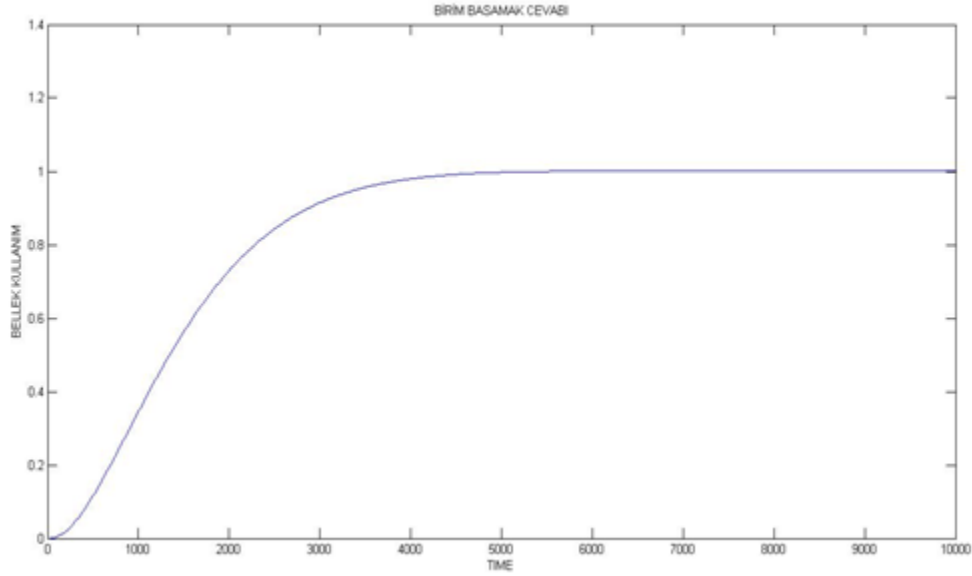
$$\lim_{z \rightarrow 1} [(1 - z^{-1}) Y(z)] = \lim_{z \rightarrow 1} [(1 - z^{-1}) T(z) R(z)]$$

$$\lim_{z \rightarrow 1} \left[(1 - z^{-1}) \frac{0,000586 z^2 - 0,0005646 z}{z^5 - 2,928z^4 + 2,858z^3 - 0,9299z^2} \frac{z}{(z-1)} \right] = 1 \text{ olarak elde edilir.}$$

Bellek SISO modeline ilişkin sürekli hal hatası $y(\infty)1$, $e_{ss} = r - y(\infty) = 1 - 1 = 0$ olarak elde edilmiştir. Bellek SISO sistemi için kapalı çevrimde sistemin birim basamak cevabı girişi için sürekli hal hatası oluşmamıştır. Beş yüz örnekleme zamanı sonunda birim basamak girişine karşılık bellek kullanım sistem çıkışı aşağıdaki Şekil 6.22'de gösterilmiştir.

$[y,t]=step(feedback(C*sys,1),10000);$

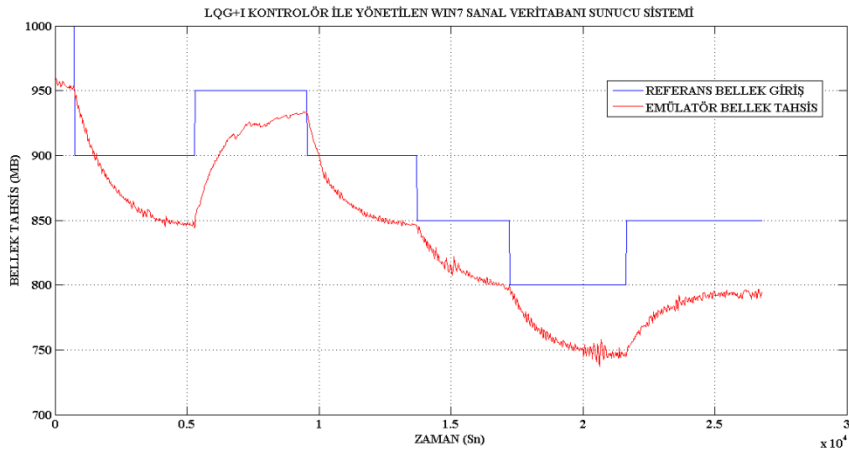
$plot(t,y);$



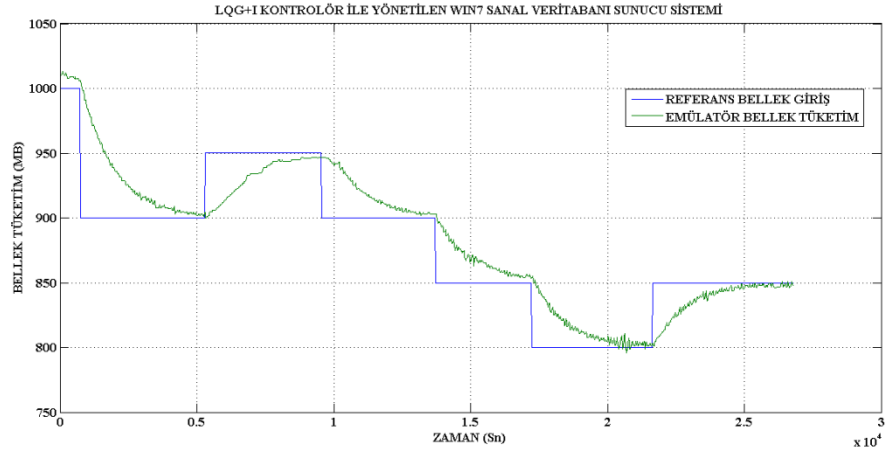
Şekil 6.22 : Simülör ortamda veri tabanı sunucusu bellek modeline ilişkin birim basamak girişi cevabı.

6.3.4 Bellek tahsis-bellek kullanım SISO sistemine ilişkin simülör ve emülörden elde edilen deney sonuçları

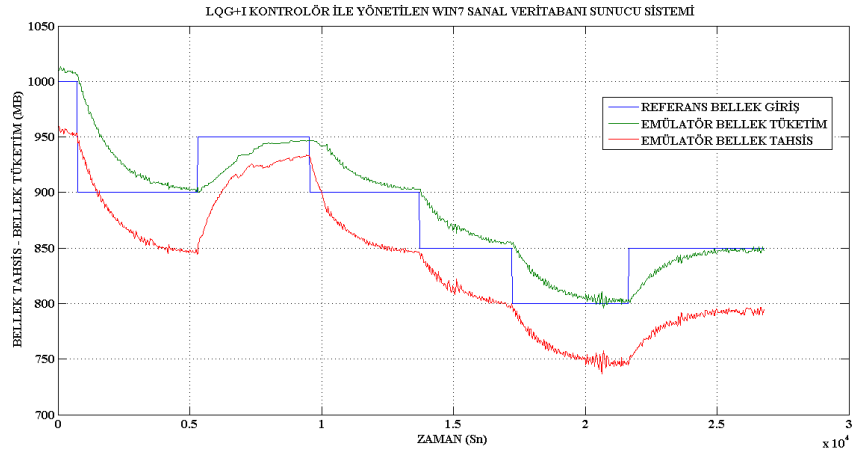
Bellek tahsis-bellek kullanım SISO sistemine ilişkin elde edilen deney sonuçları Şekil 6.23-27 şekil numaraları arasındaki grafiklerle gösterilmiştir.



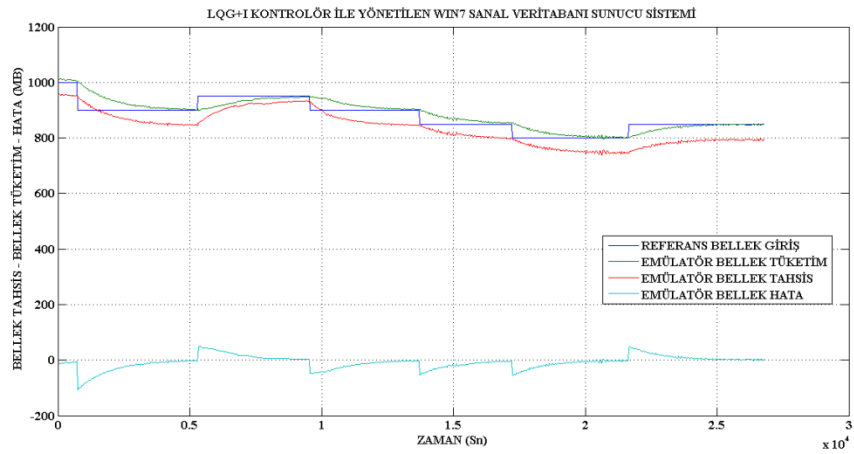
Şekil 6.23 : Bellek SISO sistem için emülör ortamında referans değeri ve bellek tahsis gösterimi.



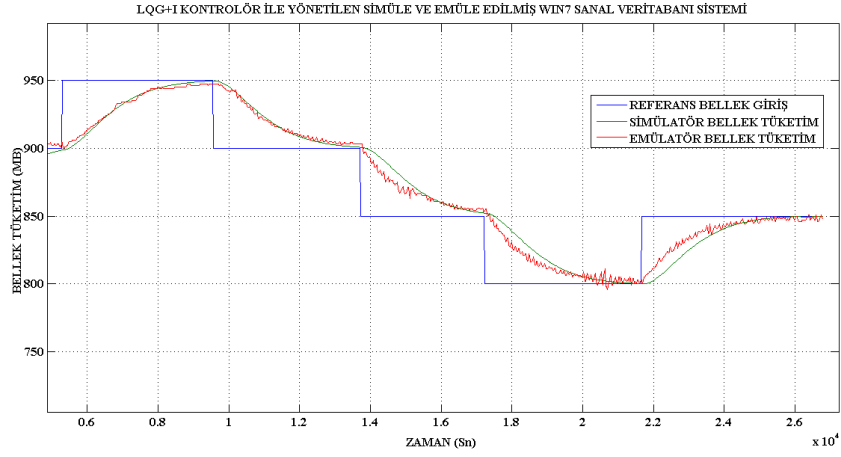
Şekil 6.24 : Bellek SISO sistem için emülatör ortamında referans değer ve bellek kullanım gösterimi.



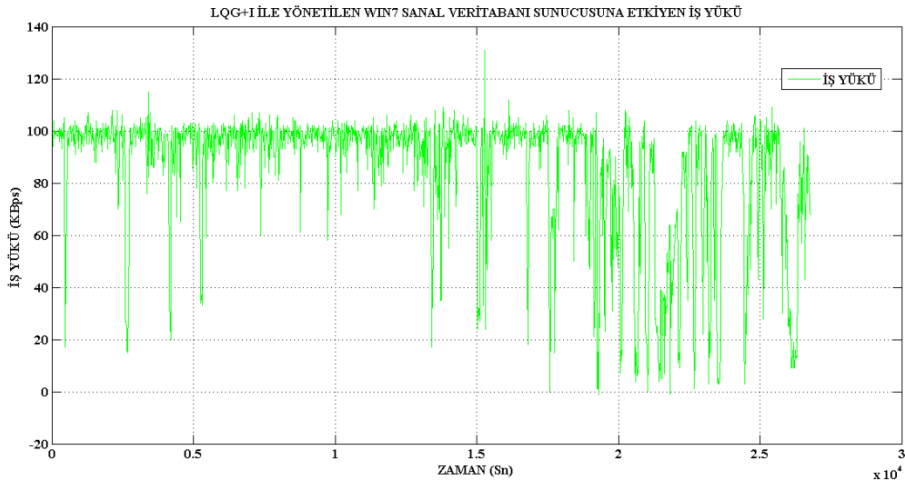
Şekil 6.25 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis ve bellek kullanım gösterimi.



Şekil 6.26 : Bellek SISO sistem için emülatör ortamında referans değer, bellek tahsis, bellek kullanım ve referans ile sistem çıkışı olan bellek kullanımı arasındaki hata bilgilerinin gösterimi.



Şekil 6.27 : Bellek SISO sistemi için simülasyon ve emülasyon ortamları için referans değeri, bellek kullanım gösterimi.

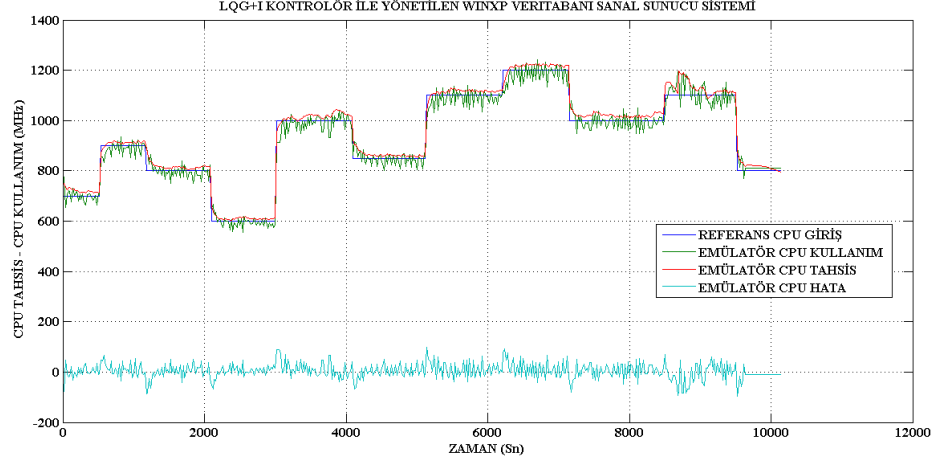


Şekil 6.28 : Bellek SISO sistemine etkiyen iş yükü.

6.4 Win7 ve WinXP Veri tabanı Sanal Sunucuları Paralel Çalışma Deneyleri

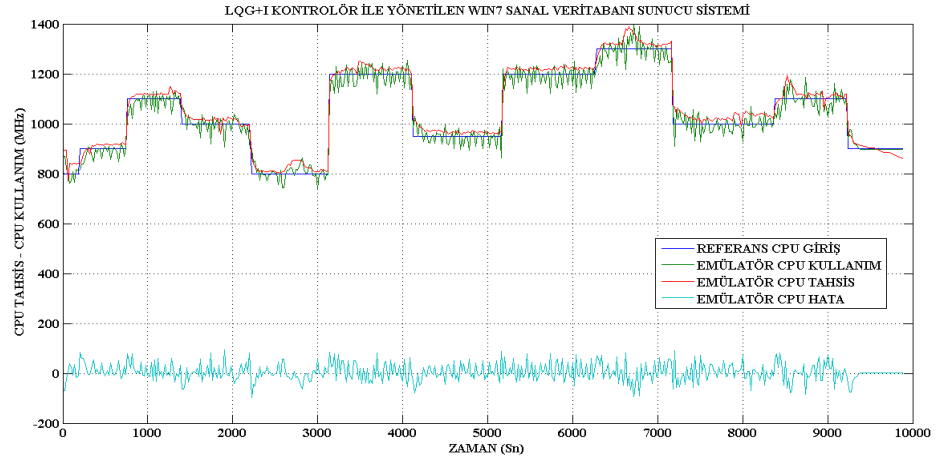
Deney 1: WinXP ve Win7 CPU sonuçları

Paralel çalışma durumunda LQG+I kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları Şekil 6.29 ile gösterilmiştir. Bu deney için RMSE WinXP CPU **31,3935** olarak elde edilmiştir.



Şekil 6.29 : WinXP sanal makinası emülatör CPU sonuçları.

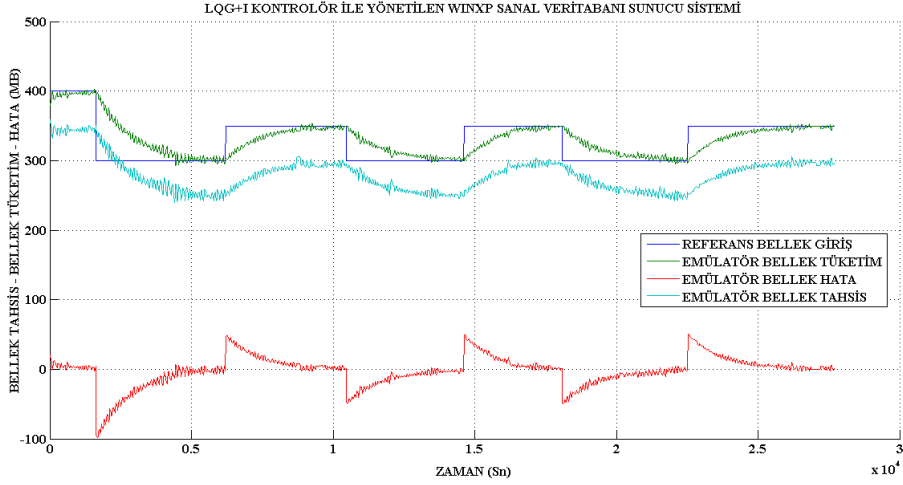
Paralel çalışma durumunda LQG+I kontrolör ile yönetilen Win7 makinasına ilişkin CPU sonuçları Şekil 6.30 ile gösterilmiştir. Bu deney için RMSE Win7 CPU **35,5313** olarak elde edilmiştir.



Şekil 6.30 : Win7 sanal makinası emülatör CPU sonuçları.

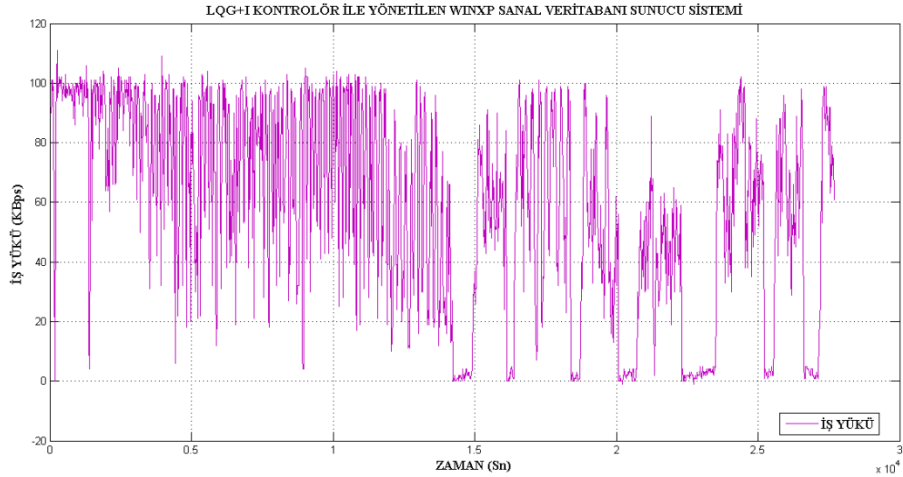
Deney 2: WinXP ve Win7 bellek sonuçları

Paralel çalışma durumunda LQG+I kontrolör ile yönetilen WinXP makinasına ilişkin bellek sonuçları Şekil 6.31 ile gösterilmiştir. Bu deney için RMSE WinXP bellek **21,7717** olarak elde edilmiştir.



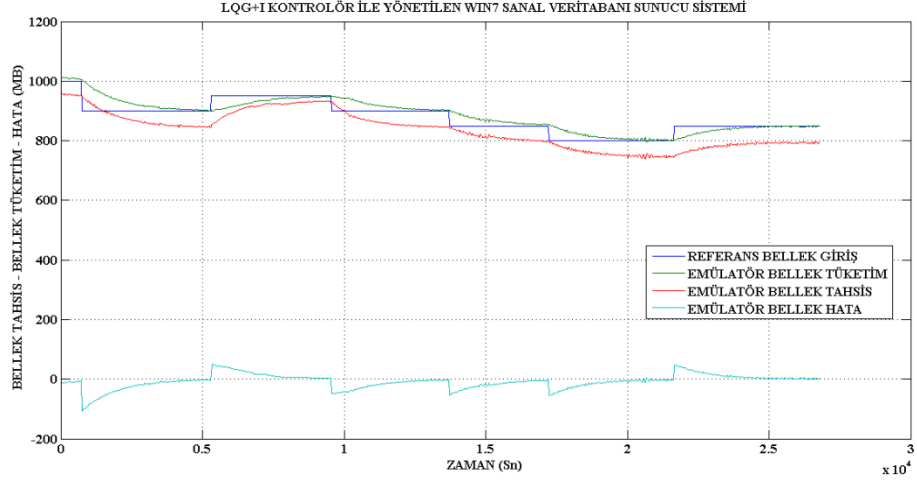
Şekil 6.31 : WinXP sanal makinası emülatör bellek sonuçları.

Paralel çalışma durumunda LQG+I kontrolör ile yönetilen WinXP makinasına etkiyen iş yükü Şekil 6.32 ile gösterilmiştir.



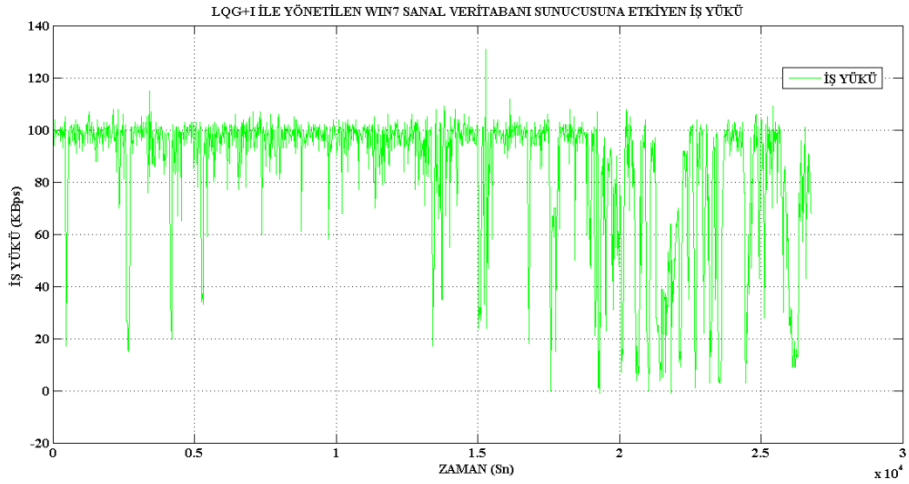
Şekil 6.32 : WinXP sanal makinasına etkiyen iş yükü.

Paralel çalışma durumunda LQG+I kontrolör ile yönetilen Win7 makinasına ilişkin bellek sonuçları Şekil 6.33 ile gösterilmiştir. Bu deney için RMSE Win7 bellek **24,9136** olarak elde edilmiştir.



Şekil 6.33 : Win7 sanal makinası emülatör bellek sonuçları.

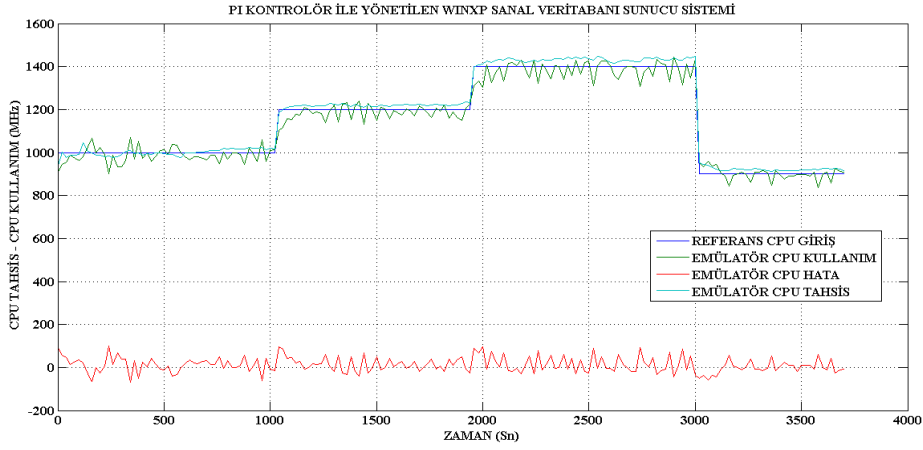
Paralel çalışma durumunda LQG+I kontrolör ile yönetilen Win7 makinasına etkiyen iş yükü Şekil 6.34 ile gösterilmiştir.



Şekil 6.34 : Win7 sanal makinasına etkiyen iş yükü.

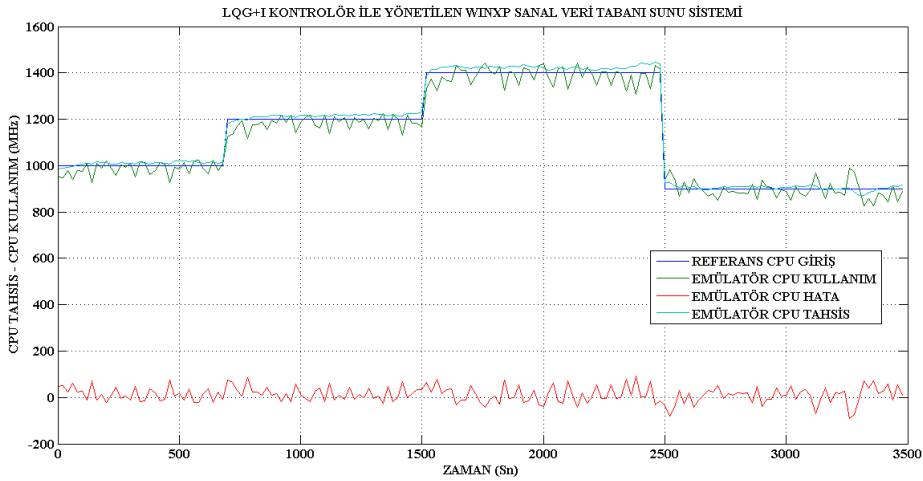
Deney 3: WinXP CPU sonuçları

WinXP makinesi tek olarak çalıştığı durumda PI kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları Şekil 6.35 ile gösterilmiştir. Bu deney için RMSE WinXP **37,4980** olarak elde edilmiştir.



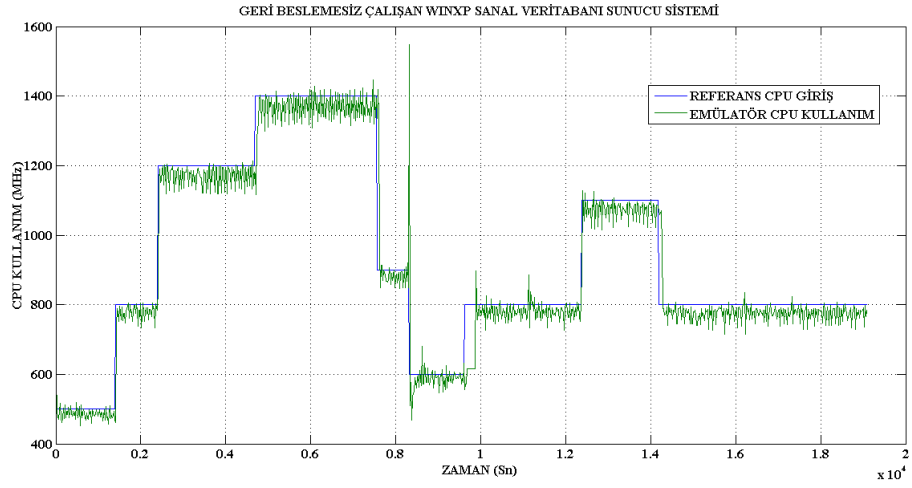
Şekil 6.35 : PI kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları.

WinXP makinası tek olarak çalıştığı durumda LQG+I kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları Şekil 6.36 ile gösterilmiştir. Bu deney için RMSE WinXP **35,8894** olarak elde edilmiştir.

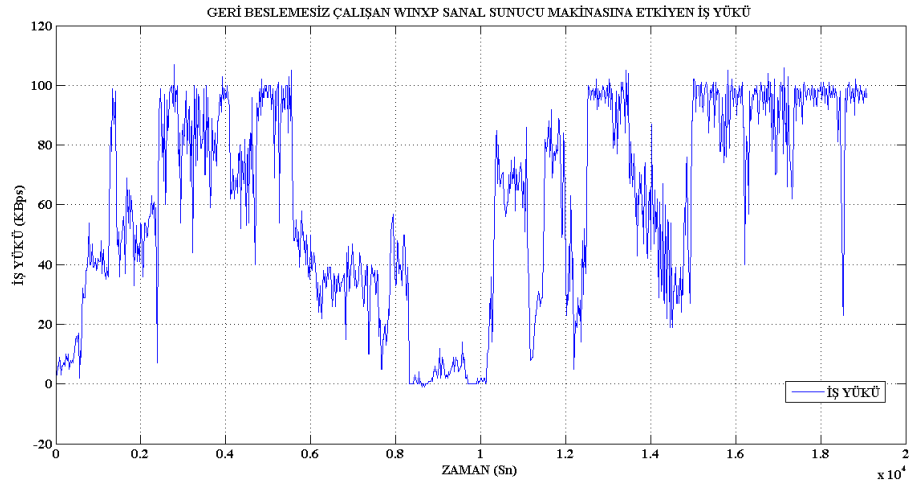


Şekil 6.36 : LQG+I kontrolör ile yönetilen WinXP makinasına ilişkin CPU sonuçları.

Deney 4: WINXP ve WIN7 veri tabanı sanal makinaları geri beslemesiz çalıştığı durumda CPU sistem çıkışının aldığı değerler ve sisteme etkiyen iş yükü Şekil 6.37-40 ile gösterilmiştir. Geri beslemesiz çalışma durumunda WinXP makinasına ilişkin RMSE WinXP CPU **63,8190** olarak elde edilmiştir.

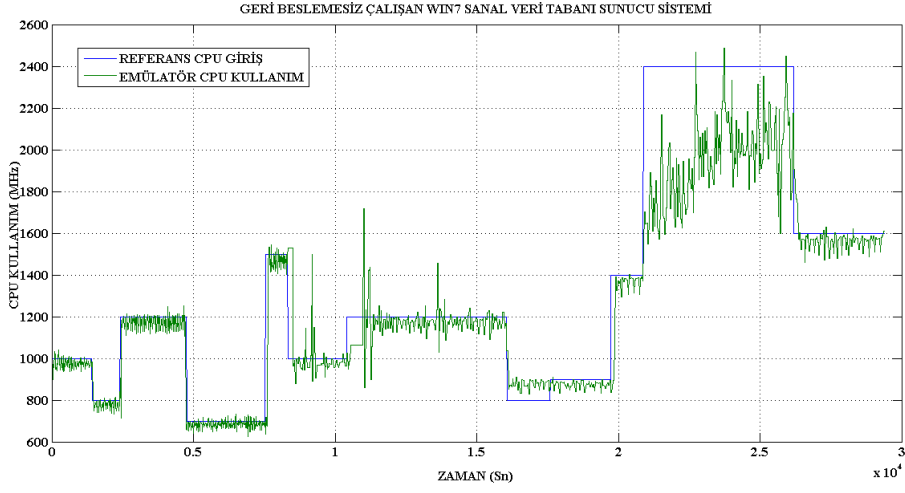


Şekil 6.37 : WinXP sanal makinası emülatör CPU sonuçları.

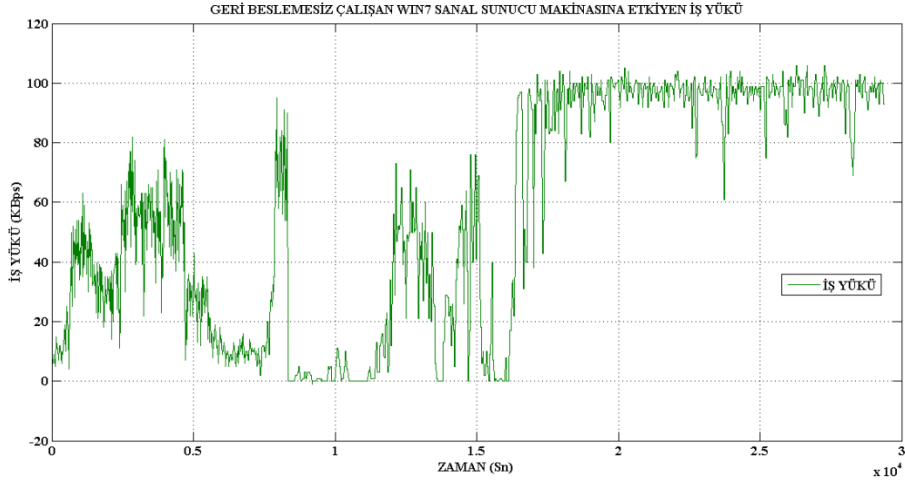


Şekil 6.38 : WinXP sanal makinasına etkiyen iş yükü.

Geri beslemesiz çalışma durumunda Win7 makinasına ilişkin RMSE Win7 CPU 233,2767 olarak elde edilmiştir.

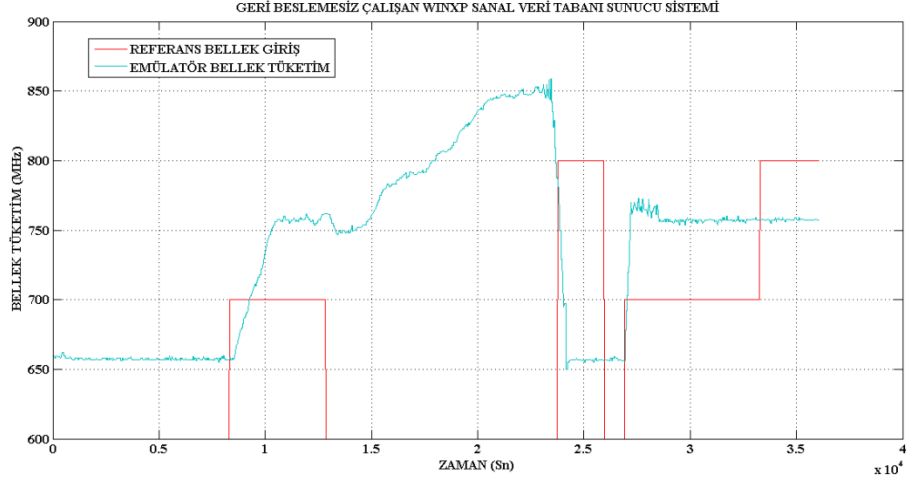


Şekil 6.39 : Win7 sanal makinası emülatör CPU sonuçları.

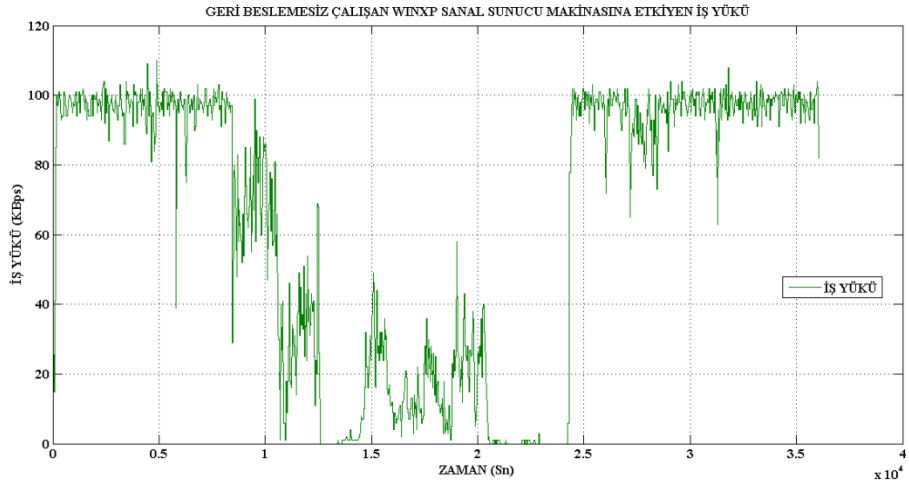


Şekil 6.40 : Win7 sanal makinasına etkiyen iş yükü.

Deney 5: WINXP ve WIN7 veri tabanı sanal makinaları geri beslemesiz çalıştığı durumda bellek sistem çıkışının aldığı değerler ve sisteme etkiyen iş yükü Şekil 6.41-44 ile gösterilmiştir. Geri beslemesiz çalışma durumunda WinXP makinasına ilişkin RMSE WinXP CPU **126,6529** olarak elde edilmiştir.

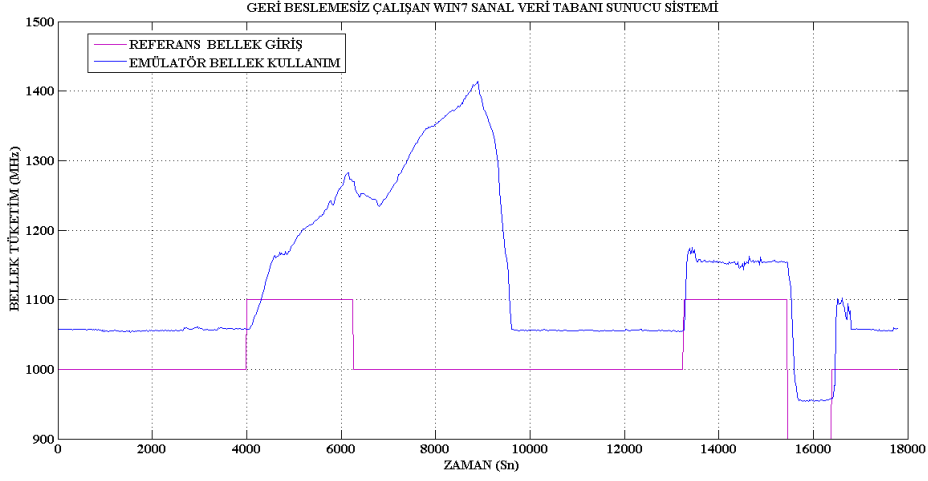


Şekil 6.41 : WinXP sanal makinası emülatör bellek sonuçları.

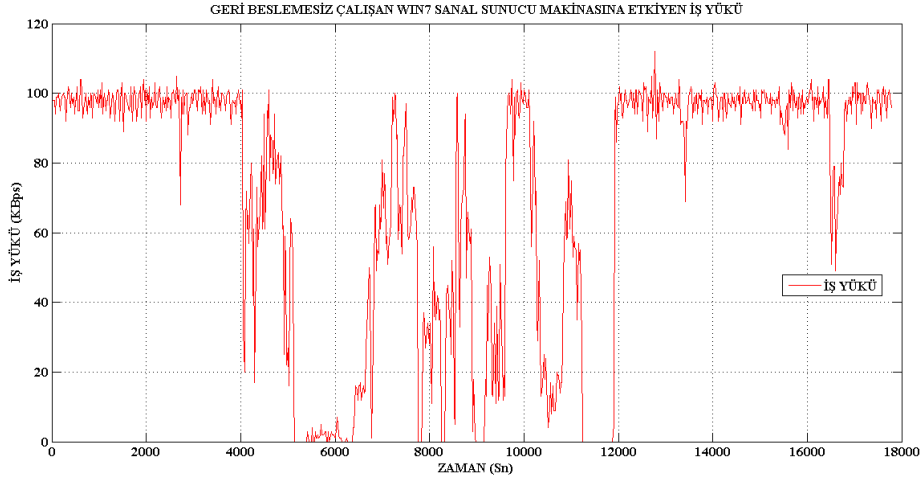


Şekil 6.42 : WinXP sanal makinasına etkiyen iş yükü.

Geri beslemesiz çalışma durumunda Win7 makinasına ilişkin RMSE Win7 CPU **151,4308** olarak elde edilmiştir.



Şekil 6.43 : WinXP sanal makinası emülatör bellek sonuçları.



Şekil 6.44 : Win7 sanal makinasına etkiyen iş yükü.

6.5 Deney Sonuçlarının Değerlendirilmesi

LQG servo kontrolörler ile tasarlanan geri besleme sistemde, sistem çıkışları referans değerleri izlemektedir, bu durum grafiklerle gösterilmiştir. Emülatör ve simülatörden elde edilen veri ile oluşturulan grafiklerde, PI kontrolör ile yönetilen sistem ile LQG servo kontrolör ile yönetilen sistem arasındaki performans ile ilgili farklılık gözlemlenmiştir. PI ve LQG+I kontrolörler ile sistemin kontrol edildiği durum için ve geri beslemesiz durum için sistem çıkışında görülen hataların karesel ortalamasına göre kıyaslama Çizelge 6.7 ile gösterilmiştir. LQG kontrolör ile durum geri beslemesiyle yönetilen sistem, PI kontrolör ile yönetilen sisteme kıyasla çıkışın girişi izlemedeki başarımı ve kontrol işaretinin minimum enerjiyle etkin kullanımı

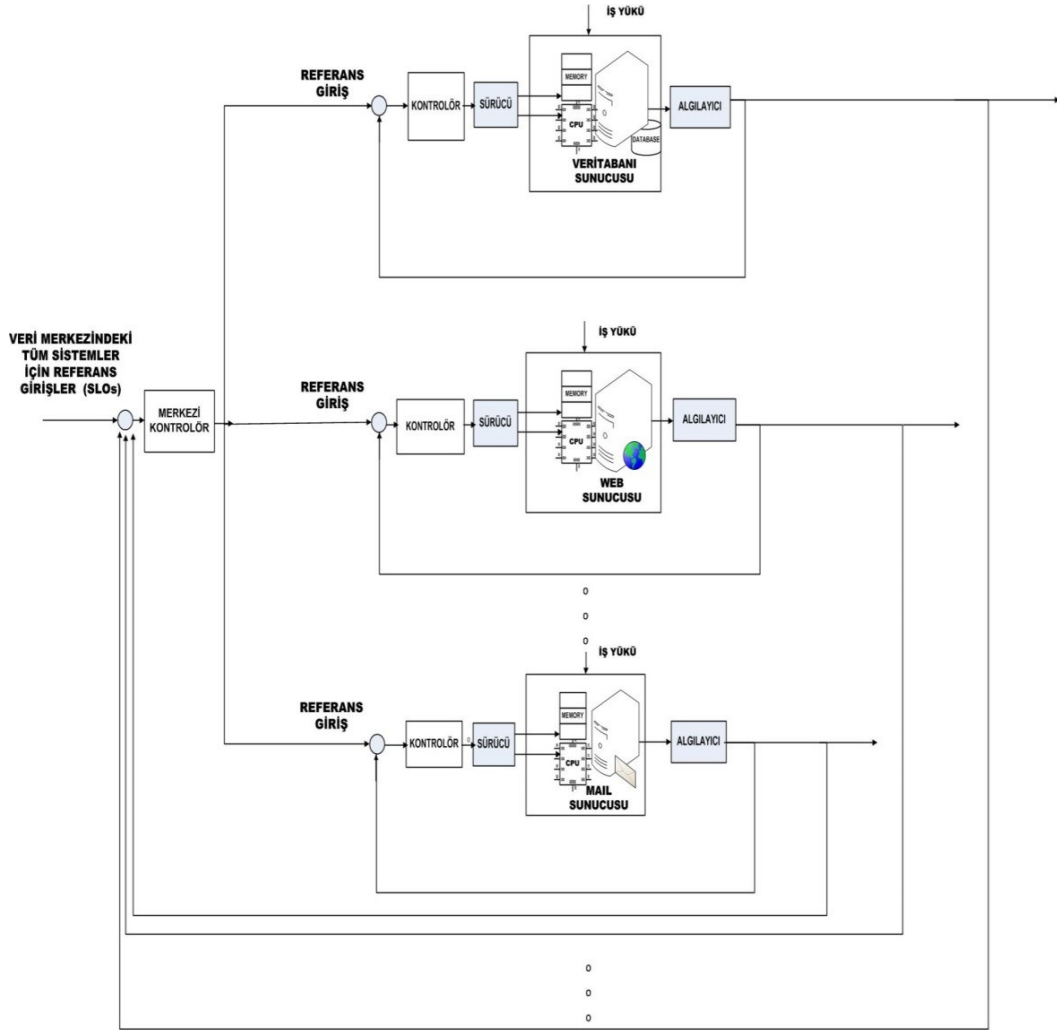
sebebiyle daha iyi performans göstermiştir. Deney sonuçları; her iki kontrol yaklaşımının, sanal sunucu makinasının kaynaklarının etkin kullanımını sağladığını ve bulut ortamındaki servis kalitesini arttırdığını göstermiştir.

Çizelge 6.7 : Geri beslemeli durum ve geri beslemeli olmayan durum için hataların karesel ortalamasına göre kıyaslama.

	LQG+I KONTROLÖR RMSE	PI KONTROLÖR RMSE	GERİ BESLEMESİZ RMSE
WINXP BELLEK	21,7717	25,5955	126,6529
WIN7 BELLEK	24,9136	25,6602	151,4308
WINXP CPU	31,3935	35,5580	63,8190
WIN7 CPU	35,5313	38,5679	233,2767

7. SONUÇLAR

Bu doktora tezinde kontrol kuramının, veri tabanı sanal sunucusu kaynak kullanımı probleminde uygulanması ile ilgili bir çalışma yapılmıştır. Bulut ortamındaki veri tabanı sanal sunucu makinasının kaynak yönetim mimarisi sunulmuştur. İlk olarak sistemi kontrol etmek için sistem giriş çıkışlarına ilişkin matematiksel model elde edilmiştir. Emülatörden elde edilen giriş-çıkış verisine farklı sistem tanıma yöntemleri uygulanarak sistemin hem MIMO hem de çoklu SISO modelleri oluşturulmuştur. Veri tabanı sunucu sisteminin modelini elde etmede, temel yasaların yokluğu sebebiyle deney ve gözlem yolu izlenmiştir. DBVM makinasının matematik modelini elde etmek için özellikle yoğun iş yükü altındaki DBVM üzerinde farklı genlik ve frekanslı giriş işaretleriyle pek çok deney gerçekleştirilmiştir. CPU tahsis, bellek tahsis sistem girişleri ile CPU kullanım ve bellek tüketim sistem çıkışlarındaki ilişki belli bir yaklaşıklık ile lineer olarak elde edilebileceğini teklif edilmiştir. CPU tahsis-CPU kullanım SISO modeli ve bellek tahsis-bellek tüketim SISO modeli, sistem dinamiklerini yansıtmada MIMO sistem modeli kadar iyi bir başarımlı göstermiştir. Bununla birlikte, SISO sistem modellerinin derecesi MIMO sistem model derecesinden düşük olduğundan ve düşük mertebeli sistemleri kontrol etmek daha kolay olduğundan veri tabanı sanal sunucusunun kontrolünde SISO modeller tercih edilmiştir. İkinci olarak, geri besleme çevrimini kapatmak üzere algılayıcı ve sürücüler tasarlanmış ve gerçekleştirilmiştir. Son olarak, her bir SISO sistem için ayrı olarak PI ve integral etkili LQG servo kontrolörler tasarlanmıştır. Geri besleme kontrol sistemi, veri tabanı sanal sunucu makinasının kaynak ihtiyacına göre kaynak tahsisi yapılmasını garanti etmeye çalışır. Deney sonuçları, kontrol teorisi yaklaşımının veri tabanı sanal makinasının kaynak kullanımında hem verimliliği hem de servis kalitesini arttırdığını göstermiştir.



Şekil 7.1 : Veri merkezindeki tüm sunucuların CPU ve bellek kaynaklarını yöneten merkezi kontrolör.

İleride gerçekleştirilecek iş adımı olarak, veri merkezindeki Web sunucu, mail sunucu, dosya sunucu gibi diğer sanal uygulama sunucularının CPU ve bellek kaynaklarının modellenmesi ve kontrolü için çalışmalarımızı ilerletmek istiyoruz. Böylelikle veri merkezindeki, tüm CPU ve bellek kaynakları ihtiyaçlar dikkate alınarak farklı uygulama sunucuları arasında paylaştırılabilecektir. Veri merkezindeki farklı uygulama sanal sunucuları kontrol eden genel geri besleme sistemi Şekil 7.1 ile gösterilebilir. Gelecekte ek olarak, veri merkezindeki sanal uygulama sunucularının CPU ve bellek kaynaklarının yanı sıra katı disk ve ağ kaynaklarını da adaptif, H_∞ kontrol gibi farklı kontrol tekniklerini dikkate alarak kontrol etmeyi planlamaktayız.

KAYNAKLAR

- [1] **Url-1** <<http://www.mathworks.com/products/sysid>>, **MATLAB** System Identification Toolbox, 2013.
- [2] **Url-2** <<http://homes.esat.kuleuven.be/~smc/daisy/daisydata.html>>, Tanıma veri tabanı, Alındığı tarih: 06.10.2010.
- [3] **Url-3** <ftp://ftp.esat.kuleuven.be/pub/SISTA/data/timeseries/internet_traffic.txt>, One hour of internet traffic between the Lawrence Berkeley Laboratory and the rest of the world, Alındığı tarih: 06.10.2010.
- [4] **Padala P., Shin K. G., Zhu X., Uysal M., Wang Z., Singhal S., ... Salem K.** (2007). Adaptive control of virtualized resources in utility computing environments. In 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, EuroSys'07. New York, NY, USA: ACM.p. 289-302. doi: <http://dx.doi.org/10.1145/1272996.1273026>.
- [5] **Url-4** <<http://www.VMware.com/>>, VMware: Virtualization via Hypervisor, Virtual Machine & Server Consolidation, Alındığı Tarih 01.02.2013
- [6] **Url-5** <<http://www.microsoft.com/SQLserver/2008/tr/tr/r2.aspx>>, Microsoft SQL server 2008, Alındığı Tarih: 01/02/2013.
- [7] **Gündem T. İ. ve Armağan Ö.** (2006). Efficient storage of healthcare data in XML-based smart cards. J Computer Methods and Programs in Biomedicine 2006; 81(1):26-40. doi:<http://dx.doi.org/10.1016/j.cmpb.2005.10.007>.
- [8] **Chandra A., Gong W. ve Shenoy P.** (2003). Dynamic resource allocation for shared data centers using online measurements. Newsletter ACM SIGMETRICS Performance Evaluation Review, Haziran 2003; 31(1):300-1. doi:<http://dx.doi.org/10.1145/885651.781067>.
- [9] **Padala P., Hou K.-Y., Shin K. G., Zhu X., Uysal M., Wang Z., ... Merchant A.** (2009). Automated control of multiple virtualized resources. In: Proceedings of the 4th ACM European conference on Computer systems, EuroSys'09. New York, NY, USA: ACM; p.13–26. doi:<http://dx.doi.org/10.1145/1519065.1519068>.
- [10] **Ljung L.** (1999). System Identification: Theory for the User (2nd Edition), Prentice Hall.
- [11] **Hellerstein J.L., Diao Y., Parekh S. ve Tilbury D. M.** (2004). Feedback Control of Computing Systems. Wiley-Interscience, 2004.
- [12] **Keshav S.** (1991). A control-theoretic approach to flow control. In Proceedings of ACM SIGCOMM '91, Eylül 1991.
- [13] **Li K., Shor M. H., Walpole J., Pu C. ve Steere D. C.** (2001). Modeling the effect of shortterm rate variations on tcp-friendly congestion control behavior. In Proceedings of the American Control Conference, pp. 3006–3012, 2001.
- [14] **Hollot C., Misra V., Towsley D. ve Gong W. B.** (2001). A control theoretic analysis of RED. In Proceedings of IEEE INFOCOM '01, Anchorage, AK, Nisan 2001.

- [15] **Benmohamed L. ve Meerkov S. M.** (1993). Feedback control of congestion in packet switching networks: the case of a single congested node. *IEEE Transactions on Networking*, 1(6), Aralık 1993.
- [16] **Johansson P. ve Nilsson A. A.** (1997). Discrete time stability analysis of an explicit rate algorithm for the abr service. In *IEEE ATM Workshop*, pages 229–350, 1997.
- [17] **Diao Y., Gandhi N., Hellerstein J.L., Parekh S. ve Tilbury D.M.** (2002). Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. In: *Network Operations and Management Symposium, NOMS'02*. IEEE/IFIP, p.219–34. doi:http://dx.doi.org/10.1109/ NOMS.2002.1015566.
- [18] **Parekh S., Gandhi N., Hellerstein J., Tilbury D., Jayram T. ve Bigus J.** (2002). Using Control Theory to Achieve Service Level Objectives In *Performance Management. Real-Time Systems Volume 23, Numbers 1-2*, 127-141, DOI: 10.1023/ A:1015350520175.
- [19] **Lui S., Xue L., Ying Lu ve Abdelzaher T.** (2002). Queueing model based network server performance control; *Real-Time Systems Symposium, 2002. RTSS, 23rd IEEE*.pp. 81 - 90,2002.
- [20] **Chenyang Lu, Lu Y., Abdelzaher T.F., Stankovic J.A. ve Sang Hyuk Son.** (2006). Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers. J in *IEEE Transactions on Parallel and Distributed Systems*, Eylül 2006; 17(9):1014-27. doi:http://dx.doi.org /10.1109/ TPDS.2006.123.
- [21] **Årzén K.-E., Robertsson A., Henriksson D., Johansson M., Hjalmarsson H. ve Johansson K. H.** (2006). ARTIST2 roadmap on control of computing systems. *Newsletter ACM SIGBED Review - Special issue on major international initiatives on real-time and embedded systems*, Temmuz 2006; 3(3):11-20. doi:http://dx.doi.org/ 10.1145/1164050.1164053.
- [22] **Abdelzaher T., Shin K. G.ve Bhatti N.** (2002). Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach. J in *IEEE Transactions on Parallel and Distributed Systems*, Ocak 2002; 13(1):80-96. doi:http://dx.doi.org/ 10.1109/71.980028.
- [23] **Gandhi N., Tilbury D.M., Parekh S. ve Hellerstein J.** (2001). Feedback control of a Lotus Notes server: modeling and control design. In: *Proceedings of American Control Conference*, vol. 4. Arlington, VA, USA; 2001. p. 3000-5. Doi: http:// dx.doi.org/ 10.1109/ACC.2001.946372.
- [24] **Abdelzaher T., Lu Y., Zhang R. ve Henriksson D.** (2004). Practical Application of Control Theory to Web Services. In: *Proceedings of American Control Conference*, vol. 3. Boston, MA, USA; p. 1992-97.
- [25] **Gandhi N., Tilbury D.M., Diao Y., Hellerstein J. ve Parekh S.** (2002). MIMO control of an Apache Web server: modeling and controller design. In: *Proceedings of American Control Conference, ACC'02*, vol.6. p. 4922-27. doi:http://dx.doi.org/ 10.1109/ ACC.2002.1025440.
- [26] **Diao Y., Hellerstein J.L., Storm A.J., Surendra M., Lightstone S., Parekh S., ... Garcia-Arellano C.** (2004). Using MIMO linear control for load balancing in computing systems. In: *Proceedings of the American Control Conference, ACC'04*, vol. 3. Boston, MA, USA; Haziran 2004. p. 2045-50.

- [27] **Lu Y., Abdelzaher T.F. ve Saxena A.** (2004). Design, implementation, and evaluation of differentiated caching services. *J IEEE Transactions on Parallel and Distributed Systems* Mayıs 2004; 15(5):440-52. doi:http://dx.doi.org/10.1109/TPDS.2004.1278101.
- [28] **Abdelzaher T.F. ve Lu C.** (2000). Modeling and performance control of Internet servers. *Decision and Control*. In: *Proceedings of the 39th IEEE Conference*, vol. 3. Sydney, NSW, Australia; Aralık 2000. p. 2234–39. doi:http://dx.doi.org/10.1109/CDC.2000.914129.
- [29] **Lu C., Abdelzaher T.F., Stankovic J. ve Son S.** (2001). A feedback control approach for guaranteeing relative delays in Web servers. In: *Seventh IEEE Real-Time Technology and Applications Symposium, RTTAS'01*. Taipei; p. 51-62. doi:http://dx.doi.org/10.1109/RTTAS.2001.929865.
- [30] **Url-6** <<http://www.apache.org>>, Apache Software Foundation, Alındığı tarih: 31.03.2012.
- [31] **Lu Y., Abdelzaher T. ve Tao G.** (2003). Direct Adaptive Control of A Web Cache System. *2003 American Control Conference*; Denver, CO; USA; 4-6 Haziran 2003. pp. 1625-1630.
- [32] **Banga G. ve Druschel P.** (1999). Measuring the capacity of a Web server under realistic loads, *World Wide Web Volume 2*, Numbers 1-2, 69-83, DOI: 10.1023/A:1019292504731.
- [33] **Ying Lu, Abdelzaher T., Lu C. ve Tao G.** (2002). An adaptive control framework for QoS guarantees and its application to differentiated caching. *Quality of Service*. Tenth IEEE International Workshop on, pp. 23 - 32.
- [34] **Allen A. O.** (1990). *Probability, Statistics, and Queueing Theory*, (2nd ed.) Academic Press, San Diego, CA, 1990.
- [35] **Kleinrock L.** (1975). *Queueing Systems*. (2nd ed.) Wiley-Interscience, New York, 1975.
- [36] **Zhang L., Zhao Z., Shu Y., Wang L. ve Yang O.** (2002). Load balancing of multipath source routing in ad hoc networks. In *International Conference on Communications*, 2002.
- [37] **Iyengar A., Challenger J., Dias D. ve Dantzig P.** (2000). High-performance Web site design techniques. *IEEE Internet Computing*, 4(2):17–26, Şubat 2000.
- [38] **Abdelzaher T. ve Shin K. G.** (1999). QoS provisioning with qContracts in Web and multimedia servers. *Real-Time Systems Symposium*, pp. 44–53, 1999.
- [39] **Goel A., Shor M. H., Walpole J., Steere D. ve Pu C.** (2001) Using feedback control for a network and CPU resource management application. *American Control Conference*, pp. 2974–2980, Haziran 2001.
- [40] **Shinners S. M.** (1998). *Modern Control System Theory and Design*. Second edition, New York: John Wiley & Sons Press; 1998.
- [41] **Parekh S., Rose K., Diao Y., Chang V., Hellerstein J., Lightstone S., ... Huras M.** (2004). Throttling utilities in the IBM DB2 universal database server. In: *Proceedings of American Control Conference*, vol. 3. Boston/MA, USA: 2004. p. 1986-91.
- [42] **Kang K. D., Oh J. ve Son S. H. Chronos.** (2007). Feedback Control of a Real Database System Performance. In: *Proceedings of the 28th IEEE International Real Time Systems Symposium, RTSS*. Tucson, AZ: IEEE; 2007. p. 267-76. Doi: http://dx.doi.org/10.1109/RTSS.2007.16.

- [43] **Kang W, Son S.H., Stankovic J.A. ve Amirijoo M.** (2007). I/O-Aware Deadline Miss Ratio Management in Real-Time Embedded Databases. In: The 28th IEEE International Real-Time Systems Symposium, RTSS'07. Tucson, AZ; Aralık 2007. p. 277- 87. doi:<http://dx.doi.org/10.1109/RTSS.2007.19>.
- [44] **Woochul K., Son S.H., Stankovic J.A. ve Amirijoo M.** (2007). I/O-Aware Deadline Miss Ratio Management in Real-Time Embedded Databases. Real-Time Systems Symposium, RTSS, 28th IEEE International, pp. 277 - 287, Tucson, AZ, 3-6 Aralık 2007.
- [45] **Kang K. D., Son S.H. ve Stankovic J.A.** (2004). Managing deadline miss ratio and sensor data freshness in real-time databases. Knowledge and Data Engineering, IEEE Transactions on, pp. 1200 - 1216, Ekim 2004.
- [46] **Karamanolis C., Karlsson M. ve Zhu X.** (2005). Designing controllable computer systems. In: Proceedings of the 10th conference on Hot Topics in Operating Systems, HOTOS'05. Berkeley, CA, USA: ACM; 2005. p. 9-9.
- [47] **Diao Y., Hellerstein J.L., Parekh, S., Griffith R., Kaiser G.E. ve Phung D.** (2005). A control theory foundation for self-managing computing systems. J in IEEE Journal on Selected Areas in Communications, 2005; 23(12):2213-22.
- [48] **Diao Y., Hellerstein J.L., ve Parekh, S.** (2006). Control of Large Scale Computing Systems. Newsletter ACM SIGBED Review, Nisan 2006; 3(2):17-22. Doi: <http://dx.doi.org/10.1145/1143489.1143494>.
- [49] **Lu C, Wang X. ve Koutsoukos X.** (2005). Feedback Utilization Control in Distributed Real-Time Systems with End-to-End Tasks. J IEEE Transactions on Parallel and Distributed Systems Haziran 2005; 16(6):550-61. doi:<http://dx.doi.org/10.1109/TPDS.2005.73>.
- [50] **Sha L., Liu, X. Lu Y. ve Abdelzaher T.** (2002). Queuing model based network server performance control. In: The 23rd IEEE Real-Time Systems Symposium, RTSS'02. p. 81-90. doi:<http://dx.doi.org/10.1109/REAL.2002.1181564>.
- [51] **Tipsuwan Y. ve Chow M.** (2003). Control methodologies in networked control systems Control Engineering Practice 11 (2003) 1099–1111.
- [52] **Wei Z., Branicky M.S. ve Phillips S.M.** (2001). Stability of networked control systems. Control Systems, IEEE, pp. 21 Issue:1, Şubat 2001.
- [53] **Stankovic J.A., Lu C., Son S.H. ve Tao G.** (1999). The case for feedback control real-time scheduling. Real-Time Systems, Proceedings of the 11th Euromicro Conference on, pp. 11 - 20, York , UK, 09 Haziran - 11 Temmuz 1999.
- [54] **Lu C., Stankovic J.A., Son S.H. ve Tao G.** (2000). Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms. Real-Time Systems Volume 23, Numbers 1-2, 85-126, DOI: 10.1023/A:1015398403337.
- [55] **Lu C., Wang X. ve Gill C.** (2003). Feedback Control Real-Time Scheduling in ORB Middleware, Real-Time and Embedded Technology and Applications Symposium, IEEE, pp. 37, Ninth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03).
- [56] **Steere D.C., Shor M.H., Goel A., Walpole J. ve Calton Pu.** (2000). Control and modeling issues in computer operating systems: resource management for real-rate computer applications. Decision and Control, Proceedings of the 39th IEEE Conference on, pp. 2212 - 2221 vol.3, Sydney, NSW , Australia, 12 - 15 Aralık 2000.

- [57] **Zhu X, Uysal M., Wang Z., Singhal S., Merchant A., Padala P., ... Shin K.** (2009). What Does Control Theory Bring to Systems Research? Newsletter ACM SIGOPS Operating Systems Review, Ocak 2009; 43(1):62-69. doi:<http://dx.doi.org/10.1145/1496909.1496922>.
- [58] **Liu X., Zhu X., Singhal S. ve Arlitt M.** (2005). Adaptive entitlement control of resource partitions on shared servers. In: The 9th International Symposium on Integrated Network Management; Mayıs 2005.
- [59] **Wang Z., Zhu X. ve Singhal S.** (2005). Utilization and SLO-based control for dynamic sizing of resource partitions. In: Proceedings of the 16th IFIP/IEEE Distributed Systems on Operations and Management. Barcelona, Spain: IEEE; Ekim 2005. p. 24-26.
- [60] **An B., Lesser V., Irwin D. ve Zink M.** (2010). Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing. In: Proceedings of the 9th International Conference on Autonomous Agents and Multi-agent Systems. Toronto. p. 981-88.
- [61] **Zhu X., Wang Z. ve Singhal S.** (2006). Utility-driven workload management using nested control design. In: American Control Conference, ACC'06.
- [62] **Armağan Ö. ve Gören-Sümer L.** (2014). Feedback Control for Multi-Resource Usage of Virtualised Database Server. Computers and Electrical Engineering.40(5):1683-1702. Haziran 2014. doi:<http://dx.doi.org/10.1016/j.compeleceng.2014.04.017>.
- [63] **Bruce A. M.** (1997). An Empirical Model of HTTP Network Traffic, Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, p.592, Nisan 09-11, 1997.
- [64] **Leland W. E., Taqqu M. S., Willinger W. ve Wilson D. V.** (1994). On the self-similar nature of Ethernet traffic (extended version), IEEE/ACM Transactions on Networking (TON), v.2 n.1, p.1-15, Şubat 1994.
- [65] **Paxson V. ve Floyd S.** (1994). Wide-area traffic: the failure of Poisson modeling, Proceedings of the conference on Communications architectures, protocols and applications, p.257-268, Ağustos 31-Eylül 02, London, United Kingdom.
- [66] **Crovella M. E. ve Bestavros A.** (1996). Self-similarity in World Wide Web Traffic, IEEE/ACM Transactions on Networking (TON) archive Volume 5 , Issue 6 (December 1997) Pages: 835 - 846 Yayın yılı: 1997.
- [67] **Paxson V.** (1994). Empirically derived analytic models of wide-area TCP connections, IEEE/ACM Transactions on Networking (TON) archive Volume 2 , Issue 4, p.316 - 336, Ağustos 1994.
- [68] **Url-7** <http://www.paessler.com/windows_network_monitor>, Alındığı tarih: 02.11.2010.
- [69] **Url-8** <<http://www.byteplant.com/products/bytemon/bytemon-network-traffic-monitoring.html>>, Alındığı tarih: 01.12.2010.
- [70] **Chen Y., Das A., Qin W., Sivasubramaniam A., Wang Q. ve Gautam N.** (2005). Managing server energy and operational costs in hosting centers, Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Haziran 06-10, 2005, Banff, Alberta, Canada [doi>10.1145/1064212.1064253].
- [71] **Url-9** <[http://technet.microsoft.com/en-us/library/dd364124\(v=WS.10\).aspx](http://technet.microsoft.com/en-us/library/dd364124(v=WS.10).aspx)> Microsoft Internet Information Server (IIS) - Microsoft, Alındığı Tarih: 01.04.2013.

- [72] **Url-10** < <http://www.VMware.com/support/developer/vc-sdk/>> vSphere SDK for .NET Developer's Guide VMware, Alındığı Tarih: 17.04.2011.
- [73] **Url-11** <<http://www.SQLstress.com/>>, A SQL Server Stress Test Tool, Alındığı Tarih: 01.05.2013.
- [74] **Url-12** <<http://www.loadtestingtool.com/index.shtml>>, Load, Stress and Performance Testing for Web Applications, Alındığı tarih: 16.11.2011.
- [75] **Url-13** <<http://www-03.ibm.com/systems/power/software/aix/sysmgmt/wlm/>>, IBM Application Workload Manager Alındığı tarih: 06.06.2012.
- [76] **Url-14** <<http://h71028.www7.hp.com/enterprise/cache/257279-0-0-000-121.html>>, HP Workload Manager, Alındığı tarih: 26.04.2012.
- [77] **Url-15** http://www.dell.com/content/topics/topic.aspx/global/segments/topics/hea/us/streamlined_app_workload?c=us&l=en&cs=RC968571>, Dell Streamlined Application and Workload Management, Alındığı tarih: 26/04/2012.
- [78] **Xiao Z., Song W. ve Chen Q.** (2013). Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment, Parallel and Distributed Systems, IEEE Transactions on , vol.24, no.6, pp.1107-1117, Haziran 2013. doi: 10.1109/TPDS.2012.283.
- [79] **Adamuthe A.C., Pandharpatte R.M. ve Thampi G.T.** (2013). Multiobjective Virtual Machine Placement in Cloud Environment, Cloud & Ubiquitous Computing and Emerging Technologies (CUBE), 2013 International Conference on, On page(s): 8 – 13.
- [80] **Lu F., Pan H., Lei X., Liao X. ve Jin H.** (2013). A Virtualization-Based Cloud Infrastructure for IMS Core Network, Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, On page(s): 25 - 32 Volume: 1, 2-5 Aralık 2013.
- [81] **Jiang W., Zhao Z. ve Laat C. D.** (2013). An Autonomous Security Storage Solution for Data-Intensive Cooperative Cloud Computing, eScience (eScience), 2013 IEEE 9th International Conference on, On page(s): 369 – 372.
- [82] **Domanal S.G. ve Reddy G.R.M.** (2014). Optimal load balancing in cloud computing by efficient utilization of virtual machines, Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on, On page(s): 1 – 4.
- [83] **Mencagli G., Vanneschi M. ve Vespa E.** (2013). Control-theoretic adaptation strategies for autonomic reconfigurable parallel applications on cloud environments, High Performance Computing and Simulation (HPCS), 2013 International Conference on, vol., no., pp.11,18, 1-5 July 2013.doi: 10.1109/HPCSim.2013.6641387.
- [84] **Xiong P.** (2012). Dynamic management of resources and workloads for RDBMS in cloud: a control-theoretic approach. In Proceedings of the on SIGMOD/PODS 2012 PhD Symposium (PhD '12). ACM, New York, NY, USA, 63-68. DOI=10.1145/2213598.2213614 <http://doi.acm.org/10.1145/2213598.2213614>.
- [85] **Sood S. K.** (2013). A Value Based Dynamic Resource Provisioning Model in Cloud, International Journal of Cloud Applications and Computing, v.3 n.1, p.1-12, Ocak 2013.

- [86] **Wu H., Tantawi A. N., Diao Y. ve Wang W.** (2011). Adaptive memory load management in cloud data centers. *IBM J. Res. Dev.* 55, 6 (Kasım 2011), 377-386. DOI=10.1147/JRD.2011.2170869 <http://dx.doi.org/10.1147/JRD.2011.2170869>.
- [87] **Mehrotra R., Abdelwahed S. ve Erradi.** (2013). A Distributed Control Approach for Autonomic Performance Management in Cloud Computing Environment. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC '13)*. IEEE Computer Society, Washington, DC, USA, 269-272. DOI=10.1109/UCC.2013.54 <http://dx.doi.org/10.1109/UCC.2013.54>.
- [88] **Zhang Q., Zhu Q. ve Boutaba, R.** (2011). Dynamic Resource Allocation for Spot Markets in Cloud Computing Environments, *Utility and Cloud Computing (UCC)*, 2011 Fourth IEEE International Conference on , vol., no., pp.178,185, 5-8 Aralık 2011 doi: 10.1109/UCC.2011.33.
- [89] **Xiong P., Chi Y., Zhu S., Moon H. J., Pu C. ve Hacigumus H.** (2011). Intelligent management of virtualized resources for database systems in cloud environment. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE '11)*. IEEE Computer Society, Washington, DC, USA, 87-98. DOI=10.1109/ICDE.2011.5767928 <http://dx.doi.org/10.1109/ICDE.2011.5767928>.
- [90] **Buchmann A., Ivanov T. ve Petrov I.** (2012). A Survey on Database Performance in Virtualized Cloud Environments. *Int. J. Data Warehous. Min.* 8, 3 (Temmuz'12), 1-26. DOI=10.4018/jdwm.2012070101 <http://dx.doi.org/10.4018/jdwm.2012070101>.
- [91] **Mohan N. R. R. ve Raj E. B.** (2012). Resource Allocation Techniques in Cloud Computing -- Research Challenges for Applications, Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on , vol., no., pp.556,560, 3-5 Kasım 2012. doi: 10.1109/CICN.2012.177.
- [92] **Minarolli D. ve Freisleben B.** (2013). Virtual Machine Resource Allocation in Cloud Computing via Multi-Agent Fuzzy Control, *Cloud and Green Computing (CGC)*, 2013 Third International Conference on , vol., no., pp.188,194, Eylül 30 2013- Ekim 2 2013 doi: 10.1109/CGC.2013.35.
- [93] **Pandya P. P. ve Bheda H. A.** (2014). Dynamic Resource Allocation Techniques in Cloud Computing, *International Journal of Advance Research in Computer Science and Management Studies*. Volume 2, Issue 1, Ocak 2014.
- [94] **Manvi S.S. ve Shyam G. K.** (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey, *Journal of Network and Computer Applications*, Volume 41, Mayıs 2014, Pages 424-440, ISSN 1084-8045, <http://dx.doi.org/10.1016/j.jnca.2013.10.004>.

EKLER

EK A: Sanallaştırma Kavramı ve Sanal Mimaride Yer alan Objeler

EK B: Test Ortamı (Emülatör) Konfigürasyonu

EK C: Veri tabanı Sunucusu İçin Geliştirilen Emülatör ve Simülatör Programları

EK D: Çevrimiçi Ödeme Sistemi İstemci-Sunucu Haberleşmesinde Kullanılan Web servis Metotlarından *Borç sorgulama* Metodu Parametre Ve XML Veri Formatı

EK E: Çevrimiçi Ödeme Sistemi Uyarıcı Ve Kontrolör Yazılımı

EK F: Çevrimiçi Ödeme Sistemi Sunucu İle HTTP Bağlantı Kuran, Sorgu, Ödeme, Ödeme İptal gibi İşlemlerin Gerçekleştirilmesini Sağlayan Yazılım

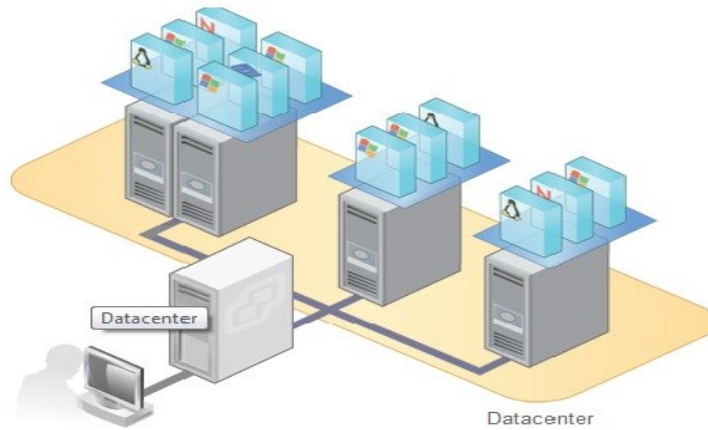
EK A: Sanallaştırma Kavramı Ve Sanal Mimaride Yer alan Objeler

Sanallaştırma (Virtulization) : Fiziksel donanım ile işletim sistemi arasındaki bağlantıyı kesen soyut ara bir katmandır. Sanal makineler sanal alt yapının temel parçalarıdır. Sanallaştırma birbirinden farklı işletim sistemi ve uygulamalar içeren sanal makinaların birbirinden ayrı ve izole olarak aynı fiziksel makina üzerinde çalışmasını sağlar. Bu ayrı yapı Şekil A.1 ile gösterilmiştir.



Şekil A.1 : Sanallaştırma kavramını ifade eden şema.

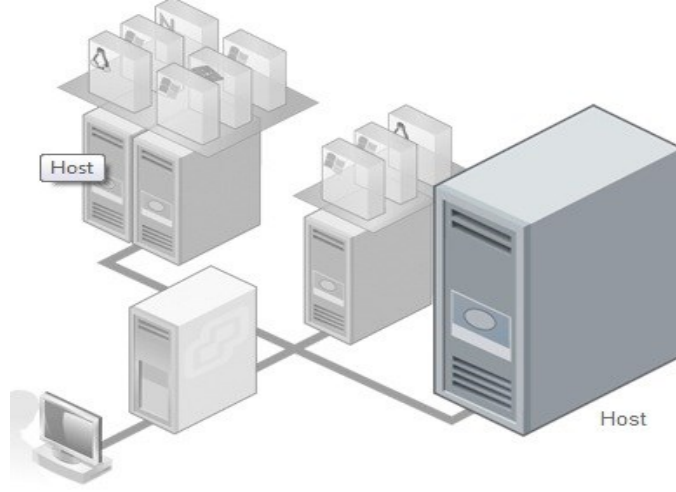
Veri merkezi (Datacenter) : Sanal makine, konak makine gibi objelerin içinde bulunduğu en tepedeki bir taşıyıcı objedir. Konak, küme ya da katalog objeleri veri merkezine eklenebilir. Veri merkezi ile ilgili şema Şekil A.2 ile gösterilmiştir.



Şekil A.2 : Veri merkezi kavramına ilişkin şema.

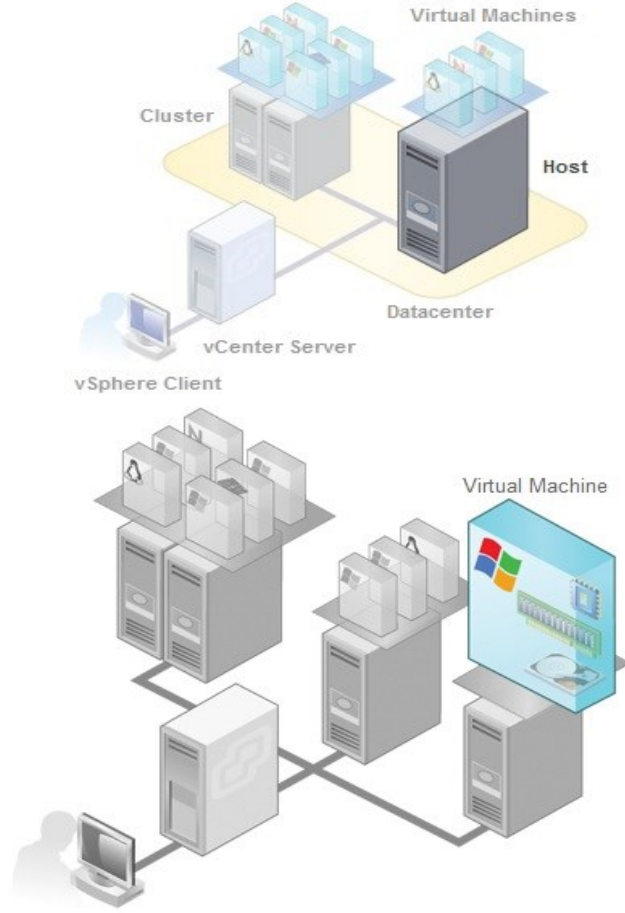
Konak (Host) : Konak, sanal makineleri çalıştırmak için ESX, ESXi gibi sanallaştırma yazılımlarını kullanan bilgisayarlardır. Konak makine, sanal makinelerin kullandığı CPU ve bellek kaynaklarını sağlar. Konak makine aynı zamanda sanal makinaların veri saklama ortamına ve ağa erişimini sağlar. Birden

fazla sanal makine aynı konak veya küme (birden fazla konaktan oluşan yapılar) üzerinde çalışabilir. Konak makinasına ilişkin şema Şekil A.3'te gösterilmiştir.



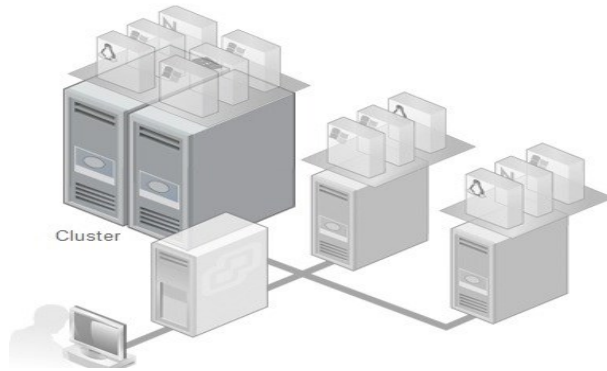
Şekil A.3 : Konak sistemi.

Sanal makine, sanal PC (Virtual machine) : Üzerinde işletim sistemi ve uygulamalar bulunan fiziksel bilgisayar benzeri yazılımsal bilgisayarlardır. Sanal makine üzerinde bulunan işletim sistemine konuk işletim sistemi (guest operating system) denir. Her bir sanal makine fiziksel donanımın aynı fonksiyonları yerine getiren sanal araçlara sahiptir. Sanal makineler CPU, bellek, veri depolama ve ağ kaynaklarını üzerinde bulunduğu konak makinasından alırlar. Sanal makineler emülatör ya da simülatör değildir ve fiziksel makinelerin yaptığı işleri yapabilen gerçek makinelerdir. Sanal makinelerin taşıdığı özelliklere ilişkin şemalar Şekil A.4 'de gösterilmiştir.



Şekil A.4 : Sanal makina ve bu kavramla ilgili şemalar.

Küme (Cluster) : Yönetim ara yüzü ve kaynakları paylaşan konak makinaların oluşturduğu bir gruptur. Kümeye yeni bir konak eklendiği zaman konağın kaynakları kümenin kaynaklarının bir parçası olur. Küme içinde bulunan tüm konak makinalarının kaynaklarına erişir ve bu makinalardaki kaynakları yönetir. Küme ile ilgili şema Şekil A.5 ile gösterilmiştir.



Şekil A.5 : Küme bir ya da birden fazla konak makinasından oluşabilir ve kümede yer alan konakların kaynaklarına sahiptir.

EK B: Test Ortamı (Emülatör) Konfigürasyonu

Bu bölüm bu çalışma kapsamında testlerin yapıldığı yazılım ve donanımlar hakkında bilgi verir.

Sunucu donanımı:

- Hp pavilion dv6 ACPI x64-based PC
- Intel Core 2 adet çekirdek 4 adet mantıksal i7 CPU Q720 1.6GHz işlemci
- Memory: 4.5 GB
- Yerel depolama: ST9640320AS 60 GB katı disk + 140 GB katı disk

ESX Sunucu yazılımı:

- VMware ESXi Server 4.0.0 Build 261974

vCenter Sunucu Yazılımı:

- vCenter Server 4.0

Sanal makina Konfigürasyonu:

- CPU: 1 ya da 2 işlemcili makinalar
- Memory: 2 sanal makinaya toplamda maksimum 3.5 GB bellek
- Bağlantı: vmxnet ile yerel ağ bağlantısı (LAN) ağ bağlantısı
- Konuk işletim sistemi: Windows 7 Professional (32 bit) işletim sistemi (OS), Win XP OS.
- SQL Server: SQL Server 2008 Enterprise Edition SP2 (32 bit)
- IIS Server: Microsoft Internet Information Services IIS 7.5 (32 bit) ve IIS Manager
- VMware PowerCLI versiyon: 4.1.1
- Microsoft Visual Studio Tools for Applications
- SQLStress, WAPT iş yükü ve performans yazılımları

EK C: Veri tabanı Sunucusu İçin Geliştirilen Emülatör ve Simülatör Programları

EK C1: SQL Server yazılım parçaları ve izlek

```
USE [DS]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [SQLStress].[Data](
    [Client] [nvarchar](64) NOT NULL,
    [UserThread] [int] NOT NULL,
    [Id] [uniqueidentifier] ROWGUIDCOL NOT NULL,
    [Server] [nvarchar](64) NOT NULL,
    [Data] [int] NOT NULL,
    [AccessCount] [int] NOT NULL,
    [Username] [nvarchar](256) NULL,
    [Document] [image] NULL,
    [DocumentType] [nvarchar](50) NULL,
    CONSTRAINT [PK_SQLStress] PRIMARY KEY NONCLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
ALTER TABLE [SQLStress].[Data] ADD DEFAULT (newid()) FOR [Id]
GO
ALTER TABLE [SQLStress].[Data] ADD DEFAULT ((1)) FOR
[AccessCount]
GO
ALTER TABLE [SQLStress].[Data] ADD DEFAULT (suser_sname()) FOR
[Username]
GO
=====
USE [DS]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [SQLStress].[SP]
@Client nvarchar(64),      -- Client Nodename
@Server nvarchar(64),      -- Server Nodename
@User int,                 -- Usernumber
@Data int,                 -- Random Usernumber
@Id uniqueidentifier -- Id
```

```

AS
BEGIN
    INSERT
    INTO SQLStress.Data( Client, Server, UserThread, Data, Id )
    VALUES( @Client, @Server, @User, @Data, @Id )
END

```

EK C2 : MATLAB programı

%Sistem tanıma aracı ile CPU ve bellek modellerinin elde edilmesi

```

ident;
modeldata(:,5) = modeldata(:,5)/1024;
sysgir=modeldata(:,1:3);
syscik = modeldata(:,4:5);

```

% Bir örnekleme zamanı gecikmeli bellek SISO sistemi durum uzayı

```

% parametreleri
mod = idss(arx111);
a_mem_111=mod.A;
b_mem_111=mod.B;
c_mem_111=mod.C;
d_mem_111=mod.D;

```

% Gecikmesiz CPU SISO sistemi durum uzayı

```

% parametreleri
mod = idss(arx110);
a_CPU_110=mod.A;
b_CPU_110=mod.B;
c_CPU_110=mod.C;
d_CPU_110=mod.D;

```

%Gecikmesiz CPU tahsis- CPU kullanım SISO sistemi ve bir gecikmeli bellek
%tahsis- bellek kullanım SISO sistem modelleri

```

a_CPU=0.4286;
b_CPU=[0.2425 -0.0623];
c_CPU = 1;
d_CPU = [0.5657 -0.1453];
a_mem=0.9722;
b_mem=[0.0207 0.0005];
c_mem= 1;
d_mem=[0 0];

```

```

sys=ss(a_CPU,b_CPU,c_CPU,d_CPU,20);
sys2=ss(a_mem,b_mem,c_mem,d_mem,20);
[num1, den1]=ss2tf(a_CPU,b_CPU,c_CPU,d_CPU,1);
[num2, den2]=ss2tf(a_CPU,b_CPU,c_CPU,d_CPU,2);
[num3, den3]=ss2tf(a_mem,b_mem,c_mem,d_mem,1);
[num4, den4]=ss2tf(a_mem,b_mem,c_mem,d_mem,2);
Ts = 20;

```

```
tfCPU2CPU = tf(num1,den1,Ts);
tfisyuk2CPU = tf(num2,den2,Ts);
tfmem2mem = tf(num3,den3,Ts);
tfisyuk2mem = tf(num4,den4,Ts);
```

%2 gecikmeli CPU tahsis- CPU kullanım SISO sistemi modeli ve LQG Servo kontrolör

```
numx = [0 0 0.5657];
denx = [1 -0.4286 0];
Ts = 20;
sys3 = tf(numx,denx,Ts);
numy = [0 0 -0.1453];
deny = [1 -0.4286 0];
Ts = 20;
sys4 = tf(numy,deny,Ts);
sys5=[sys3 sys4];

CPU=ss(sys5);
a_CPU=CPU.a;
b_CPU=CPU.b;
c_CPU=CPU.c;
d_CPU=CPU.d;
sys5=ss(a_CPU,b_CPU,c_CPU,d_CPU,20)
```

```
Qn=[0.25];
Rn=0.5*5;
Nn=0;
[kest,L,P] = kalman(sys5,Qn,Rn,Nn);
kest
k_a=kest.a
k_b=kest.b
k_c=kest.c
k_d=kest.d
L
P
Q=[0.0001 0 0;0 0.0001 0; 0 0 0.0001];
R=[0.99];
N=0;
[K,S,e] = lqi(ss(sys3),Q,R,N);
K
S
e
```

% Kontrolör girişi e hata işareti

```
C= lqgtrack(kest,K,'1dof');
sys6=ss(C.a,C.b,C.c,C.d,20);
cont =tf(sys6);
cont
```

% Kontrolör girişi r ve y işareti

```

C2= lqgtrack(kest,K);
sys7=ss(C2.a,C2.b,C2.c,C2.d,20);
cont2 =tf(sys7);
cont2

```

%2 gecikmeli bellek %tahsis- bellek kullanım SISO sistem modeli ve LQG Servo kontrolör

```

numxx = [0 0 0.0207];
denxx = [1 -0.9722 0];
Ts = 20;
sys13 = tf(numxx,denxx,Ts);

```

```

numyy = [0 0 0.0005];
denyy = [1 -0.9722 0];
Ts = 20;
sys14 = tf(numyy,denyy,Ts);
sys15=[sys13 sys14];

```

```

mem=ss(sys15);
a_mem=mem.a;
b_mem=mem.b;
c_mem=mem.c;
d_mem=mem.d;
sys15=ss(a_mem,b_mem,c_mem,d_mem,20)

```

```

%Qn2=[0.25];
%Rn2=0.25;
%Nn2=0;
Qn2=[0.25];
Rn2=[0.25];
Nn2=[0.25];
[kest2,L2,P2] = kalman(sys15,Qn2,Rn2,Nn2);
kest2
k_a2=kest2.a
k_b2=kest2.b
k_c2=kest2.c
k_d2=kest2.d
L2
P2
%Q2=[0.000001 0 0;0 0.000001 0; 0 0 0.000001];
%R2=[1.9];
%N2=0;
Q2=[0.000001 0 0;0 0.000001 0; 0 0 0.000001];
R2=[2.5];
N2=0
[K2,S2,e2] = lqi(ss(sys13),Q2,R2,N2);
K2
S2
e2

```

% Kontrolör girişi e hata işareti

```
C= lqgtrack(kest2,K2,'1dof');
sys16=ss(C.a,C.b,C.c,C.d,20);
cont3 =tf(sys16);
cont3
```

```
% Kontrolör girişi r ve y işareti
```

```
C2= lqgtrack(kest2,K2);
sys17=ss(C2.a,C2.b,C2.c,C2.d,20);
cont4 =tf(sys17);
cont4
```

```
%LQG servo kontrolör emulatör ve simülator sonuçlarının karşılaştırılması
```

```
%-----
```

```
y=x02062013processed_tab2;
plot(y(730:1448,2));
hold all;
plot(y(730:1448,3));
y1=y(730:1448,2);
yuk=y(730:1448,1);
x=20:20:14380;
mCPU=[x ; y1(:,1)'];
save('CPU_reference.mat', 'mCPU', '-v7.3');
myuk =[x ; yuk(:,1)'];
save('load_reference.mat', 'myuk', '-v7.3');
ysimulasyon=Y_CPU.data;
ysimulasyonu=U_CPU.data;
load_CPU=LOAD_CPU.data;
```

```
%-----
```

```
plot(x,y1(:,1));
hold all;
plot(x,y(730:1448,3));
hold all;
plot(x, ysimulasyon);
hold all;
```

```
%-----
```

```
plot(x,y1(:,1));
hold all;
plot(x, ysimulasyonu);
hold all;
plot(x,y(730:1448,4));
```

```
%-----
```

```
plot(x,load_CPU);
```

```
%-----
```

```
plot(x,y(730:1448,2));
hold all;
plot(x,y(730:1448,3));
```

```
%-----
```

```
plot(x,y(730:1448,2));
hold all;
plot(x,y(730:1448,4));
```

```

%-----
plot(x,y(730:1448,2));
hold all;
plot(x,y(730:1448,3));
hold all;
plot(x,y(730:1448,4));
%-----
plot(x,y(730:1448,2));
hold all;
plot(x,y(730:1448,2)-y(730:1448,3));
hold all;
plot(x,y(730:1448,3));
hold all;
plot(x,y(730:1448,4));
%-----
plot(x,y(730:1448,2));
hold all
plot(x, ysimulasyon);
%-----
plot(x,y(730:1448,2));
hold all
plot(x, ysimulasyonu);
%-----
plot(x,y(730:1448,2));
hold all
plot(x, ysimulasyon);
hold all
plot(x, ysimulasyonu);
%-----
xx=0:20:14360
plot(xx,Y_CPU.data);
plot(xx,U_CPU.data);

%-----LQG servo kontrolör ile yönetilen kapalı çevrim sistem transfer
%fonksiyonu, kutuplarının yerleri,basamak cevabı ve köklerini yer egrisi
T=feedback(cont,sys3);
Kokler = roots([1 -1.757 0.9141 -0.06024 -0.03743 0]);
[y,t]=step(feedback(cont,sys3),300)
plot(t,y)
Openloop=cont*sys3;
rlocus(Openloop);
%-----%

```

EK C3 : C# programı

```

using System;
using System.IO;
using System.Windows.Forms;
using VMware.Vim;
using System.Collections.Specialized;

```

```
using System.Timers;
```

```
namespace Wmware
```

```
{
```

```
    public partial class Form1 : Form
```

```
    {
```

```
        public Form1()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        // private static int Url_type          = 2;          // 1 : ESX Server , 2 : vCenter  
Server
```

```
        // private static int Virtual_machine_type = 2;          // 1 : WINXP      , 2 :  
WIN7
```

```
        private static int Controller_type    = 1;          // 1 : PI          , 2 : LQG + I
```

```
        private static int Memory_resource_type = 0;          // 0 : NOT EXISTS , 1 :  
EXISTS
```

```
        private static int CPU_resource_type  = 1;          // 0 : NOT EXISTS , 1 :  
EXISTS
```

```
        private static      System.Timers.Timer aTimer;
```

```
        public static double REF_CPU    = 1000;
```

```
        public static double REF_CPU_T_1 = 0;
```

```
        public static double REF_CPU_T_2 = 0;
```

```
        public static double Y_CPU_T    = 1000;
```

```
        public static double Y_CPU_T_1 = Y_CPU_T;
```

```
        public static double Y_CPU_T_2 = Y_CPU_T;
```

```
        public static double U_CPU_T    = Y_CPU_T;
```

```
        public static double U_CPU_T_1 = Y_CPU_T;
```

```
        public static double U_CPU_T_2 = Y_CPU_T;
```

```
        public static double U_CPU_T_3 = Y_CPU_T;
```

```
        public static double REF_MEM    = 1000;
```

```
        public static double REF_MEM_T_1 = 0;
```

```
        public static double REF_MEM_T_2 = 0;
```

```
        public static double Y_MEM_T    = 1000;
```

```
        public static double Y_MEM_T_1 = Y_MEM_T;
```

```
        public static double Y_MEM_T_2 = Y_MEM_T;
```

```
        public static double U_MEM_T    = Y_MEM_T;
```

```
        public static double U_MEM_T_1 = U_MEM_T;
```

```

public static double U_MEM_T_2 = U_MEM_T;
public static double U_MEM_T_3 = U_MEM_T;

public static double E_CPU_T;
public static double E_CPU_T_1;
public static double E_CPU_T_2;

public static double E_MEM_T;
public static double E_MEM_T_1;
public static double E_MEM_T_2;

public static Int32 sayacymem = 0;
public static Int32 sayacycpu = 0;
public static Int32 sayacmem = 0;
public static Int32 sayaccpu = 0;

public static long T = 20;
public static Int32 sayacxx = 0;
public static Int32 sayacyy = 0;
private void button1_Click(object sender, EventArgs e)
{

    aTimer = new System.Timers.Timer(20000);
    aTimer.Elapsed += new ElapsedEventHandler(OnTimedEvent);
    aTimer.Enabled = true;
    aTimer.Start();
    Console.ReadLine();
}

private static void OnTimedEvent(object source, ElapsedEventArgs e)
{
    string[] args = new string[5];

    /*if (Url_type == 1)
    {
        args[0] = "https://192.168.10.10/sdk/vimService"; //url
        args[1] = "root"; //user
        args[2] = "esxserver"; //password
    }
    else if (Url_type == 2)
    {*/
        args[0] = "https://192.168.10.13/sdk/vimService"; //url
        args[1] = "hp"; //user
        args[2] = "pass"; //password
    /* }*/

    /* if (Virtual_machine_type == 1)
    {
        args[3] = "WinXP"; //vm name
    }*/

```

```

/* else if (Virtual_machine_type == 2)
{
    args[3] = "Win7"; //vm name
/* }*/
args[4] = "onn"; //operation

VmPowerOps vmPowerOps = new VmPowerOps(args);
try
{
    vmPowerOps.DoPowerOps();
    Console.WriteLine("The Elapsed event was raised at {0}", e.SignalTime);
}
catch (VmException ex)
{
    Console.WriteLine("A server fault of type {0} with message '{1}' occurred
while performing requested operation.",
        ex.MethodFault.GetType().Name,
        ex.Message);
}
catch (Exception ex)
{
    Console.WriteLine("An exception of type {0} with message '{1}' occurred
while performing requested operation.",
        ex.GetType(),
        ex.Message);
}
}

public class VmPowerOps
{
    private string serviceUrl = null;
    private string userName = null;
    private string password = null;
    private string vmName = null;
    private string operation = null;

    public VmPowerOps(string[] args)
    {
        ParseArguments(args);
    }

    public void DoPowerOps()
    {

```

```

VimClient client = new VimClient();
client.Connect(serviceUrl);
client.Login(userName, password);
NameValueCollection filter = new NameValueCollection();
filter.Add("name", "^" + vmName + "$");
VirtualMachine vm
=(VirtualMachine)client.FindEntityView(typeof(VirtualMachine), null,filter, null);

if (vm != null)
{
    VirtualMachineConfigSpec hh = new VirtualMachineConfigSpec();
    hh.CpuAllocation = new ResourceAllocationInfo();
    hh.MemoryAllocation = new ResourceAllocationInfo();

    switch (operation)
    {
        case "onn":

            if (CPU_resource_type == 1)
            {
                if (Controller_type == 1)
                {
                    E_CPU_T = REF_CPU - Y_CPU_T;
                    Console.WriteLine("E_CPU_T " + vmName + " : " +
E_CPU_T.ToString());
                    E_CPU_T_1 = REF_CPU - Y_CPU_T_1;
                    Console.WriteLine("E_CPU_T_1 " + vmName + " : " +
E_CPU_T_1.ToString());
                    /* if ((sayacyy == 2) || (sayacyy == 3) || (sayacyy == 4))
                    {
                        U_CPU_T_1 = Y_CPU_T;
                    }*/

                    U_CPU_T = U_CPU_T_1 + (0.44 * E_CPU_T) - (0.4286 *
0.44 * E_CPU_T_1);
                    Console.WriteLine("U_CPU_T " + vmName + " : " +
U_CPU_T.ToString());
                    Console.WriteLine("U_CPU_T_1 " + vmName + " : " +
U_CPU_T_1.ToString());
                    Console.WriteLine("REF_CPU " + vmName + " : " +
REF_CPU.ToString());
                }
                else if (Controller_type == 2)
                {
                    E_CPU_T = REF_CPU - Y_CPU_T;
                    Console.WriteLine("E_CPU_T " + vmName + " : " +
E_CPU_T.ToString());
                    E_CPU_T_1 = REF_CPU - Y_CPU_T_1;

```

```

        // Console.WriteLine("E_CPU_T_1 " + vmName + " : " +
E_CPU_T_1.ToString());
        E_CPU_T_2 = REF_CPU - Y_CPU_T_2;
        // Console.WriteLine("E_CPU_T_2 " + vmName + " : " +
E_CPU_T_2.ToString());

        /* if ((sayacyy == 2) || (sayacyy == 3) || (sayacyy == 4))
        {
            U_CPU_T_1 = Y_CPU_T;
            U_CPU_T_2 = Y_CPU_T;
            U_CPU_T_3 = Y_CPU_T;
        }*/

        U_CPU_T = (int)((1.328 * U_CPU_T_1) -
            (0.3446 * U_CPU_T_2) +
            (0.0162 * U_CPU_T_3) +
            (0.0002676 * REF_CPU) +
            (0.1832 * REF_CPU_T_1) -
            (0.07845 * REF_CPU_T_2) -
            (0.0002676 * Y_CPU_T) -
            (0.1832 * Y_CPU_T_1) +
            (0.07845 * Y_CPU_T_2));
        Console.WriteLine("U_CPU_T " + vmName + " : " +
U_CPU_T.ToString());
        Console.WriteLine("U_CPU_T_1 " + vmName + " : " +
U_CPU_T_1.ToString());
        Console.WriteLine("U_CPU_T_2 " + vmName + " : " +
U_CPU_T_2.ToString());
        // Console.WriteLine("U_CPU_T_3 " + vmName + " : " +
U_CPU_T_3.ToString());
        Console.WriteLine("REF_CPU " + vmName + " : " +
REF_CPU.ToString());
        Console.WriteLine("REF_CPU_T_1 " + vmName + " : " +
REF_CPU_T_1.ToString());
        // Console.WriteLine("REF_CPU_T_2 " + vmName + " : " +
REF_CPU_T_2.ToString());
    }

    hh.CpuAllocation.Limit = (int)U_CPU_T;
    hh.CpuAllocation.ExpandableReservation = false;
    hh.CpuAllocation.OverheadLimit = -1;
    hh.CpuAllocation.Reservation = 0;
    //sayacyy = sayacyy + 1;
}

if (Memory_resource_type == 1)
{
    if (Controller_type == 1)
    {
        E_MEM_T = REF_MEM - Y_MEM_T ;
    }
}

```

```

        Console.WriteLine("E_MEM_T " + vmName + " : " +
E_MEM_T.ToString());
        E_MEM_T_1 = REF_MEM - Y_MEM_T_1 ;
        Console.WriteLine("E_MEM_T_1 " + vmName + " : " +
E_MEM_T_1.ToString());

        if ((sayacxx == 2) || (sayacxx == 3) || (sayacxx == 4))
        {
            U_MEM_T_1 = Y_MEM_T;
        }
        U_MEM_T = U_MEM_T_1 + (1.4 * E_MEM_T) - (1.4 *
0.9722 * E_MEM_T_1);
        Console.WriteLine("U_MEM_T " + vmName + " : " +
U_MEM_T.ToString());
        Console.WriteLine("U_MEM_T_1 " + vmName + " : " +
U_MEM_T_1.ToString());
        Console.WriteLine("REF_MEM " + vmName + " : " +
REF_MEM.ToString());
    }
    else if (Controller_type == 2)
    {
        E_MEM_T = REF_MEM - Y_MEM_T;
        Console.WriteLine("E_MEM_T " + vmName + " : " +
E_MEM_T.ToString());
        E_MEM_T_1 = REF_MEM_T_1 - Y_MEM_T_1;
        Console.WriteLine("E_MEM_T_1 " + vmName + " : " +
E_MEM_T_1.ToString());
        E_MEM_T_2 = REF_MEM_T_2 - Y_MEM_T_2;
        Console.WriteLine("E_MEM_T_2 " + vmName + " : " +
E_MEM_T_2.ToString());
        if ((sayacxx == 2) || (sayacxx == 3) || (sayacxx == 4))
        {
            U_MEM_T_1 = Y_MEM_T;
            U_MEM_T_2 = Y_MEM_T;
            U_MEM_T_3 = Y_MEM_T;
        }
        U_MEM_T = (int)((((1.956 * U_MEM_T_1)
- (0.9564 * U_MEM_T_2)
+ (0.0005807 * U_MEM_T_3)
+ (0.02805 * REF_MEM_T_1)
- (0.02727 * REF_MEM_T_2)
- (0.02805 * Y_MEM_T_1)
+ (0.02727 * Y_MEM_T_2)))));

        /* U_MEM_T = ((1.956 * U_MEM_T_1)
- (0.9564 * U_MEM_T_1)
+ (0.0005807 * U_MEM_T_2));
        Console.WriteLine("U_MEM_T " +
vmName + " : " + U_MEM_T.ToString());

```

```

REF_MEM_T_1)
    reffark = 8*( (0.02805 *
- (0.02727 * REF_MEM_T_2)
- (0.02805 * Y_MEM_T_1)
+ (0.02727 * Y_MEM_T_2));
U_MEM_T= (int) U_MEM_T
+reffark;
    Console.WriteLine("reffark " + vmName
+ " : " + reffark.ToString());*/
    Console.WriteLine("U_MEM_T " + vmName + " : " +
U_MEM_T.ToString());
    Console.WriteLine("U_MEM_T_1 " + vmName + " : " +
U_MEM_T_1.ToString());
    Console.WriteLine("U_MEM_T_2 " + vmName + " : " +
U_MEM_T_2.ToString());
    Console.WriteLine("U_MEM_T_3 " + vmName + " : " +
U_MEM_T_3.ToString());
    Console.WriteLine("REF_MEM " + vmName + " : " +
REF_MEM.ToString());
    Console.WriteLine("REF_MEM_T_1 " + vmName + " : " +
REF_MEM_T_1.ToString());
    Console.WriteLine("REF_MEM_T_2 " + vmName + " : " +
REF_MEM_T_2.ToString());
    }
    hh.MemoryAllocation.Limit = (int)U_MEM_T;
    hh.MemoryAllocation.ExpandableReservation = false;
    hh.MemoryAllocation.OverheadLimit = -1;
    hh.MemoryAllocation.Reservation = 0;
    sayacxx = sayacxx + 1;
    }
    vm.ReconfigVM_Task(hh);
    U_CPU_T_3 = U_CPU_T_2;
    U_CPU_T_2 = U_CPU_T_1;
    U_CPU_T_1 = U_CPU_T;
    REF_CPU_T_2 = REF_CPU_T_1;
    REF_CPU_T_1 = REF_CPU;
    U_MEM_T_3 = U_MEM_T_2;
    U_MEM_T_2 = U_MEM_T_1;
    U_MEM_T_1 = U_MEM_T;
    REF_MEM_T_2 = REF_MEM_T_1;
    REF_MEM_T_1 = REF_MEM;
//*****
*****
    PerformanceManager PERF_MAN =
(PerformanceManager)client.GetView(client.ServiceContent.PerfManager, null);
    ManagedObjectReference budane = vm.MoRef;
    DateTime end = DateTime.Now.AddSeconds(0);
    DateTime start = end.AddSeconds(-80);

```

```

        PerfMetricId[] pm =
PERF_MAN.QueryAvailablePerfMetric(budane, start, end, 20);
        // Console.WriteLine("performance metric sayısı: pm.Length: " +
pm.Length);
        PerfQuerySpec pp = new PerfQuerySpec();
        pp.MetricId = pm;
        pp.Entity = budane;
        pp.IntervalId = 20;
        pp.MaxSample = 4;
        pp.StartTime = start;
        pp.EndTime = end;
        // Console.WriteLine("start : " + start);
        // Console.WriteLine("end : " + end);
        PerfQuerySpec[] ppararray = new PerfQuerySpec[1];
        ppararray[0] = pp;
        PerfEntityMetricBase[] peb = PERF_MAN.QueryPerf(pparray);
        // Console.WriteLine("peb length" + peb.Length);
        long cpu_usage = new long();
        long mem_usage = new long();
        int flag;
        long nettrans_usage = new long();
        for (int j = 0; j < peb.Length; ++j)
        {
            /* PerfSampleInfo[] infos =
((PerfEntityMetric)peb[j]).SampleInfo;
            Console.WriteLine("infos length" + infos.Length);
            for (int zz = 0; zz < infos.Length; ++zz)
            {
                Console.WriteLine("infos[" + zz + "].Interval" +
infos[zz].Interval);
                Console.WriteLine("infos[" + zz + "].Timestamp" +
infos[zz].Timestamp);
            }*/
            PerfMetricSeries[] vals = ((PerfEntityMetric)peb[j]).Value;
            // Console.WriteLine("vals length" + vals.Length);
            flag = 0;
            for (int vj = 0; vj < vals.Length; ++vj)
            {
                if (vj != 3 && vj != 12 && vj != 40 && vj != 44 && vj != 54
&& vj != 57)
                {
                    continue;
                }

                if (
                    vals[vj].Id.CounterId.Equals(6) ||
                    vals[vj].Id.CounterId.Equals(90) ||
                    vals[vj].Id.CounterId.Equals(116))
                {

```

```

if (vals[vj].GetType().Name.Equals("PerfMetricIntSeries"))
{
    PerfMetricIntSeries val = (PerfMetricIntSeries)vals[vj];
    // Console.WriteLine("val length" + val.Value.Length);
    long[] longs = val.Value;
    // Console.WriteLine("longs length" + longs.Length);

    if (Memory_resource_type == 1)
    {
        if (vals[vj].Id.CounterId.Equals(90))
        {
            mem_usage = longs[longs.Length - 1]/1024;
            Y_MEM_T = mem_usage;
            Console.WriteLine("Y_MEM_T " + vmName + " : "
+ Y_MEM_T.ToString());
            Y_MEM_T_1 = longs[longs.Length - 2]/1024;
            Console.WriteLine("Y_MEM_T_1 " + vmName + "
: " + Y_MEM_T_1.ToString());
            Y_MEM_T_2 = longs[longs.Length - 3]/1024;
            Console.WriteLine("Y_MEM_T_2 " + vmName + "
: " + Y_MEM_T_2.ToString());
        }
    }
    if (CPU_resource_type == 1)
    {
        if (vals[vj].Id.CounterId.Equals(6))
        {
            flag = flag + 1;
            if (flag == 1)
            {
                cpu_usage = longs[longs.Length - 1];
                Y_CPU_T = cpu_usage;
                Console.WriteLine("Y_CPU_T " + vmName + " :
" + Y_CPU_T.ToString());
                Y_CPU_T_1 = longs[longs.Length - 2];
                Console.WriteLine("Y_CPU_T_1 " + vmName +
" : " + Y_CPU_T_1.ToString());
                Y_CPU_T_2 = longs[longs.Length - 3];
                // Console.WriteLine("Y_CPU_T_2 " + vmName
+ " : " + Y_CPU_T_2.ToString());
            }
        }
    }
    if (vals[vj].Id.CounterId.Equals(116))
    {
        nettrans_usage = longs[longs.Length - 1];
    }
}

```

```

    }
}

}

//*****
*****

try
{
    string dosya = @"C:\SISTEM_GIRISLERI\" + vmName +
"CPU_C_1_12062014.txt";
    FileStream fs = new FileStream(dosya, FileMode.Append,
FileAccess.Write);
    StreamWriter sw = new StreamWriter(fs);
    string ls_yaz;
    //ls_yaz = DateTime.Now.ToString() + "\t" +
hh.CpuAllocation.Limit.ToString() + "\t" + hh.MemoryAllocation.Limit.ToString()
+ "\t" + netrecv_usage.ToString() + "\t" + nettrans_usage.ToString() + "\t" +
net_usage.ToString() + "\t" + cpu_usage.ToString() + "\t" + mem_usage.ToString();
    //ls_yaz = DateTime.Now.ToString() + "\t" +
hh.CpuAllocation.Limit.ToString() + "\t" + hh.MemoryAllocation.Limit.ToString()
+ "\t" + nettrans_usage.ToString() + "\t" + cpu_usage.ToString() + "\t" +
mem_usage.ToString();
    //ls_yaz = DateTime.Now.ToString() + "\t" +
cpu_limit.ToString() + "\t" + mem_limit.ToString() + "\t" +
nettrans_usage.ToString() + "\t" + cpu_usage.ToString() + "\t" +
mem_usage.ToString();
    ls_yaz = DateTime.Now.ToString() + "\t" +
nettrans_usage.ToString() + "\t" + REF_CPU.ToString() + "\t" +
Y_CPU_T_1.ToString() + "\t" + E_CPU_T.ToString() + "\t" + U_CPU_T.ToString()
+ "\t" + "<-->" + "\t" + REF_MEM.ToString() + "\t" + Y_MEM_T_1.ToString() +
"\t" + (Y_MEM_T_1 / 1024).ToString() + "\t" + E_MEM_T.ToString() + "\t" +
U_MEM_T.ToString() + "\t" + "|";
    sw.WriteLine(ls_yaz);
    sw.Flush();
    sw.Close();
    fs.Close();
}
catch (Exception ex)
{
    // Handle user code exception
    Console.WriteLine("An exception of type {0} with message '{1}'
occured while performing requested operation.",ex.GetType(), ex.Message);
}

break;
case "off":
    vm.PowerOffVM();
    Console.WriteLine(

```

```

        "Virtual Machine '{0}' under host powered off successfully.",
vm.Name);
        break;
    case "suspend":
        vm.SuspendVM();
        Console.WriteLine(
            "Virtual Machine '{0}' under host suspended successfully.",
vm.Name);
        break;
    case "reset":
        vm.ResetVM();
        Console.WriteLine(
            "Virtual Machine '{0}' under host reset successfully", vm.Name);
        break;
    case "rebootGuest":
        vm.RebootGuest();
        Console.WriteLine(
            "Virtual Machine '{0}' under host reboot successfully.", vm.Name);
        break;
    case "shutdownGuest":
        vm.ShutdownGuest();
        Console.WriteLine(
            "Virtual Machine '{0}' under host shutdown successfully.",
vm.Name);
        break;
    case "standbyGuest":
        vm.StandbyGuest();
        Console.WriteLine(
            "Virtual Machine '{0}' under host put into standby mode.",
vm.Name);
        break;
    default:
        Console.WriteLine("Invalid operation: {0}", operation);
        break;
    }
}
else
{
    Console.WriteLine("Unable to find VirtualMachine named '{0}' in
Inventory", vmName);
}
// logout from the vSphere server
client.Disconnect();
}
/// <summary>
/// define the main entry point for the application
private void ParseArguments(string[] args)
{
    serviceUrl = args[0];
    userName = args[1];
}

```

```

        password = args[2];
        vmName = args[3];
        operation = args[4];
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    sayaccpu = sayaccpu + 1;

    REF_CPU = Convert.ToInt64(textBox1.Text);
    if (sayaccpu==1 )
    {
        REF_CPU_T_1 = REF_CPU;
        REF_CPU_T_2 = REF_CPU;
    }
    Console.WriteLine("YENI REF CPU:" + REF_CPU.ToString());
}

private void textBox2_TextChanged(object sender, EventArgs e)
{
    sayacmem = sayacmem + 1;
    REF_MEM = Convert.ToInt64(textBox2.Text);

    if (sayacmem == 1)
    {
        REF_MEM_T_1 = REF_MEM;
        REF_MEM_T_2 = REF_MEM;
    }
    Console.WriteLine("YENI REF MEM:" + REF_MEM.ToString());
}

private void textBox3_TextChanged(object sender, EventArgs e)
{
    sayacycpu = sayacycpu + 1;
    Y_CPU_T = Convert.ToInt64(textBox2.Text);

    if (sayacycpu == 1)
    {
        Y_CPU_T_1 = Y_CPU_T;
        Y_CPU_T_2 = Y_CPU_T;
        U_CPU_T_1 = Y_CPU_T;
        U_CPU_T_2 = Y_CPU_T;
        U_CPU_T_3 = Y_CPU_T;
    }
    Console.WriteLine("YENI CPU OUTPUT:" + Y_CPU_T.ToString());
}

```

```

private void textBox4_TextChanged(object sender, EventArgs e)
{
    sayacymem = sayacymem + 1;

    Y_MEM_T = Convert.ToInt64(textBox2.Text);
    if (sayacymem == 1)
    {
        Y_MEM_T_1 = Y_MEM_T;
        Y_MEM_T_2 = Y_MEM_T;
    }
    Console.WriteLine("YENI MEMORY OUTPUT:" +
Y_MEM_T.ToString());
}

}
}

```

EK D : Çevrimiçi Ödeme Sistemi İstemci-Sunucu Haberleşmesinde Kullanılan Web servis Metotlarından *Borcsorgulama* Metodu Parametre Ve XML Veri Formatı

Örnek olması için istemci-sunucu arasındaki haberleşmesinde kullanılan Web servis metotlarından *borcsorgulama* metoduna ile ilgili parametre ve XML veri formatı aşağıda verilmiştir.

borcsorgulama sorgu metodu ile sunucuya gönderilen parametreler:

sozlesmeNo(String 8)	:Müşteri sözleşme numarası
bankaKodu(String 2)	:Her bankaya verilen kod
anahtar(String 38)	:Sunucu tarafından verilen anahtar veri

borcsorgulama metodunun yanıt olarak döndüğü parametreler:

sozlesmeNo(String 8)	:Müşteri sözleşme numarası
bankaKodu(String 2)	:Her bankaya verilen kod
ilceKodu(String 2)	:Sözleşme numarasının bağlı olduğu ilçe
kontrolSayisi(String 1)	:Sözleşme numarası kontrol sayısı
adi(String 40)	:Müşteri adı
soyadi(String 20)	:Müşteri soyadı
toplamBorc(String 15)	:Toplam borç
islemKodu(String 3)	:İşlem sonucunu gösteren kod

Ödeme sisteminde *bocrsorgulama* metoduna ilişkin istek ve yanıt XML şeması aşağıda belirtilmiştir.

Borç sorgu istek XML mesaj şeması:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsb="WSBanka">

  <soapenv:Body>

    <wsb:bnkBorcsorgulama

soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```

        <sozlesmeNo xsi:type="xsd:string">?</sozlesmeNo>

        <bankaKodu xsi:type="xsd:string">?</bankaKodu>

        <anahtar xsi:type="xsd:string">?</anahtar>

    </wsb:bnkBorcsorgulama>

</soapenv:Body>

</soapenv:Envelope>

Borç sorgu istek yanıt XML mesaj şeması:

<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <SOAP-ENV:Body>

        <ns1:bnkBorcsorgulamaResponse

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="WSBanka">

            <return si:type="ns2:com_webservis_veri_BnkBorcSorgulama"
xmlns:ns2="http://com.webservis/IWSBanka.xsd">

                <adi xsi:nil="true" xsi:type="xsd:string"/>

                <bankaKodu xsi:nil="true" xsi:type="xsd:string"/>

                <ilceKodu xsi:nil="true" xsi:type="xsd:string"/>

                <islemKodu xsi:type="xsd:string">210</islemKodu>

                <kontrolSayisi xsi:nil="true" xsi:type="xsd:string"/>

                <soyAdi xsi:nil="true" xsi:type="xsd:string"/>

                <sozlesmeNo xsi:nil="true" xsi:type="xsd:string"/>

                <toplamBorc xsi:nil="true" xsi:type="xsd:string"/>

            </return>

        </ns1:bnkBorcsorgulamaResponse>

```

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

EK E : Çevrimiçi Ödeme Sistemi Uyarıcı Ve Kontrolör Yazılımı

Çevrimiçi ödeme sistemi için ORACLE veritabanında PL/SQL dili ile geliştirilen bu yazılım kontrolörün yapması gereken hesaplamaları yapar. Log tablosunda tutulan bilgiden yola çıkarak istatistiksel bilgi hazırlar. Bu bilgi son dakikada yapılan tüm çevrimiçi işlemlerin adedi, başarı yüzdesi ve hata yüzdesini içerir. Kontrolör yazılımı sistem yönetici tarafından belirlenen referans giriş ile istatistiksel bilgileri karşılaştırarak bir kontrol işareti üretir. Üretilen işaret sistemi çevrimiçi çalışma modundan offline çalışma moduna ya da offline çalışma modundan çevrimiçi çalışma moduna geçirir. Kontrol işareti aynı zamanda mail sunucu için de bir giriş işaretidir. Mail sunucu sistem çalışma modundaki değişiklikleri, işlemlerin başarı yüzdesi, belirlenen en son referans giriş bilgilerini mail yoluyla sistem yöneticisine raporlar. Sistem yöneticisi bu sonuçları değerlendirir, gerekli gördüğü takdirde referans giriş parametresini bir arayüz vasıtasıyla değiştirerek geri besleme kontrol sisteminin çalışmasına gerekli müdahalede bulunur. Bu işleri gerçekleştiren yazılım prosedürü aşağıda verilmiştir.

PROCEDURE SP_CONTROLLER_ALERTER IS

```
CURSOR mail_receipt_oku IS  
SELECT e_mail FROM mail WHERE program= 'SUNUCU';
```

```
mail_receipt      mail_receipt_oku%ROWTYPE;  
refcursor         sys_refcursor;  
ls_online_offline param.kod%TYPE;  
li_sayi           NUMBER;  
ls_subject        VARCHAR2(100);  
ls_mesaj          VARCHAR2(500);  
ll_hatakod        NUMBER;  
ls_hataaciklama   VARCHAR2(100);  
ls_islzaman       VARCHAR2(14);  
ls_saat           VARCHAR2(2) ;  
ls_dakika         VARCHAR2(2) ;  
li_toplamadet     NUMBER;  
li_hatali_adet    NUMBER;  
li_basarili_adet  NUMBER;  
ll_hata_yuzdesi   NUMBER;  
-- Özgür Armağan
```

```
BEGIN
```

```
    SELECT kod  
    INTO ls_online_offline  
    FROM param  
    WHERE secimtip = 8888;
```

```
IF ls_online_offline= '1' THEN  
BEGIN
```

```

SELECT x.saat SAAT,
       X.dakika DAKIKA ,
       X.adet TOPLAMISLEMADET ,
       NVL(y.adethatali,0) HATALIADET,
       (NVL(x.adet,0) - NVL(y.adethatali,0) ) BASARILIADET ,
       round(NVL(y.adethatali,0)/x.adet,2)* 100 HATAYUZDESI
       INTO ls_saat, ls_dakika, li_toplamadet, li_hatali_adet , li_basarili_adet, ll_hata_yuzdesi
       FROM
(SELECT substr(islzaman,1,2) saat , substr(islzaman,4,2) dakika ,count(*) adet
FROM onlinelog
WHERE isltarih = trunc(sysdate)
AND kurumkodu =11
AND (to_number(to_char(sysdate,'hh24')) * 3600) + (to_number(to_char(sysdate,'mi')) * 60) +
to_number(to_char(sysdate,'ss')) -
((to_number(substr(islzaman,1,2)) * 3600) + (to_number(substr(islzaman,4,2)) * 60) +
to_number(substr(islzaman,7,2))) <120
AND (to_number(to_char(sysdate,'hh24')) * 3600) + (to_number(to_char(sysdate,'mi')) * 60) +
to_number(to_char(sysdate,'ss')) -
((to_number(substr(islzaman,1,2)) * 3600) + (to_number(substr(islzaman,4,2)) * 60) +
to_number(substr(islzaman,7,2))) >= 60
GROUP BY substr(islzaman,1,2) , substr(islzaman,4,2)) x ,
(SELECT substr(islzaman,1,2) saat, substr(islzaman,4,2) dakika ,count(*) adethatali
FROM onlinelog
WHERE isltarih = trunc(sysdate)
AND kurumkodu =11
AND (to_number(to_char(sysdate,'hh24')) * 3600) + (to_number(to_char(sysdate,'mi')) * 60) +
to_number(to_char(sysdate,'ss')) -
((to_number(substr(islzaman,1,2)) * 3600) + (to_number(substr(islzaman,4,2)) * 60) +
to_number(substr(islzaman,7,2))) <120
AND (to_number(to_char(sysdate,'hh24')) * 3600) + (to_number(to_char(sysdate,'mi')) * 60) +
to_number(to_char(sysdate,'ss')) -
((to_number(substr(islzaman,1,2)) * 3600) + (to_number(substr(islzaman,4,2)) * 60) +
to_number(substr(islzaman,7,2))) > 60
AND gelenxml NOT LIKE '%<islemKodu xsi:type="xsd:string"%'
GROUP BY substr(islzaman,1,2) , substr(islzaman,4,2)) y
WHERE x.saat = y.saat (+)
AND x.dakika = y.dakika (+)
ORDER BY x.saat , x.dakika;
EXCEPTION
WHEN OTHERS THEN
ls_saat := '00';
ls_dakika := '00';
li_toplamadet := 0;
li_hatali_adet := 0;
li_basarili_adet := 0;
ll_hata_yuzdesi := 0;
END;
ls_subject := 'SUNUCU SISTEM HATASI';
ls_mesaj := 'Sunucu ile mevcut olan baglantida problem mevcuttur.';

IF ( li_hatali_adet >=10 AND ll_hata_yuzdesi > 85 ) THEN

UPDATE param
SET kod = 0
WHERE secimtip = 9999;

IF SQL%ROWCOUNT > 0 THEN
ls_mesaj := ls_mesaj || ' Sistem offline calismaya alinmistir.';
COMMIT;

```

```

END IF;

    OPEN mail_receipt_oku ;
    LOOP
        FETCH mail_receipt_oku INTO mail_receipt;
        EXIT WHEN mail_receipt_oku%NOTFOUND;
    SP_MAIL_SEND('CONTROLLER@SIS.com.tr',mail_receipt.e_mail,ls_subject,ls_mesaj,ll_hatakod,l
s_hataaciklama);
        END LOOP;
    CLOSE mail_receipt_oku;
END IF;

ELSE

SELECT to_char(sysdate,'hh24:mi:ssss') INTO ls_islzaman FROM dual;

    SP_ONLINE_SORGU_CONT('2222',refcursor);


SELECT count(*) INTO li_sayi
FROM onlinelog
WHERE kurumkodu = 11
AND isltarih = trunc(sysdate)
AND fonksiyontip = 'bnkBorcsorgulama'
AND islzaman >= ls_islzaman
AND gelenxml LIKE '%<islemKodu xsi:type="xsd:string"%';

ls_subject := 'SISTEM ONLINE CALISMAYA ACILMISTIR';
ls_mesaj := 'Mevcut baglantida olan problem giderilmistir.';

IF li_sayi >= 1 THEN

    UPDATE param
    SET kod = 1
    WHERE secimtip = 9998;

    IF SQL%ROWCOUNT > 0 THEN
        ls_mesaj := ls_mesaj || ' Sistem çevrimiçi çalışmaya alınmıştır.';
        COMMIT;
    END IF;

    OPEN mail_receipt_oku ;
    LOOP
        FETCH mail_receipt_oku INTO mail_receipt;
        EXIT WHEN mail_receipt_oku%NOTFOUND;

    SP_MAIL_SEND('CONTROLLER@SIS.com.tr',mail_receipt.e_mail,ls_subject,ls_mesaj,ll_hatakod,l
s_hataaciklama);
        END LOOP;
    CLOSE mail_receipt_oku;
END IF;
END IF;
END; -- SP_CONTROLLER_ALERTER

```

EK F : Sunucu İle HTTP Bağlantı Kuran, Sorgu, Ödeme, Ödeme İptal İşlemlerin Gerçekleştirilmesini Sağlayan Yazılım

Aşağıdaki yazılım çevrimiçi ödeme sistemi için ORACLE veritabanında PL/SQL dili ile geliştirmiştir ve istemci sunucu arasında HTTP bağlantıyı sağlar. Bu bağlantı üzerinden daha önceden formatı belli olan XML veri, aşağıdaki yazılım vasıtasıyla sunucunun beklediği formatta XML veriye çevrilerek sunucuya gönderilir ve yanıt alınır. XML formatındaki yanıt aşağıdaki yazılım vasıtasıyla ayrıştırılarak (parsing) ilgili tablolarda tutulur ve kullanıcının anlayacağı biçimde kullanıcı ekranına getirilir. Kullanılan yazılımın ufak bir parçası örnek olması açısından aşağıda verilmiştir.

PACKAGE BODY PK_ONLINE

IS

```
-----
FUNCTION ENVELOP
RETURN VARCHAR2 IS
    str VARCHAR2(1000);
BEGIN
    str := '<?xml version="1.0" encoding="UTF-8"?'>';
    str := str||'<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
';
    str := str||'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ';
    str := str||'xmlns:xsd="http://www.w3.org/2001/XMLSchema">';
    RETURN (str);
END;
-----
FUNCTION SOAP_COMMUNICATION (
    ADRES    IN      VARCHAR2,
    EYLEM    IN      VARCHAR2,
    ENVELOP  IN      VARCHAR2)
RETURN     CLOB IS
    RESULT   CLOB;
    http_req UTL_HTTP.req;
    http_resp UTL_HTTP.resp;
    NAME     VARCHAR2(256);
    value    VARCHAR2(1024);
    BUFFER   VARCHAR2(32000);

BEGIN
    UTL_HTTP.set_persistent_conn_support (false);
    UTL_HTTP.set_transfer_timeout(10);
    UTL_HTTP.set_wallet('file:/etc/ORACLE/WALLETS','XXX');
    UTL_HTTP.set_proxy (NULL);
    http_req := UTL_HTTP.begin_request (ADRES, 'POST', 'HTTP/1.1');
    UTL_HTTP.set_header (http_req, 'Content-Type', 'text/xml; charset=utf-8');
    UTL_HTTP.set_header (http_req, 'Content-Length', LENGTH(ENVELOP));
    UTL_HTTP.set_header (http_req, 'SOAPAction', EYLEM);
    UTL_HTTP.write_text (http_req, ENVELOP);
    http_resp := UTL_HTTP.get_response (http_req);
    RESULT := ";
```

```

        LOOP
            UTL_HTTP.read_text (http_resp,BUFFER);
            RESULT := RESULT || BUFFER;
        END LOOP;
        UTL_HTTP.end_response (http_resp);
        RETURN(RESULT);
    EXCEPTION

        WHEN UTL_HTTP.end_of_body THEN
--      LOG(EYLEM,ENVELOP,'HATA END OF
BODY:'||to_char(UTL_HTTP.get_detailed_sqlcode)||':'||UTL_HTTP.get_detailed_sqlerrm);
            UTL_HTTP.end_response (http_resp);
            RETURN(RESULT);
        WHEN OTHERS THEN
            LOG(EYLEM,ENVELOP,'HATA SOAP
COMMUNICATION:'||to_char(UTL_HTTP.get_detailed_sqlcode)||':'||UTL_HTTP.get_detailed_sqlerrm);
            UTL_HTTP.end_response (http_resp);
            RETURN(RESULT);
        END;

-----
FUNCTION SOAP_COMMUNICATION_LONG (
    ADRES    IN        VARCHAR2,
    EYLEM    IN        VARCHAR2,
    ENVELOP  IN        VARCHAR2)
RETURN      CLOB IS
RESULT      CLOB;
http_req    UTL_HTTP.req;
http_resp   UTL_HTTP.resp;
NAME        VARCHAR2(256);
value       VARCHAR2(1024);
BUFFER      VARCHAR2(32000);

BEGIN
    UTL_HTTP.set_persistent_conn_support (true);
    UTL_HTTP.set_transfer_timeout(120);
    UTL_HTTP.set_proxy (NULL);
    http_req := UTL_HTTP.begin_request (ADRES, 'POST', 'HTTP/1.1');
    UTL_HTTP.set_header (http_req, 'Content-Type', 'text/xml; charset=utf-8');
    UTL_HTTP.set_header (http_req, 'Content-Length', LENGTH(ENVELOP));
    UTL_HTTP.set_header (http_req, 'SOAPAction', EYLEM);
    UTL_HTTP.write_text (http_req, ENVELOP);

    http_resp := UTL_HTTP.get_response (http_req);

    RESULT := "";
    LOOP
        UTL_HTTP.read_text (http_resp,BUFFER);
        RESULT := RESULT || BUFFER;
    END LOOP;
    UTL_HTTP.end_response (http_resp);
    RETURN(RESULT);
EXCEPTION
    WHEN UTL_HTTP.end_of_body THEN
--      LOG(EYLEM,ENVELOP,'HATA END OF
BODY:'||to_char(UTL_HTTP.get_detailed_sqlcode)||':'||UTL_HTTP.get_detailed_sqlerrm);
        UTL_HTTP.end_response (http_resp);
        RETURN(RESULT);

```

```

        WHEN OTHERS THEN
            LOG(EYLEM,ENVELOP,'HATA SOAP_LONG
COMMUNICATION:'||to_char(UTL_HTTP.get_detailed_sqlcode)||':'||UTL_HTTP.get_detailed_sqlerrm);
            UTL_HTTP.end_response (http_resp);
            RETURN(RESULT);

```

```

END;

```

```

-----
PROCEDURE LOG(p_fonksiyontip IN varchar2,
              p_gidenmesaj IN varchar2,
              p_gelenmesaj IN varchar2)

```

```

AS

```

```

    PRAGMA AUTONOMOUS_TRANSACTION;

```

```

BEGIN

```

```

    INSERT INTO onlinelog(isltarih,islzaman,fonksiyontip,gidenmesaj,gelenxml,kurumkodu)

```

```

VALUES(trunc(sysdate),TO_CHAR(sysdate,'hh24:mi:sssss'),p_fonksiyontip,substr(p_gidenmesaj,1,4000),substr(p_gelenmesaj,1,4000),11);

```

```

    COMMIT;
END;

```

```

-----
-- FOR i IN 1..utl_http.get_header_count(http_resp) LOOP
--   utl_http.get_header(http_resp, i, NAME, value);
--   dbms_output.put_line(NAME || ': ' || value);
-- END LOOP;
--
dbms_output.put_line('utl_http.get_detailed_sqlerrm...'||SUBSTR(utl_http.get_detailed_sqlerrm,1,100));
--
dbms_output.put_line('utl_http.get_detailed_sqlcode...'||TO_CHAR(utl_http.get_detailed_sqlcode));
-----

```

```

FUNCTION FONKSIYON_SEC(li_kod IN NUMBER)
RETURN VARCHAR2 IS
EYLEM VARCHAR2(30);
BEGIN
    SELECT isim
    INTO EYLEM
    FROM online_fonksiyonlar
    WHERE numara = li_kod;
    return EYLEM;
EXCEPTION
WHEN OTHERS THEN
    EYLEM := '';
    return EYLEM;
END;

```

```

-----
FUNCTION ENVELOP_BODY
( parametre IN online_parametre_type,
  fonksiyon_tip IN NUMBER)
RETURN VARCHAR2 IS

```

```

CURSOR online IS

```

```

SELECT numara,sira,isim,tip,durum
  FROM online_alanlar
 WHERE numara = fonksiyon_tip
 ORDER BY sira;

online_rec  online_alanlar%ROWTYPE;
online_rec2 online_fonksiyonlar%ROWTYPE;
str         VARCHAR2(4000) := '';
i           NUMBER := 0;

BEGIN
  BEGIN
    SELECT isim , buffer
      INTO online_rec2.isim , online_rec2.buffer
      FROM online_fonksiyonlar
     WHERE numara = fonksiyon_tip
           AND durum = 1;
  EXCEPTION
    WHEN others THEN
      RETURN '';
  END;

  str := '<SOAP-ENV:Body>';
  str := str||'<ns1:'||online_rec2.isim||' '||online_rec2.buffer||'>';
  OPEN online;
  LOOP
    FETCH online INTO online_rec;
    EXIT WHEN online%NOTFOUND;
    i := i + 1;
    str:= str ||'<'||online_rec.isim||'
xsi:type=||'xsd:'||online_rec.tip||'>'||parametre(i)||'<'||online_rec.isim||'>';
  END LOOP;
  CLOSE online;
  str := str||'</ns1:'||online_rec2.isim||'>';
  str := str||'</SOAP-ENV:Body>';
  str := str||'</SOAP-ENV:Envelope>';
  RETURN(str);
END;

-----
FUNCTION PARAMETREOLUSTUR(STR IN VARCHAR2)
RETURN online_parametre_type IS

online_parametre online_parametre_type;
position_start NUMBER;
i               NUMBER;
BEGIN
  position_start := 1;
  FOR i IN 1..LENGTH(STR) LOOP
    IF SUBSTR(STR,i,1) = '|' THEN
      online_parametre(TO_NUMBER(SUBSTR(STR,position_start,1))) :=
SUBSTR(STR,position_start+1,i-1-position_start);
      position_start := i+1;
    END IF;
  END LOOP;
  RETURN(online_parametre);
END;

-----

FUNCTION XML_OLUSTUR(STR IN VARCHAR2,fonksiyontip IN NUMBER)

```

```

RETURN VARCHAR2 IS
online_parametre online_parametre_type;
ls_string VARCHAR2(1000);

BEGIN

online_parametre := PARAMETREOLUSTUR(STR);
ls_string := ENVELOP();
ls_string := ls_string||ENVELOP_BODY(online_parametre,fonksiyontip);

RETURN(ls_string);

END;

-----

PROCEDURE ANAHTAR_BUL(p_bankakodu OUT VARCHAR2,p_anahtar OUT VARCHAR2)
AS

BEGIN

SELECT kod,kodaciklama
INTO p_bankakodu,p_anahtar
FROM param
WHERE secimtip = 9900;

EXCEPTION
WHEN OTHERS THEN
p_bankakodu := '';
p_anahtar := '';
END;

-----

PROCEDURE WEBSERVISBUL
AS

BEGIN
SELECT kodaciklama
INTO WEBSERVISADRES
FROM param
WHERE secimtip = 8888;

EXCEPTION
WHEN OTHERS THEN
WEBSERVISADRES := '';
END;

-----

PROCEDURE BORC_SORGULAMA2(XMLDOC IN VARCHAR2,
pRetCur IN OUT SYS_REFCURSOR)

AS
myParser xmlparser.Parser;
doc xmldom.DOMDocument;
n_dummy xmldom.DOMNode;
n_node xmldom.DOMNode;
nodelist xmldom.DOMNodeList;
len NUMBER;
adi VARCHAR2(100):= '';
bankakodu VARCHAR2(20):= '';
ilcekodu VARCHAR2(20):= '';
islemkodu VARCHAR2(20):= '';

```

```

kontrolsayisi    VARCHAR2(20):= '';
soyadi           VARCHAR2(100):= '';
sozlesmeno       VARCHAR2(20):= '';
toplamborc       VARCHAR2(20):= '';
FLAG             NUMBER(1):=0;
islemaciklama    VARCHAR2(100):= '';
sqlcodum         VARCHAR2(100);
ls_durum         CHAR(1);
ls_str           VARCHAR2(410);

```

```
BEGIN
```

```

adi              := '';
bankakodu        := '0';
ilcekodu         := '';
islemkodu        := '0';
kontrolsayisi    := '0';
soyadi           := '';
sozlesmeno       := '';
toplamborc       := '0';
islemaciklama    := '';

```

```
DELETE FROM onlinetemp;
```

```
PK_ONLINE.SP_ONLINE_CALISMA(2,ls_durum ,ls_str);
```

```
if ls_durum = '1' then
```

```

myParser := xmlparser.newParser;
xmlparser.setValidationMode( myParser, FALSE);
xmlparser.parseBuffer(myParser,XMLDOC);
doc       := xmlparser.getDocument(myParser);
nodelist := xmldom.getElementsByTagName(doc, '*');
len       := xmldom.getLength(nodelist);
FOR i IN 0..len-1 LOOP

```

```

    n_node := xmldom.item(nodelist,i);
    n_dummy := xmldom.getFirstChild(n_node);

```

```

IF NOT xmldom.isNull(n_dummy) THEN
    IF (upper(xmldom.getNodeName(n_node)) = 'RETURN') THEN
        FLAG := 1;
    END IF;

```

```

    IF (upper(xmldom.getNodeName(n_node)) = 'ADI') THEN
        adi := NVL(xmldom.getNodeValue(n_dummy),'');
--        dbms_output.put_line(adi);
    END IF;

```

```

    IF (upper(xmldom.getNodeName(n_node)) = 'BANKAKODU') THEN
        bankakodu := NVL(xmldom.getNodeValue(n_dummy),'0');
--        dbms_output.put_line(bankakodu);
    END IF;

```

```

    IF (upper(xmldom.getNodeName(n_node)) = 'ILCEKODU') THEN
        ilcekodu := NVL(xmldom.getNodeValue(n_dummy),'');
--        dbms_output.put_line(ilcekodu);
    END IF;

```

```
IF (upper(xmldom.getNodeName(n_node)) = 'ISLEMKODU') THEN
```

```

        islemkodu := NVL(xmlDOM.getNodeValue(n_dummy),'0');
--      dbms_output.put_line(islemkodu);
    END IF;

    IF (upper(xmlDOM.getNodeName(n_node)) = 'KONTROLSAYISI') THEN
        kontrolsayisi:= NVL(xmlDOM.getNodeValue(n_dummy),'0');
--      dbms_output.put_line(kontrolsayisi);
    END IF;

    IF (upper(xmlDOM.getNodeName(n_node)) = 'SOYADI') THEN
        soyadi:= NVL(xmlDOM.getNodeValue(n_dummy),' ');
--      dbms_output.put_line(soyadi);
    END IF;

    IF (upper(xmlDOM.getNodeName(n_node)) = 'SOZLESMENO') THEN
        sozlesmeno := NVL(xmlDOM.getNodeValue(n_dummy),' ');
--      dbms_output.put_line(sozlesmeno);
    END IF;

    IF (upper(xmlDOM.getNodeName(n_node)) = 'TOPLAMBORC') THEN
        toplamborc := NVL(xmlDOM.getNodeValue(n_dummy),'0');
--      dbms_output.put_line(toplamborc);
    END IF;
END IF;
END LOOP;
xmlDOM.freeDocument(doc);
xmlparser.freeParser(myParser);

BEGIN
SELECT aciklama
  INTO islemaciklama
  FROM hata
 WHERE kod = 10000 + TO_NUMBER(ISLEMKODU);
EXCEPTION
WHEN OTHERS THEN
  islemaciklama := ' ';
END;

IF FLAG = 1 THEN

  BEGIN

    INSERT INTO onlinetemp
    ( islemkodu , islemaciklamasi , adi,
      soyadi , bankakodu , sozlesmeno ,
      toplamborc , ilcekodu , kontrolsayisi)
    VALUES(NVL(islemkodu,' '),NVL(islemaciklama,' '),NVL(adi,' '),
      NVL(soyadi,' '),NVL(bankakodu,' '),NVL(sozlesmeno,'
'),TO_NUMBER(REPLACE(toplamborc,',','')),
      NVL(ilcekodu,' '),TO_NUMBER(kontrolsayisi));
    EXCEPTION
    WHEN OTHERS THEN
      sqlcodum := to_char(sqlcode);
    INSERT INTO onlinetemp
    ( islemkodu , islemaciklamasi , adi,
      soyadi , bankakodu , sozlesmeno ,
      toplamborc , ilcekodu , kontrolsayisi)
    VALUES('999',sqlcodum,' ',' ',' ',0,'0');
  END;

```

```

END IF;
elsif ls_durum = '0' then
  INSERT INTO onlinetemp
  ( islemkodu , islemaciklamasi , adi,
    soyadi , bankakodu , sozlesmeno ,
    toplamborc , ilcekodu , kontrolsayisi)
  VALUES(NVL('501',' '),NVL('Sistem ONLINE çalışmaya kapatıldı.İşlem yapamazsınız.','
'),NVL(' ',' '),
    NVL(' ',' '),NVL(' ',' '),NVL(' ',' '),TO_NUMBER(REPLACE('0',' ','')),
    NVL(' ',' '),TO_NUMBER('0'));
end if;
OPEN pRetCur FOR SELECT * FROM onlinetemp ;
EXCEPTION
WHEN OTHERS THEN
  xmldom.freeDocument(doc);
  xmlparser.freeParser(myParser);
END;

```

```

-----
PROCEDURE _BORC_SORGULAMA(XMLDOC IN VARCHAR2, p_str OUT NOCOPY
VARCHAR2)

```

```

AS
myParser      xmlparser.Parser;
doc           xmldom.DOMDocument;
n_dummy      xmldom.DOMNode;
n_node       xmldom.DOMNode;
nodelist     xmldom.DOMNodelist;
len          NUMBER;
adi          VARCHAR2(100):= ' ';
bankakodu    VARCHAR2(20):= ' ';
ilcekodu     VARCHAR2(20):= ' ';
islemkodu    VARCHAR2(20):= ' ';
kontrolsayisi VARCHAR2(20):= ' ';
soyadi      VARCHAR2(100):= ' ';
sozlesmeno   VARCHAR2(20):= ' ';
toplamborc   VARCHAR2(20):= ' ';
FLAG        NUMBER(1):=0;
ls_aciklama  hata.aciklama%type:= ' ';
BEGIN

```

```

  adi      := ' ';
  bankakodu := '0';
  ilcekodu := ' ';
  islemkodu := '0';
  kontrolsayisi := '0';
  soyadi   := ' ';
  sozlesmeno := ' ';
  toplamborc := '0';

```

```

  myParser := xmlparser.newParser;
  xmlparser.setValidationMode( myParser, FALSE);
  xmlparser.parseBuffer(myParser,XMLDOC);
  doc := xmlparser.getDocument(myParser);
  nodelist := xmldom.getElementsByTagName(doc, '*');
  len := xmldom.getLength(nodelist);
  FOR i IN 0..len-1 LOOP
    n_node := xmldom.item(nodelist, i);
    n_dummy := xmldom.getFirstChild(n_node);

```

```

    IF NOT xmldom.isNull(n_dummy) THEN
      IF (upper(xmldom.getNodeName(n_node)) = 'RETURN') THEN

```

```

        FLAG := 1;
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'ADI') THEN
        adi := NVL(xmldom.getNodeValue(n_dummy),' ');
        dbms_output.put_line(adi);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'BANKAKODU') THEN
        bankakodu := NVL(xmldom.getNodeValue(n_dummy),'0');
        dbms_output.put_line(bankakodu);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'ILCEKODU') THEN
        ilcekodu := NVL(xmldom.getNodeValue(n_dummy),' ');
        dbms_output.put_line(ilcekodu);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'ISLEMKODU') THEN
        islemkodu := NVL(xmldom.getNodeValue(n_dummy),'0');
        dbms_output.put_line(islemkodu);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'KONTROLSAYISI') THEN
        kontrolsayisi := NVL(xmldom.getNodeValue(n_dummy),'0');
        dbms_output.put_line(kontrolsayisi);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'SOYADI') THEN
        soyadi := NVL(xmldom.getNodeValue(n_dummy),' ');
        dbms_output.put_line(soyadi);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'SOZLESMENO') THEN
        sozlesmeno := NVL(xmldom.getNodeValue(n_dummy),' ');
        dbms_output.put_line(sozlesmeno);
    END IF;

    IF (upper(xmldom.getNodeName(n_node)) = 'TOPLAMBORC') THEN
        toplamborc := NVL(xmldom.getNodeValue(n_dummy),'0');
        dbms_output.put_line(toplamborc);
    END IF;
END IF;
END LOOP;
xmldom.freeDocument(doc);
xmlparser.freeParser(myParser);

BEGIN
    SELECT aciklama
    INTO ls_aciklama
    FROM hata
    WHERE kod = 99000 + TO_NUMBER(ISLEMKODU);
EXCEPTION
    WHEN OTHERS THEN
        ls_aciklama := 'Dönen Koda Ait Açıklama Bulunamadı.';
END;

IF FLAG = 1 AND ISLEMKODU = '401' THEN

```

```

        p_str := 'OKEY' || LPAD(ISLEMKODU,3,'0') || LPAD(SUBSTR(NVL(adi,''),1,100),100,' ')
    ||
        LPAD(NVL(bankakodu,''),20,' ') || LPAD(NVL(ilcekodu,''),20,' ') ||
        LPAD(NVL(kontrolsayisi,''),20,' ') || LPAD(SUBSTR(NVL(soyadi,''),1,20),100,' ') ||
        LPAD(NVL(sozlesmeno,''),20,' ') || LPAD(NVL(toplamborc,''),20,' ') ||
        LPAD(NVL(ls_aciklama,''),100,' ') ;

ELSE

        p_str := 'HATA' || LPAD(ISLEMKODU,3,'0') || LPAD(SUBSTR(NVL(adi,''),1,100),100,' ')
    ||
        LPAD(NVL(bankakodu,''),20,' ') || LPAD(NVL(ilcekodu,''),20,' ') ||
        LPAD(NVL(kontrolsayisi,''),20,' ') || LPAD(SUBSTR(NVL(soyadi,''),1,20),100,' ') ||
        LPAD(NVL(sozlesmeno,''),20,' ') || LPAD(NVL(toplamborc,''),20,' ') ||
        LPAD(NVL(ls_aciklama,''),100,' ') ;

END IF;
EXCEPTION
WHEN OTHERS THEN
    xmldom.freeDocument(doc);
    xmlparser.freeParser(myParser);
END;

```

```

-----
PROCEDURE _TAHSILAT_ISLEM(XMLDOC IN VARCHAR2,

```

```

    adi                OUT VARCHAR2,
    bankakodu          OUT VARCHAR2,
    ilcekodu           OUT VARCHAR2,
    islemkodu          OUT VARCHAR2,
    kontrolsayisi      OUT VARCHAR2,
    makbuzno           OUT VARCHAR2,
    referansno         OUT VARCHAR2,
    soyadi             OUT VARCHAR2,
    sozlesmeno         OUT VARCHAR2,
    subekodu           OUT VARCHAR2,
    tarih              OUT VARCHAR2,
    ykr                OUT VARCHAR2,
    ytl                OUT VARCHAR2)

```

```

AS

```

```

    myParser           xmlparser.Parser;
    doc                xmldom.DOMDocument;
    n_dummy            xmldom.DOMNode;
    n_node             xmldom.DOMNode;
    nodelist            xmldom.DOMNodelist;
    len                NUMBER;

```

```

BEGIN

```

```

    adi                := '';
    bankakodu          := '0';
    ilcekodu           := '';
    islemkodu          := '0';
    kontrolsayisi      := '0';
    makbuzno           := '0';
    referansno         := '0';
    soyadi             := '';
    sozlesmeno         := '';
    subekodu           := '0';

```

```

tarih      := ' ';
ykr        := '0';
ytl        := '0';

myParser := xmlparser.newParser;
xmlparser.setValidationMode(myParser, FALSE);
xmlparser.parseBuffer(myParser,XMLDOC);
doc      := xmlparser.getDocument(myParser);
nodelist := xmldom.getElementsByTagName(doc, '*');
len      := xmldom.getLength(nodelist);
FOR i IN 0..len-1 LOOP
    n_node := xmldom.item(nodelist, i);
    n_dummy := xmldom.getFirstChild(n_node);

    IF NOT xmldom.isNull(n_dummy) THEN
        IF (upper(xmldom.getNodeName(n_node)) = 'ADI') THEN
            adi := NVL(xmldom.getNodeValue(n_dummy),' ');
--            dbms_output.put_line(adi);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'BANKAKODU') THEN
            bankakodu := NVL(xmldom.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(bankakodu);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'ILCEKODU') THEN
            ilcekodu := NVL(xmldom.getNodeValue(n_dummy),' ');
--            dbms_output.put_line(ilcekodu);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'ISLEMKODU') THEN
            islemkodu := NVL(xmldom.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(islemkodu);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'KONTROLSAYISI') THEN
            kontrolsayisi := NVL(xmldom.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(kontrolsayisi);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'MAKBUZNO') THEN
            makbuzno := NVL(xmldom.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(makbuzno);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'REFERANSNO') THEN
            referansno := NVL(xmldom.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(referansno);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'SOYADI') THEN
            soyadi := NVL(xmldom.getNodeValue(n_dummy),' ');
--            dbms_output.put_line(soyadi);
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'SOZLESMENO') THEN
            sozlesmeno := NVL(xmldom.getNodeValue(n_dummy),' ');
--            dbms_output.put_line(sozlesmeno);
        END IF;
    END IF;
END LOOP;

```

```

        IF (upper(xmlDOM.getNodeName(n_node)) = 'SUBEKODU') THEN
            subekodu := NVL(xmlDOM.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(subekodu);
        END IF;

        IF (upper(xmlDOM.getNodeName(n_node)) = 'TARİH') THEN
            tarih := NVL(xmlDOM.getNodeValue(n_dummy),'');
--            dbms_output.put_line(tarih);
        END IF;

        IF (upper(xmlDOM.getNodeName(n_node)) = 'YKR') THEN
            ykr := NVL(xmlDOM.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(subekodu);
        END IF;

        IF (upper(xmlDOM.getNodeName(n_node)) = 'YTL') THEN
            ytl := NVL(xmlDOM.getNodeValue(n_dummy),'0');
--            dbms_output.put_line(subekodu);
        END IF;
    END IF;
END LOOP;
xmlDOM.freeDocument(doc);
xmlParser.freeParser(myParser);
EXCEPTION
WHEN OTHERS THEN
    xmlDOM.freeDocument(doc);
    xmlParser.freeParser(myParser);
END;
```

```

-----
PROCEDURE _TAHSILAT_IPTAL_ISLEM(XMLDOC IN VARCHAR2,
```

```

    adi                OUT VARCHAR2,
    bankakodu           OUT VARCHAR2,
    ilcekodu            OUT VARCHAR2,
    islemkodu           OUT VARCHAR2,
    kontrolsayisi       OUT VARCHAR2,
    makbuzno            OUT VARCHAR2,
    referansno          OUT VARCHAR2,
    soyadi              OUT VARCHAR2,
    sozlesmeno          OUT VARCHAR2,
    tarih               OUT VARCHAR2)
AS
```

```

    myParser            xmlParser.Parser;
    doc                 xmlDOM.DOMDocument;
    n_dummy             xmlDOM.DOMNode;
    n_node              xmlDOM.DOMNode;
    nodelist            xmlDOM.DOMNodeList;
    len                 NUMBER;
```

```

BEGIN
```

```

    soyadi              := '';
    adi                := '';
    bankakodu           := '0';
    ilcekodu            := '';
    kontrolsayisi       := '0';
    makbuzno            := '0';
    referansno          := '0';
    sozlesmeno          := '';
    tarih               := '';
    islemkodu           := '0';
```

```

myParser := xmlparser.newParser;
xmlparser.setValidationMode(myParser, FALSE);
xmlparser.parseBuffer(myParser,XMLDOC);
doc := xmlparser.getDocument(myParser);
nodelist := xmldom.getElementsByTagName(doc, '*');
len := xmldom.getLength(nodelist);
FOR i IN 0..len-1 LOOP
    n_node := xmldom.item(nodelist, i);
    n_dummy := xmldom.getFirstChild(n_node);
    IF NOT xmldom.isNull(n_dummy) THEN
        IF (upper(xmldom.getNodeName(n_node)) = 'ADI') THEN
            adi := NVL(xmldom.getNodeValue(n_dummy),'');
            dbms_output.put_line(adi);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'BANKAKODU') THEN
            bankakodu := NVL(xmldom.getNodeValue(n_dummy),'0');
            dbms_output.put_line(bankakodu);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'ILCEKODU') THEN
            ilcekodu := NVL(xmldom.getNodeValue(n_dummy),'');
            dbms_output.put_line(ilcekodu);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'ISLEMKODU') THEN
            islemkodu := NVL(xmldom.getNodeValue(n_dummy),'0');
            dbms_output.put_line(islemkodu);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'KONTROLSAYISI') THEN
            kontrolsayisi:= NVL(xmldom.getNodeValue(n_dummy),'0');
            dbms_output.put_line(kontrolsayisi);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'MAKBUZNO') THEN
            makbuzno:= NVL(xmldom.getNodeValue(n_dummy),'0');
            dbms_output.put_line(makbuzno);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'REFERANSNO') THEN
            referansno := NVL(xmldom.getNodeValue(n_dummy),'0');
            dbms_output.put_line(referansno);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'SOYADI') THEN
            soyadi := NVL(xmldom.getNodeValue(n_dummy),' ');
            dbms_output.put_line(soyadi);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'SOZLESMENO') THEN
            sozlesmeno := NVL(xmldom.getNodeValue(n_dummy),' ');
            dbms_output.put_line(sozlesmeno);
        --
        END IF;

        IF (upper(xmldom.getNodeName(n_node)) = 'TARİH') THEN
            tarih := NVL(xmldom.getNodeValue(n_dummy),' ');
            dbms_output.put_line(tarih);
        --
    
```

```

        END IF;
    END IF;
END LOOP;
xmldom.freeDocument(doc);
xmlparser.freeParser(myParser);
EXCEPTION
WHEN OTHERS THEN
    xmldom.freeDocument(doc);
    xmlparser.freeParser(myParser);
END;

```

```

PROCEDURE TAHSILAT_BORC_SORGU_GIDEN (ls_sozlesmeno IN VARCHAR2,ls_bankakodu
IN VARCHAR2,ls_anahtar IN VARCHAR2,ENVELOP OUT NOCOPY VARCHAR2)
AS
    str          VARCHAR2(200);
BEGIN
    str := '1'||ls_sozlesmeno||'2'||ls_bankakodu||'3'||ls_anahtar||';
    ENVELOP := XML_OLUSTUR(str,1);
END;

```

```

PROCEDURE TAHSILAT_GIDEN (ls_tarih IN DATE,ls_sozlesmeno IN
VARCHAR2,ls_bankakodu IN VARCHAR2,ls_subekodu IN VARCHAR2,ls_referansno IN
VARCHAR2,ls_tutar IN VARCHAR2,ls_anahtar IN VARCHAR2,ENVELOP OUT NOCOPY
VARCHAR2)
AS
    str VARCHAR2(200);
BEGIN
    str :=
'1'||TO_CHAR(ls_tarih,'dd/mm/yyyy')||'2'||ls_sozlesmeno||'3'||ls_bankakodu||'4'||ls_subekodu||'5'||ls_re
feransno||'6'||NVL(SUBSTR(ls_tutar,1,INSTR(ls_tutar,',',1)-
1),0)||'7'||NVL(SUBSTR(ls_tutar,INSTR(ls_tutar,',',1)+1,LENGTH(ls_tutar)-
INSTR(ls_tutar,',',1)),0)||'8'||ls_anahtar||';
    ENVELOP := XML_OLUSTUR(str,2);
END;

```

```

PROCEDURE IPTAL_GIDEN (ls_tarih IN DATE,ls_sozlesmeno IN VARCHAR2,ls_bankakodu IN
VARCHAR2,ls_referansno IN VARCHAR2,ls_makbuzno IN VARCHAR2,ls_anahtar IN
VARCHAR2,ENVELOP OUT NOCOPY VARCHAR2)
AS
    str VARCHAR2(200);
BEGIN
    str :=
'1'||TO_CHAR(ls_tarih,'dd/mm/yyyy')||'2'||ls_sozlesmeno||'3'||ls_bankakodu||'4'||ls_referansno||'5'||ls_
makbuzno||'6'||ls_anahtar||';
    ENVELOP := XML_OLUSTUR(str,3);
END;

```

```

PROCEDURE BORCSORGU(ls_sozlesmeno IN VARCHAR2,
    ls_bankakodu IN VARCHAR2,
    ls_anahtar IN VARCHAR2,
    P_STR OUT VARCHAR2)
AS
    ENVELOP VARCHAR2(4000);
    EYLEM VARCHAR2(30);

```

```

XMLDOC VARCHAR2(32000);
BEGIN
TAHSILAT_BORC_SORGU_GIDEN(ls_sozlesmeno,ls_bankakodu,ls_anahtar,ENVELOP);

EYLEM:= FONKSIYON_SEC(1);

PK__ONLINE.WEBSERVISBUL;

-- dbms_output.put_line('WEBSERVISADRES: '||WEBSERVISADRES);

XMLDOC:= SOAP_COMMUNICATION (WEBSERVISADRES,EYLEM,ENVELOP);

LOG(EYLEM,ENVELOP,XMLDOC);

_BORC_SORGULAMA(XMLDOC,p_str);
END;

```

```

-----

PROCEDURE BORCSORGU2(ls_sozlesmeno IN VARCHAR2,
                    ls_bankakodu IN VARCHAR2,
                    ls_anahtar IN VARCHAR2,
                    pRetCur IN OUT SYS_REFCURSOR)
AS
ENVELOP VARCHAR2(4000);
EYLEM VARCHAR2(30);
XMLDOC VARCHAR2(32000);
ls_durum CHAR(1);
ls_str VARCHAR2(410);
li_calisma_sekli NUMBER;
BEGIN
TAHSILAT_BORC_SORGU_GIDEN(ls_sozlesmeno,ls_bankakodu,ls_anahtar,ENVELOP);
EYLEM:= FONKSIYON_SEC(1);
PK__ONLINE.SP__ONLINE_CALISMA(2,ls_durum ,ls_str);
PK__ONLINE.WEBSERVISBUL;
if ls_durum = 'I' THEN
    li_calisma_sekli:= PK__ONLINE.CALISMA_SEKLI;
    if li_calisma_sekli = 1 then
        XMLDOC := f_tcp_call('127.0.0.1','18571','JM@/Client@BorcSorgulama@' ||
WEBSERVISADRES || chr(4) || '25' || chr(4) ||
ls_sozlesmeno||chr(4)||ls_bankakodu||chr(4)||ls_anahtar) ;
    else
        XMLDOC:= SOAP_COMMUNICATION (WEBSERVISADRES,EYLEM,ENVELOP);
    end if;
    LOG(EYLEM,ENVELOP,XMLDOC);
end if;
_BORC_SORGULAMA2(XMLDOC,pRetCur);
END;

```

```

-----

PROCEDURE BORCSORGU3(ls_sozlesmeno IN VARCHAR2,
                    ls_bankakodu IN VARCHAR2,
                    ls_anahtar IN VARCHAR2,
                    pRetCur IN OUT SYS_REFCURSOR)
AS
ENVELOP VARCHAR2(4000);
EYLEM VARCHAR2(30);
XMLDOC VARCHAR2(32000);
ls_durum CHAR(1);
ls_str VARCHAR2(410);

```

```

BEGIN
TAHSILAT_BORC_SORGU_GIDEN(ls_sozlesmeno,ls_bankakodu,ls_anahtar,ENVELOP);
EYLEM:= FONKSIYON_SEC(1);
PK__ONLINE.WEBSERVISBUL;
XMLDOC:= SOAP_COMMUNICATION (WEBSERVISADRES,EYLEM,ENVELOP);
LOG(EYLEM,ENVELOP,XMLDOC);
_BORC_SORGULAMA2(XMLDOC,pRetCur);
END;
-----
PROCEDURE TAHSILAT (ls_tarih IN DATE,
ls_sozlesmeno IN VARCHAR2,
ls_bankakodu IN VARCHAR2,
ls_subekodu IN VARCHAR2,
ls_referansno IN VARCHAR2,
ls_tutar IN VARCHAR2,
ls_anahtar IN VARCHAR2,
adi OUT VARCHAR2,
bankakodu OUT VARCHAR2,
ilcekodu OUT VARCHAR2,
islemkodu OUT VARCHAR2,
kontrolsayisi OUT VARCHAR2,
makbuzno OUT VARCHAR2,
referansno OUT VARCHAR2,
soyadi OUT VARCHAR2,
sozlesmeno OUT VARCHAR2,
subekodu OUT VARCHAR2,
tarih OUT VARCHAR2,
ykr OUT VARCHAR2,
ytl OUT VARCHAR2)
AS
ENVELOP VARCHAR2(4000);
EYLEM VARCHAR2(30);
XMLDOC VARCHAR2(32000);

BEGIN
TAHSILAT_GIDEN (ls_tarih,ls_sozlesmeno,ls_bankakodu,ls_subekodu ,ls_referansno
,trim(ls_tutar) ,ls_anahtar,ENVELOP);

EYLEM:= FONKSIYON_SEC(2);

PK__ONLINE.WEBSERVISBUL;

XMLDOC:= SOAP_COMMUNICATION (WEBSERVISADRES,EYLEM,ENVELOP);

LOG(EYLEM,ENVELOP,XMLDOC);

_TAHSILAT_ISLEM(XMLDOC,adi,bankakodu,ilcekodu,islemkodu,kontrolsayisi,makbuzno,referans
no,soyadi,sozlesmeno,subekodu,tarih,ykr,ytl);

END;
-----
PROCEDURE IPTAL(ls_tarih IN DATE,
ls_sozlesmeno IN VARCHAR2,
ls_bankakodu IN VARCHAR2,
ls_referansno IN VARCHAR2,

```

```

ls_makbuzno IN VARCHAR2,
ls_anahtar IN VARCHAR2,
adi OUT VARCHAR2,
bankakodu OUT VARCHAR2,
ilcekodu OUT VARCHAR2,
islemkodu OUT VARCHAR2,
kontrolsayisi OUT VARCHAR2,
makbuzno OUT VARCHAR2,
referansno OUT VARCHAR2,
soyadi OUT VARCHAR2,
sozlesmeno OUT VARCHAR2,
tarih OUT VARCHAR2)

AS

ENVELOP VARCHAR2(4000);
EYLEM VARCHAR2(30);
XMLDOC VARCHAR2(32000);

BEGIN

IPTAL_GIDEN
(ls_tarih,ls_sozlesmeno,ls_bankakodu,ls_referansno,ls_makbuzno,ls_anahtar,ENVELOP);

EYLEM:= FONKSIYON_SEC(3);

PK__ONLINE.WEBSERVISBUL;

XMLDOC:= SOAP_COMMUNICATION (WEBSERVISADRES,EYLEM,ENVELOP);

LOG(EYLEM,ENVELOP,XMLDOC);

_TAHSILAT_IPTAL_ISLEM(XMLDOC,adi,bankakodu,ilcekodu,islemkodu,kontrolsayisi,makbuzno
,referansno,soyadi,sozlesmeno,tarih);

END;

-----
PROCEDURE SP__ONLINE_CALISMA(pi_sorgutip IN NUMBER,ps_durum OUT
VARCHAR2,ps_str OUT VARCHAR2)
--Pi_sorgutip Gişe Sorgulaması için 1, internet ve yöm için 2 değerini alır.
AS
BEGIN

SELECT kod
INTO ps_durum
FROM param
WHERE secimtip = 10000;

if ps_durum = '0' AND pi_sorgutip = 1 then
ps_str := 'HATA'||LPAD('501',3,'0')|| LPAD(SUBSTR(NVL(' ',1),1,100),100,'') ||
LPAD(NVL(' ',20,' ') || LPAD(NVL(' ',20,' ') ||
LPAD(NVL('0',20,' ') || LPAD(SUBSTR(NVL(' ',1),1,20),100,'') ||
LPAD(NVL(' ',20,' ') || LPAD(NVL('0',20,' ') ||
LPAD(NVL('Sistem ONLINE çalışmaya kapatıldı.İşlem yapamazsınız.',100,''));
end if;

EXCEPTION
WHEN OTHERS THEN

```

```
ps_durum := '1';  
END;
```

```
-----  
FUNCTION CALISMA_SEKLI  
RETURN NUMBER IS  
li_calisma_sekli number(2);  
BEGIN  
begin  
SELECT to_number(kod) into li_calisma_sekli from param  
where secimtip = 8888;  
exception  
when others then  
li_calisma_sekli := 0 ;  
end;  
RETURN (li_calisma_sekli);  
END;  
end; --PK ONLINE
```

ÖZGEÇMİŞ



Ad Soyad: Özgür Armağan

Doğum Yeri ve Tarihi: İzmir – 17/04/1977

Adres: İstanbul Teknik Üniversitesi – Kontrol ve Otomasyon Mühendisliği Bölümü
Maslak İstanbul.

E-Posta: ozgurarm@yahoo.com

Lisans: İstanbul Teknik Üniversitesi – Kontrol ve Bilgisayar Mühendisliği (1999).

Yüksek Lisans: Boğaziçi Üniversitesi – Sistem ve Kontrol Mühendisliği (2004).

Mesleki Deneyim ve Ödüller: Kontrol teorisi ve farklı uygulama alanları ile ilgili olarak on yedi senelik tecrübeye sahiptir. Bankacılık ve finans sektöründe, bilgi teknolojileri alanında on dört seneye yakın bir süre yazılım geliştirme, yazılım projeleri yöneticiliği ve teknik danışmanlık deneyimi bulunmaktadır.

Yayın ve Patent Listesi:

- Gündem T. İ. ve **Armağan Ö.** (2006). Efficient storage of healthcare data in XML-based smart cards. J Computer Methods and Programs in Biomedicine 2006; 81(1):26-40. doi:http://dx.doi.org/10.1016/j.cmpb.2005.10.007.

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Armağan Ö.** ve Gören-Sümer L. (2014). Feedback Control for Multi-Resource Usage of Virtualised Database Server. J Computers and Electrical Engineering 2014; 40(5):1683-1702. doi:http://dx.doi.org/10.1016/j.compeleceng.2014.04.017.

