T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BULUT ORTAMINDA KULLANICI VERİLERİNİ YAZMA VE OKUMA AMAÇLI YÜK DENGELEME YAKLAŞIMLARI VE BİR ÇÖZÜM ÖNERİSİ

NURULLAH KILIÇ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
DR. ÖĞR. ÜYESİ YUNUS EMRE SELÇUK

İSTANBUL, 2018

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BULUT ORTAMINDA KULLANICI VERİLERİNİ YAZMA VE OKUMA AMAÇLI YÜK DENGELEME YAKLAŞIMLARI VE BİR ÇÖZÜM ÖNERİSİ

Nurullah KILIÇ tarafından hazırlanan tez çalışması 02.05.2018 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Dr. Öğr. Üyesi Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Dr. Öğr. Üyesi Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi _____

Doç Dr. Veli HAKKOYMAZ

Yıldız Teknik Üniversitesi _____

Doç Dr. Hasan SÖZER

Özyeğin Üniversitesi _____

ÖNSÖZ

Bu tezin ortaya çıkmasında en büyük pay sahibi şüphesiz danışmanım Dr. Öğr. Üyesi Yunus Emre SELÇUK'a aittir. Motivasyonumun en düştüğü zamanlarda bile beni tekrardan akademik çalışmalara döndüren eşsiz yönlendirmeleri ve birikimiyle sürekli desteğini hissettiğim hocama teşekkürlerimi sunarım. Bunun yanı sıra Yüksek Lisans eğitimim boyunca bana desteğini sürekli hissettiren aileme ve mesai arkadaşlarıma özellikle Screen Tracking uygulamasının geliştirilmesinde değerli katkılarını sunan ve tez süresince bahsi geçen mimarilerin kurgulanması, testlerinin yapılması ve mimari üstündeki değişikliklerin sisteme yansıtılması konusunda bana sabırla destek olan Sword-IT çalışanlarına teşekkürlerimi sunarım.

Son olarak başta anneme, babama ve ilkokuldan bu yana eğitim ve öğretim hayatıma yön veren bütün öğretmenlerime beni bugünkü ben yapmalarındaki katkılarından dolayı şükranlarımı sunarım.

Şubat, 2018

Nurullah KILIÇ

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

$avgt_{i-1}$	Bir önceki iterasyonda hesaplanan kuyruk yük değeri
f(avg)	Sistem yük değeri hesaplama fonksiyonu
L	Sistemdeki anlık ortalama müşteri sayısı
Qt_i	Son iterasyonda hesaplanan kuyrukta bekleyen yük değeri
W	Bir müşterinin ortalama hizmet alma süresi
α	Bir önceki iterasyonun toplam ortalamadaki ağırlığı
λ	Birim zamanda hizmet almak için gelen müşteri sayısı

KISALTMA LİSTESİ

AWS	Amazon Web Services
GNU	GNU Unix değildir (GNUs Not Unix)
GSI	Grid Güvenlik Altyapısı (Grid Security Infrasturcture)
IaaS	Servis Olarak Altyapı (Infrastructure As A Service)
MPI	Mesaj iletim arayüzü (Message Passing Interface)
MPPs	Yoğun-Paralel İşlemciler (Massively Paralel Processors)
NASA	Ulusal Havacılık ve Uzay Dairesi (National Aeronautics and Space Administration)
PaaS	Servis Olarak Platform (Platform As A Service)
PAYG	Ödediğin Kadar Kullan (Pay As You Go)
PVM	Paralel Sanal makinalar (Parallel Virtual Machines)
SaaS	Servis Olarak Yazılım (Software As A service)
SLAs	Güvenlik Seviyesi Anlaşmaları (Secure Level Agreements)
SMP	Symmetric Multiprocessing
SSI	Tekil Sistem İmajı (Single System Image)
UNICORE	(Uniform Interface to Computing Resource)
VO	Sanal Organizasyonlar (Virtual Organizations)

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

BULUT ORTAMINDA KULLANICI VERİLERİNİ YAZMA VE OKUMA AMAÇLI YÜK DENGELEME YAKLAŞIMLARI VE BİR ÇÖZÜM ÖNERİSİ

Nurullah KILIÇ

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Dr. Öğr. Üyesi Yunus Emre SELÇUK

Bulut Bilişim (Cloud Computing) hem akademik ortamda önemli araştırma konuları sunan hem de sektörde pratik olarak yaygın şekilde kullanılan bir yöntem haline gelmiştir. Dağıtık Sistemler (Distributed Systems) alanındaki çözülmüş ve çözülmemiş problemler Bulut Bilişim alanına da taşınmaktadır. Bu tür sistemlerde toplam işin mevcut sistemlere en iyi biçimde dağıtılması, başarımı en üst düzeye çekmektedir. Yük dengeleme (load balancing) adı verilen bu işlem, kullanılan farklı yaklaşımlara göre farklı ek yükler ve/veya en iyi olmayan sistem başarımı ortaya çıkartabilmektedir. Bulut bilişim veri merkezlerinde yük dengeleme etkin bir araştırma alanı olmayı sürdürmektedir.

Tez çalışması kapsamında literatürdeki yük dengeleme yaklaşımları incelenerek özellikle işlemsel olmayan (non-transactional) yöntemlerden en ümit verici olanlar seçilerek açık kaynak bulut ortamında denenmiştir. Ayrıca çalışmada farklı veri tipleri için başarılı bir parçalama ekseni (sharding column) seçimi ve bu seçimin yük dengelemeye ve performansa etkileri incelenmiştir.

Yapılan incelemelerde kullanılmak üzere seçilen Screen Tracker uygulaması, çalışanların verimliliğini artırmaya yönelik bir yazılımdır. Bu yazılım Sword-IT bünyesinde geliştirilmiş bir uygulama olup, çalışanların hangi uygulamada ne kadar vakit harcadıklarına dair verileri toplarlayıp analizini yaparak çalışan verimliliğini artırmayı hedeflemektedir. Uygulama, çalışanların verilerinin çevrim dışı depolanıp, daha sonra çevrimiçi sorgulanabilir hale getirilmesi amacına yönelik olarak, uygulama tarafından üretilen verilerin yazımı ve okunması sırasında yük dengeleme yaklaşımlarından

yararlanarak tez çerçevesinde yeniden tasarlanmıştır. Screen tracker uygulamasından gelen verilerin kaydedilmesi için dağıtık bir yapı kurgulanmıştır. Bu yapıda, bir adet koordinatör düğümü, yük durumuna göre kümelenebilir yeni yük düğümleri, verilerin tutulacağı saklama düğümleri ve verilerin okunacağı okuma ucu (read end-point) düğümler kurgulanmıştır. Screen Tracker uygulamasının çalışması ve mimarisi hakkında ayrıntılı bilgiye 4.Bölümde yer verilmiştir.

Yapı içerisinde, PostgreSQL Veri Tabanı Yönetim Sistemi (VTYS)'den yararlanılmıştır. Birden fazla sunucuya verilerin belli sütunlara göre parçalanarak dağıtımı yapılmış, sonuçların performans üstündeki etkileri araştırılmıştır. Bu çerçevede, mevcut olan yaklaşımlar incelenmiş, en ümit verici olanları test edilerek ve önerilen yöntem ile kıyaslanmıştır. Round Robin yaklaşımı temel olarak alınıp diğer yöntemlerin bu yaklaşıma göre sağladığı kazanımlar kıyaslanmıştır.

Anahtar Kelimeler: Ekran Takip Uygulaması, Dağıtık veri saklama, Bulut Ortamında veri yazma, Bulut ortamında yük dengeleme, Dağıtık Veri Tabanı

ABSTRACT

LOAD BALANCING APPROACHES FOR USER DATA WRITING AND READING IN CLOUD ENVIRONMENT AND A NEW APPROACH

Nurullah KILIÇ

Department of Computer Engineering

MSc. Thesis

Adviser: Asist. Prof. Yunus Emre SELÇUK

Cloud computing has become a magnificent working area both in academical and industrial researches. The solved and unsolved problems in Distributed Systems are moved into the Cloud Computing area. Distributing the load on to the whole system in a balance, levarages the performance that is called load balancing in these kind of systems. Load balancing is still a niche working area for Cloud Computing data centers.

Load balancing approaches are reviewed and especially the most promising non-transactional methods are tested in a cloud environment. Also in the study, a successful sharding column selection for different data types and the effect of this selection on load balancing and the corresponding performance is examined.

The Screen Tracker application, developed to increase the productivity of the employees, is used for aforementioned tests and examinations. This application is developed by Sword-IT and aims to increase employee productivity by collecting and analyzing the data about how much time the employees spend at work by using their computers. The application is designed by taking advantage of load balancing approaches during the writing and reading of application-generated data for the purpose of storing employee data offline and then querying online. A distributed structure will be created to record the data from the screen tracker application. In this structure, one coordinator node, new load nodes that can be clustered according to the load case, retention nodes to hold the data, and read end-point nodes to read the data will be constructed. Detailed information about the operation of the Screen Tracker application and its architecture is included in Chapter 4.

PostgreSQL Database Management System is used as a Database Management System within the Project. The application data is sharded by a column and distributed more than one server in terms of scaling and load balancing and then performance affects of different topologies are compared. Used methodologies and approaches in the industry and academic researches are reviewed and the most promising ones are tested and compared. Also the performance gain according to round robin is evaluated during the work.

Keywords: Screen Tracking Application, distributed data storage, data writing in cloud, load balancing, distributed database

BÖLÜM 1

GİRİŞ

1.1. Literatür Özeti

Bulut Bilişim (Cloud Computing) hem akademik ortamda önemli araştırma konuları sunan hem de sektörde pratik olarak yaygın şekilde kullanılan bir yöntem haline gelmiştir [1]. Dağıtık Sistemler (Distributed Systems) alanındaki çözülmüş ve çözülmemiş problemler Bulut Bilişim alanına da taşınmaktadır. Bu tür sistemlerde toplam işin mevcut sistemlere en iyi biçimde dağıtılması, başarımı en üst düzeye çekecektir. Yük dengeleme (load balancing) adı verilen bu işlem, kullanılan farklı yaklaşımlara göre farklı ek yükler ve/veya en iyi olmayan sistem başarımı ortaya çıkartabilmektedir [2]. Bulut bilişim veri merkezlerinde yük dengeleme etkin bir araştırma alanı olmayı sürdürmektedir [3].

Mevcut yük dengeleme yaklaşımları statik ve dinamik olmak üzere ikiye ayrılabilir. Statik yaklaşımlar daha basit olmakla birlikte çalışma anında ortaya çıkan fırsatları değerlendiremezler. Dinamik yaklaşımlar ise NP-tam(NP-Complete, NP: nondeterministic polynomial time) sorunlardır. NP-tam olan bir soruna önerilen bir çözüm hızlı bir biçimde doğrulanabilse de, ilk başta böylesi bir çözümü elde etmenin hızlı bir yolu bulunamamaktadır. Dinamik yük dengeleme yaklaşımları üzerinde çalışmalar sürmektedir: Dengelemeye yönelik dinamik veri toplanması sırasında yapılan sistemler arası haberleşmenin getireceği ek yükün azaltılması [3] ve acil (immediate) yerine zamanlamalı (scheduled) görev atanması yöntemlerinin incelenmesi [4] değinilen çalışma alanları arasındadır. Ayrıca, en başta veri yazımı sırasında, en önemli sorunlardan biri de dağıtımın hangi eksene göre yapılacağıdır(sharding column) [5].

1.2. Tezin Amacı

Teknolojinin hızla gelişmesiyle birlikte internete bağlanan kullanıcı sayısı ve internete bağlanan cihaz sayısında çok büyük artışlar olmaktadır. Artan kullanıcı ve cihaz sayısı ile birlikte bunlar tarafından üretilen veri miktarı da katlanarak artmaktadır. Veri miktarındaki artışa paralel olarak bu verilerin güvenli bir şekilde saklanması ve bu verilere hızlı ve güvenli bir şekilde erişilebilmesi amacıyla çok sayıda araştırma ve geliştirme çalışması yapılmaktadır.

Dünya'da bulunan tüm verilerin %90'ı geçtiğimiz iki yıl içerisinde üretilmiştir. Her gün Dünya'da 2,5 kentilyon (10*18) bayt veri üretilmektedir. Walmart şirketi her saat bir milyondan fazla müşteri işlemini yönetmektedir. Bu işlemlerin toplam boyutu 2,5 Petabayt olup çevrimiçi durumdaki veritabanına kaydedilmektedir. Kredi Kartı Dolandırıcılık Sistemi Dünya üzerinde 2 milyarın üzerinde hesabı yönetmektedir. Facebook 45 milyardan fazla fotoğrafı sistemlerinde depolamaktadır[6]. Veri miktarları böyle büyük boyutlara ulaşmışken ve bu hızla artarken en önemli sorun sistemlerde verileri sağlıklı bir şekilde depolamak ve bu verilere hızlı bir şekilde erişebilmektedir.

1.3. Hipotez

Bu tez çalışmasında, çalışanlara ait çalışma saatlerinde oluşan belli verilerin internet üzerinden merkezi sistemlere iletilerek depolanması, sorgulanması, çalışanların verimliliğine ve iş zekâsına yönelik analiz işlemlerinin gerçekleştirilmesi temeline dayanan bir sistem hayata geçirilecektir. Binlerce hatta on binlerce çalışana ait verilerin depolanması ve servise hazır hale getirilmesinde yaşanılan teknik zorlukların üstesinden gelmek amacıyla tasarlanan dağıtık sistemlerin performansını artırmaya yönelik yapılan işlem ve iyileştirmeler karşılaştırmalı olarak analiz edilecektir. Analiz sonuçlarına göre, önerdiğimiz yöntem ve sistemin başarımı gösterilecektir.

Tez metni şu şekilde bölümlenmiştir: Birinci bölümde geniş ölçekli sistemler üzerinde genel bir literatür araştırması yapılacaktır. Farklı geniş ölçekli sistemlerin mimari yapıları, bilgi sistemleri dünyasında büyük ölçekli verilerin yönetilmesinde kullanılan farklı modeller incelenecektir. İkinci bölümde büyük ölçekli verilerin yönetilmesinde yaygın olarak kullanılan, ayrıca tarihsel verilerin yönetimindeki yapıya en yakın sistem

olan dağıtık veri tabanı sistemleri hakkında yapılan detaylı çalışma ve araştırmaya yer verilmiştir. Üçüncü bölümde önerilen sistemin genel yapısı ve işleyişi, sistemin bileşenleri, kullanıcı verilerinin depolama (retention) düğümlerine dağıtılması ve sorgulanması işlemleri, başarımı artırmak için gerçekleştirilen iyileştirmeler detaylı olarak anlatılmıştır. Dördüncü bölümde alternatif olarak önerilen Hadoop mimarisi üzerinde sistem test edilmiş ve sonuçlar irdelenmiştir. Son bölümde ise sistemin artı ve eksileri alternatif mimari ile karşılaştırmalı olarak incelenmiş ve bundan sonra yapılabilecek çalışmalar için öneriler sunulmuştur.

BÖLÜM 2

GENİŞ ÖLÇEKLİ SİSTEMLER

Büyük problemleri küçük parçalara ayırarak daha kolay şekilde çözülmesinde önemli bir metot olan paralel işleme hem yüksek performans gerektiren bilimsel çalışmalarda hem de genel amaçlı kullanımlarda kullanımı son yıllarda önemli oranda artmıştır. Paralel işlemenin kabul görmesini ve yaygınlaşmasını sağlayan iki önemli gelişme ise dağıtık sistemlerin yaygınlaşması ve yoğun-paralel işlemcilerdir (MPPs: Massively Paralel Processors). Yoğun-paralel işlemciler günümüzde en güçlü bilgisayarlar olarak bilinmektedir. Sayısı birkaç yüzden başlayıp binleri bulan işlemcilerin, yine binlerle ifade edilen belleklerle birlikte tek bir kabin içerisinde yer aldığı süper bilgisayarlardır.

2.1 Küme (Cluster) Yapılar

2.1.1 Mimari Yapı

Küme yapıların ilk ortaya çıkış nedeni maliyeti düşük dağıtık bir mimariye ihtiyaç duyulmasıydı. Kişisel bilgisayarlar ve düğüm olarak adlandırılan sunucular birbirlerine Ethernet adı verilen basit ağ altyapıları ile bağlıydı. Bakım maliyetinin düşük olması ve kolay genişleyebilirlik günümüzde küme yapıların bütün ölçeklerde kullanılabilir olmasını ve on binlerce düğüm sayılarına ulaşmasını sağladı.

Küme yapılar birbirinden bağımsız çalışan bir grup sunucunun tek bir sunucu gibi çalıştığı yapılardır. Sistemlerin büyüyerek tek bir sunucu ya da veritabanı sisteminin hizmet vermekte olduğu uygulamalara, daha çok performans açısından yeterli olmadığı durumlarda çözüm olması amacı ile tasarlanmış bir yapıdır. Bu yönde yapılan ilk çalışma Beowulf ‘tur [7], [8]. Beowulf NASA’da geliştirilmiş donanımsal olarak bir süper bilgisayarın yapabileceği işi taklit edebilen küme yapı olarak tasarlanmıştır. Daha düşük

maliyetli ve pratik olmayı amaçlayan bu yapı işletim sistemi olarak GNU/Linux kullanmakta olup bu işletim sisteminin üzerinde Message Passing Interface(MPI) ve Parallel Virtual Machine(PVM) sistemlerini barındırmaktadır [9]. Yapının en belirgin özelliği ise ihtiyaç duyuldukça genişleyebilir olması ve standart hale getirilmiş yazılım paketlerinin çalıştığı daha büyük kümelerin eklenebilir olmasıdır.

Küme yapısına farklı bir yaklaşım olarak süreli yüksek verim hesaplama Condor tarafından gerçekleştirildi [10]. Bu çalışma herhangi bir süper bilgisayarın yaptığı işi taklit etmektense, eşlenik biçimde çalışabilen, amacı büyük hacimli paralel iş parçacıklarını çalıştırmak olan ve yoğun hesaplamalar yapabilen uygulamaları barındıran bilgisayarlardan oluşan bir projeydi. Günümüzde Condor küme yapısı NASA gibi kamu kurumları tarafından binlerle ifade edilen düğüm sayılarınca geniş bir ölçekte kullanılmaktadır. Büyük endüstri teknoloji devlerinden Google, Yahoo gibi şirketlerin ihtiyacı büyük ölçekli verileri en düşük maliyetle maksimum performansta işleyebilmek ve kullanabilmekti. Bu firmalardan Google, altyapısı hakkında ayrıntılı bilgi vermese de geniş bir kitle tarafından inanılan birkaç milyon işlemci ile çalışan ve on binlerle ifade edilen küme yapı gruplarının yer aldığı 25 veri merkezine sahip olduğudur. Disk üniteleri direk olarak işlemciye bağlı iken işlemciler standart ethernet bağlantıları ile birbirine bağlıdır.

Temel anlamda küme yapılar iki gruba ayrılır. Mimari yapıları ve yapısal çalışma şeklinin anlaşılabilmesi için iki farklı konu başlığında değinilecektir.

2.1.1.1 Yük Dağılımlı Küme Yapılar

Bu küme yapısında güçlü tek bir makine gibi görünmek amacıyla birçok bilgisayarın ya da sunucunun birbirlerine link ile bağlanması sonucu oluşmaktadır. Bu yapılar tekil sistem görüntüsü (Single System image-SSI) olarak da adlandırılır. Diğer sistemlerin aksine SSI sistemler görev işleri yönetmektense iş parçacığı temelli çalışmaktadır. Görevler kullanıcılar tarafından yerel olarak başlatılsa da yük dengesinin sağlanması için diğer düğümlere iş parçacıklarının nakledilmesi sağlanır. İlk SSI uygulamaları MOSIX tarafından gerçekleştirilmiştir [11], [12]. Otomatik kaynak tespiti, dinamik iş yükü dağılımı tek bir bilgisayar üzerinde bulunan birden fazla işlemci ile gerçekleşmekteydi.

Kerrighed tarafından ise ikinci bir SSI geliştirilmişti [13], [14]. Bu çalışma MOSIX ile aynı prensipler üzerine tasarlanmış fakat MOSIX'e kıyasla birkaç avantajı vardı. Bunlar oldukça geniş paylaşımlı bellek yönetimi ve iş parçacıkları için şeffaf kontrol noktaları oluşturulmasıdır. Bu kontrol noktaları iş parçacıklarının görevin tamamının bitirilmesinde çalışırken hangi adımda kaldıklarını görüntüleme işlemidir. Grafik işleme, büyük depolama alanlarının düzenlenmesi, çevrimiçi oyun gibi çalışmalarda yük dağılımlı küme yapılar tercih edilmektedir.

Yük dağılımlı küme yapılar genellikle çok fazla bellek ve işlemci gücü isteyen uygulamalar için kullanılmaktadır.

2.1.1.2 Yüksek Kullanılabilirlik Küme yapılar

Küme yapıların temel amacı yüksek geçerlikte servisler sağlamaktır. Bu işlem için veri birden fazla düğümde çoğaltılmış ve önbelleğe alınmış şekilde bulunmaktadır. Normalde hizmet vermekte olan bir sunucuda birtakım uygulamalar hata verecek ya da cevap vermeyecek olursa sistem sorumlu kişinin müdahalesi olmadan yanıt vermez. Fakat fail-over yöntemi ile bu durum yüksek geçerliliği olan küme yapılarda ortadan kalkmıştır. Bu teknoloji ağda yapılan basit yönlendirmelerden karmaşık uygulamaların çalıştığı tüm seviyelere kadar küme yapıya dâhil edilebilmektedir. Bu teknoloji küme yapıda herhangi bir sorun olması durumunda çalışmanın kaldığı yerden yapıda bulunan başka bir düğümden devam etmesine olanak sağlar. Fail-over yöntemi ile çalışan küme yapının ortak bir disk alanında ve en az iki sunucudan oluşması gerekmektedir.

2.1.2 Izgara (Grid) Yapılar

Küme yapılar oldukça etkili ve birçok organizasyon tarafından geniş bir alanda yaygın şekilde kullanılabilirliğini kanıtlamıştır. Fakat burada akla gelen soru daha güçlü dağıtık sistemler elde etmek için bu organizasyonların kaynaklarının kullanılıp kullanılmayacağıdır. Kaynakların kullanılıp kullanılmayacağı konusunda karşımıza ızgara yapılar çıkmaktadır. Izgara yapılar geniş ölçekli dağıtık uygulamalara çözüm getirmek için dağıtık geniş alan ağlarının kaynaklarını kullanarak çözüm getirmeye çalışmaktadır.

Izgara terimi ilk defa "Yüksek seviyeli işlem kapasitesine erişebilmek için güvenilir, tutarlı, yaygın ve pahalı olmayan donanım ve yazılım altyapısıdır" [15] şeklinde

tanımlanmıştır. Elektrik güç şebekesinden esinlenerek elde edilen bu fikirde, dağıtık sistemin, elektrikli bir cihazın prize takılması kadar kolay olabilecek şekilde ızgara kullanıcılarına her an hesaplama ya da işlem imkânı vermesi gerekmektedir. Bu tanım ızgara yapısının ne olduğunu tam manasıyla anlamanın gerektiği her an kullanılır olmuştur. Foster, Kesselman ve Tuecke [16] çalışmalarında "Paylaşımlı kaynak yönetimi, dinamik problem çözümü, çoklu kurumsallık ve sanal organizasyonlar" şeklinde dağıtık sistemlerin tanımını yeniden düzenlemeye çalışmışlardır.

Sanal organizasyonlar kavramı ızgara yapıların merkezindedir. İlk olarak ızgara yapılar bir görevi yerine getirmek için kaynak paylaşımını gerekli gören güvensiz katılımcıların olduğu bir konsorsiyum olarak adlandırılabilir. Bu bağlamda paylaşım uzaktan yazılım, bilgisayarlar ya da verilere direk erişim gibi karmaşık etkileşimleri işaret etmektedir. Bu etkileşimler yüksek derecede kontrol altına alınmıştır. Bu kaynak sağlayıcıların ya da talepte bulunanların nereye hangi şartlar altında kimin erişeceğinin bir otorite tarafından idare edilmesidir. Katılımcıların paylaşım kurallarının belirlendiği bu yapıya sanal organizasyon denir.

Izgara buraya kadar iyi huylu dağıtık sistem olarak göründü. [17]'de de belirtildiği gibi nadir görülen hatalar, asgari düzeyde güvenlik, tutarlı yazılım paketleri ve basit paylaşım politikaları gibi varsayımlar küme yapılar için geçerli olsa da ızgara yapılara dayandırılamaz. Foster [18] herhangi bir ızgara yapının karşılaşacağı 3 gerekliliği tanımlamıştır.

2. Merkezi denetime tabi olmayan kaynakları koordine etmesi: Grid yapılar farklı idari merkezlerin yönetimine tabi kaynakları koordine eder. Bu işlemi yapması için güvenlik, uygulanması gereken politikalar, erişim kontrolleri gibi adımları aşması gerekir. Yapıdaki sistemlerin birbirlerine olan güvensizlikleri ya da güvenlik gerekçeleri nedeniyle bu adımlar merkezi bir yerde tanımlanamaz.
3. Standartların kullanımı, açık ve genel amaçlı protokol ve arayüzler: Kaynak paylaşımı kimlik doğrulama, yetkilendirme, kaynak keşfi, kaynağa erişim gibi işlemler ile tanımlanabilecek çok amaçlı protokollere ve arayüzlere dayanır. Standartların kullanımı bu bağlamda önemlidir. Standartların kullanımı üyeler ile

geniş sayıda uygulama için yeniden kullanılabilen yapı blokları arasındaki etkileşimi kolaylaştırır.

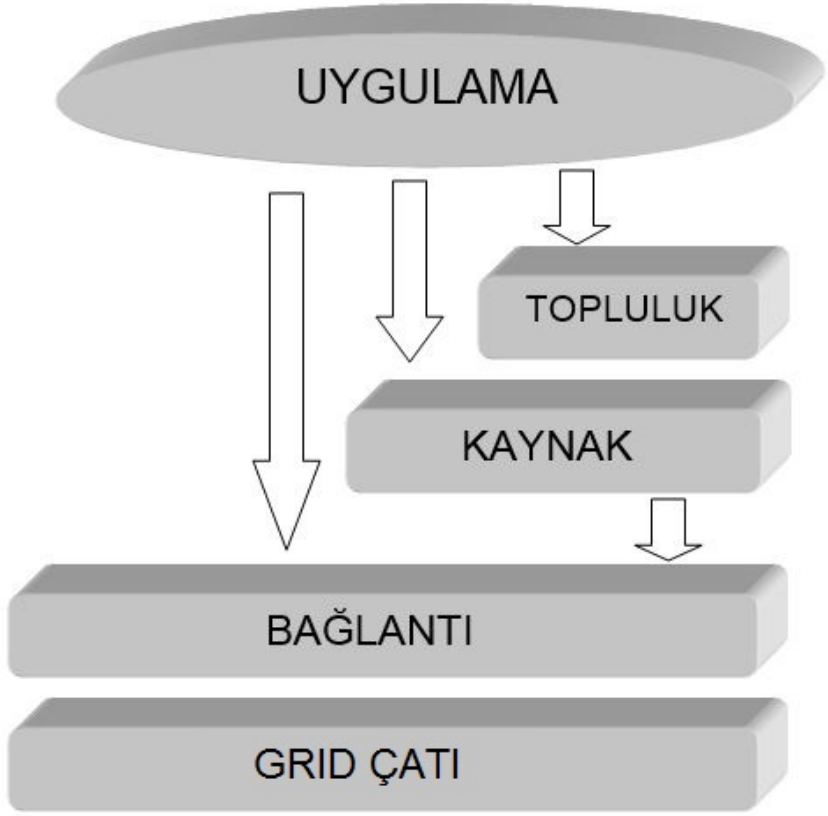
4. Servislerin önemsiz olmayan nitelik teslimatları: Izgara uygulamalarının geniş coğrafik alanlara yayılmasını sağladığından beri, verimlilik, kullanılabilirlik, güvenlik, birçok kaynağın kolektif çalışabilmesi ve karmaşık kullanıcı talepleri gibi hizmetleri sağlaması gerekmektedir. Bu sayede parça parça bir araya gelerek hizmet vermekte olan dağıtık sistemlerden daha güçlü bir dağıtık yapı haline gelir.

2.1.3 Mimari Yapı

Izgara mimarisinin (Şekil 2.1)' de görüldüğü ve [16]'de de belirtildiği gibi tasarım ve uygulamalarında bazı sınırlamalar getirdiği görülebilir.

Izgara çatı (Grid Fabric), ızgara yapısının yer aldığı sistemin işlemci, bellek gibi farklı katmanlara ulaşmasını sağlar. Bu katman mevcut olan sistem ya da bileşenleri kullanır. Çatı katmanın tanımı, erişim kaynakların izlenmesi ve yönetilmesi amacıyla birtakım sürücüler ya da bileşenler ile adapte olmuş geniş bir sistem için yapılmakta olan birleşik bir arayüz gibi düşünülebilir.

Bağlantı (Connectivity) Katmanı ızgara katmanları arasında kimlik doğrulama ve yetki protokolleri gibi güvenlik ve hızlı bağlantı işlemlerinden sorumludur. Bu bağlantı ve güvenlik işlemlerini yapan katmana aynı zamanda ızgara güvenlik altyapısı (GSI- Grid Security Infrastructure) da denir.



Şekil 2.1 Grid Yapısı

Kaynak (Resource) katmanı ızgara yapısında yer alan tekil kaynaklara erişmek isteyen kullanıcılar için bağlantı katmanının üstünde yer almaktadır. Bu yapıda ayrıca ihtiyaca göre sisteme protokol eklenmesi işlemleri yapılmaktadır. Kaynak katmanı üstünde yer alan diğer katmanlar için de iki önemli kolaylık sağlar. Biri kaynakların durumunu sorgulayabilme yeteneği diğeri ise kaynaklara erişebilmek için gerekli olan müzakere adımlarıdır. Bireysel kaynakların yayınlanması, paylaşılması ve kullanıcı hesap işlemleri bu katmanda yapılmaktadır.

Topluluk (Collective) katmanı bireysel kaynakların yönetimi için hem kaynak hem de bağlantı katmanının üzerinde bulunmaktadır. Kaynağın keşfi, planlanması ve bir araya getirilme işlemlerinin yapılmakta olduğu katmandır.

Uygulama katmanı ise tüm bu katmanların, uygulamaların sanal organizasyona (VO) tümleştirilebilmesi için gereklidir.

2.1.4 Özel Yazılım (Middleware)

Izgara yapısının oluşturulmasını ve yönetimini kolaylaştırmak için bazı topluluklar, profesyonel ekipler, üniversite konsorsiyumları özel bir yazılım geliştirmek için bir araya gelmişlerdir. Yapılacak çalışmanın, bilgisayarlarda bulunan işletim sistemlerinin çalışan programlar için yapmakta olduğu görevi yerine getirmesi gerekmektedir.

Globus Toolkit standart ızgara hesaplama özel yazılımını akademik ve endüstri alanında fiili duruma getiren ve açık kaynak ızgara topluluğu olan Globus birliği tarafından geliştirilmiştir [19]. Globus Toolkit de kendisini geliştiren birlik gibi açık kaynaklı ve açık mimarili bir araçtır.

Globus Topluluğu ızgara yapısının oluşturulmasını ve protokollerin standart hale getirilmesini amaçlamaktadır. Globus, güvenlik, kaynak yönetimi, veri yönetimi, bağlantı ve hata yönetimi için çözümler sunmaktadır. Globus Toolkit diğer katılımcıların uyumluluğunu bozmadan her bir katılımcının ızgara yapısında istenilen özelliği seçebildiği modüler bir şekilde tasarlanmıştır [20].

2.2 Bulut Yapılar

Teoride ızgara yapısı geniş bir kullanıcı ya da katılımcının aynı hedef için tüm kaynaklarını ortak şekilde kullanabilmesi sayesinde büyük hesaplamaları yapabilecek yüksek bir potansiyele sahiptir. Fakat pratikte bu çok sayıda kullanıcı ya da katılımcı birtakım sorunlara neden olmuştur. Her bir kullanıcının kendi kaynağından sorumlu olduğu ve dilediği zaman yapıya girebileceği ve dilediği zaman yapıdan çıkabilmesi sayesinde ızgara yapısı doğal olarak dinamik bir yapı haline geldi ve bu da kaliteli bir çalışma ortaya koyma işini bir hayli zorlaştırdı. Bu yüzden sanal organizasyonlar kullanımı o kadar kolay olmayan karmaşık bir güvenlik ve yönetim politikası sundular. Hem kullanıcılar hem de uygulama geliştiriciler ızgara yapısı için “gereğinden fazla yapılandırma” tanımı yaptılar. Bu amaçla Bulut paradigması (Clouds) sanallaştırılmış hesaplamalar ve depolama teknolojileri yapan gelecek nesillerin veri merkezlerinde hizmet verecek kullanıcı dostu ve güvenilir yapılar olarak ızgara teknolojisinin daha gelişmiş bir yapısı şeklinde ortaya çıktı [21], [22]. Izgara gibi bulut da elektriğin nereden

geldiğine bakmadan herhangi bir cihazı prize takar gibi problemin çözümü için kullanıcılar için hesaplama gücünü en aza indirerek gerekli olan hesaplama kaynaklarını tek bir varlık gibi kullanılıp federe edilmesini sağlar.

Izgara yapısının aksine bulut yapısı, kaynakların paylaşıldığı bir konsorsiyum oluşturmak yerine ekonomik bir model oluşturmak amacını güder. Daha belirgin ifade edecek olursak bulut yapısının sahipleri kullanıcılarına ödediğiniz kadar bulut kaynaklarından faydalanabilirsiniz diyen servis sağlayıcılarıdır. Bu sayede CPU kullanımı ya da bant genişliği örneklemeleri ile kullanıcılar sadece sorunlarını çözecek kadar kaynağa erişebilmek için para ödemektedirler.

Özellikle kullanıcılar, sağlayıcılar tarafından yayınlanan ve kullanıcılar ile yapılan sözleşmede belirtilen servis seviyesi anlaşmaları (SLAs-service level agreements) gerekli olan hizmet kalitesini belirtirler. Kullanıcılar bulut yapısında uygulamalara ve verilere dünyanın herhangi bir yerinden diledikleri zaman erişebilme garantisi altındadırlar. Garantinin veriliyor olması bulut yapısının son derece sağlam ve kullanılabilir olduğunun göstergesidir. [23]‘ de Buyya’nın da belirttiği gibi “Bulut yapısı, birbirine bağlı ve sanallaştırılmış bilgisayarların dinamik şekilde hazırlanıp bir veya daha fazla birleşik hesaplama kaynaklarının hizmete sunulduğu, servis sağlayıcılar ile kullanıcılar arasında yapılan servis seviyeli anlaşmalara dayanan paralel ve dağıtık sistemlerin bir türüdür.”

Ekonomi odaklı model olarak benimsenen bulut yapısının bazı ilginç avantajları vardır. Bu avantajlarından ilki düşük başlangıç ve bakım maliyetidir. Bulut büyük miktarlardaki hesaplama maliyetini sermaye giderlerinden işletme giderlerine çevirir. Bu herhangi bir sabit ücreti olmayan ya da kendilerine özgü bir altyapı almak istemeyen tek bir defa ya da seyrek aralıklarla sistemi kullanacak olan kullanıcılara kolaylık sağlar (PAYG system). Bulut sistemleri dinamik ve rekabetçi teklif sunarak toplam maliyeti düşürmeyi hedeflemektedir [24]. Bu sayede yatırım giderlerinin işletme giderlerinde kullanılmasını sağlamış olur.

Bulut sisteminin diğer bir avantajı elastikiyettir. Kaynakların hazır şekilde beklemesine bağlı olarak kullanıcılar ihtiyaçları doğrultusunda uygulamalarını büyültebilir ya da küçültebilir. Bu esneklik kullanıcıların veri işlem zamanlarının üst sınıra çıktığı zamanda

bu işlemler için pahalı donanımlar almasının önüne geçer. Yani kullanıcılar işletme ya da uygulamalarının büyüdüğü kadar ödemektedir.

Ölçeklenebilirlik de bulut yapılarının bir avantajıdır. Servis sağlayıcılar kullanıcı sayısının mümkün olduğu kadar artması durumunda bulut yapısını kolaylıkla büyütebilecektir. Aynı zamanda kullanıcılar açısından iş hacimlerinin artmasına bağlı olarak erişecekleri kaynak sayısının artması da yine ölçeklenebilirliğin getirdiği yararlardan biridir.

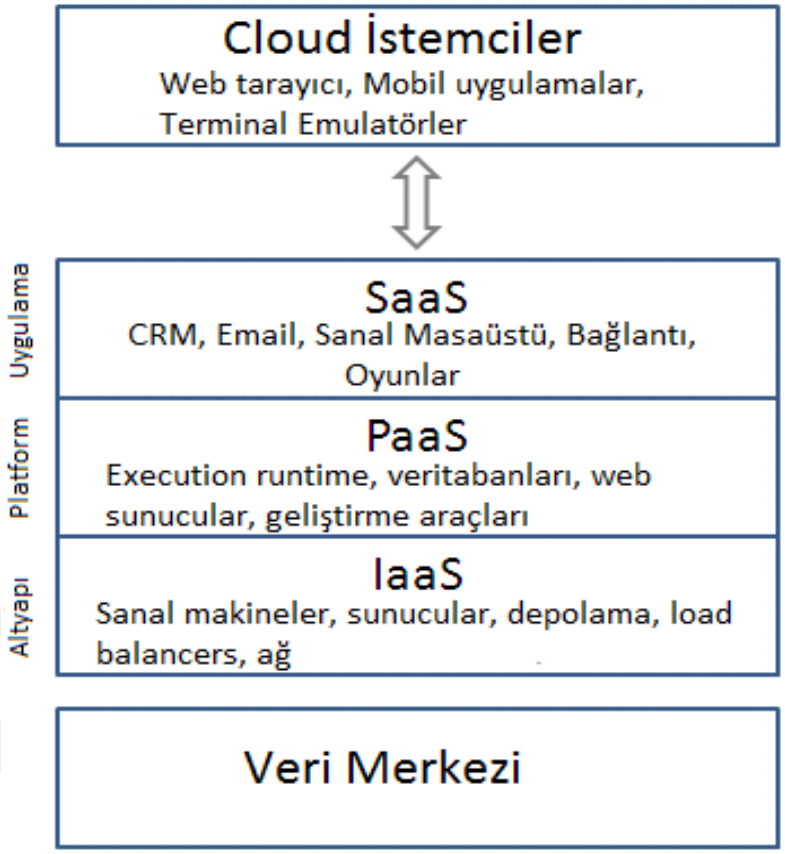
Bulut yapılarının diğer bir avantajı ise hızlı büyüyebilmesidir. Bulut sistemlerini kullanarak kullanıcıların ilgili oldukları uygulamaları çalıştırabilmeleri için kendi sahip oldukları kaynakları geliştirme ya da yeni kaynaklar satın alma zorunluluğu giderilmiş olur. Donanım, yazılım ya da bakımla ilgili tüm gerekli işlemler servis sağlayıcı tarafından halledilir. Bu sayede kullanıcıların odak noktası sadece geliştirmekte oldukları uygulamalardır.

2.2.1 Mimari Yapı

Bulut teknoloji ve servisleri bir dizi katmanlı yapıyla sınıflandırılır [25] (Şekil 2.2).

Veri merkezi katmanı bilgisayarların küme yapıları, ağ altyapısı, işletim sistemleri ve sanallaştırma teknolojileri gibi bulut teknolojilerinin üzerinde bulunan donanım ve yazılım yığınlarından oluşmaktadır.

Veri merkezlerinden sonra ilk olarak altyapı (IaaS-Infrastructure as a Service) katmanı bulunmaktadır. IaaS katmanı sırası ile dağıtık depolama sistemleri ve sanallaştırma ortamlarının bir formu olan ham hesaplama ve depolama çözümlerinin yapılmakta olduğu yerdir. Bulut kullanıcıları bir sunucuyu, yazılımı, diski ya da ağ donanımlarını kiralamak yerine sanal bir makine imajını kendi çalışma şekillerine göre uyarlayıp depolama servislerini kullanarak resim ve uygulama verileri sanal ortama yüklendikten sonra birden fazla sanal iş parçaları ile uygulamalarını çalıştırırlar. Maliyetin hesaplanma işlemi ise kullanılan ham kaynak, bant genişliği depolama space-hour ve talep edilen toplam CPU çevrimleri hesap edilerek yapılır. Bu yapıya örnek popüler bulut servis sağlayıcısı olan Amazon'un sunduğu Amazon EC2'dir [26].



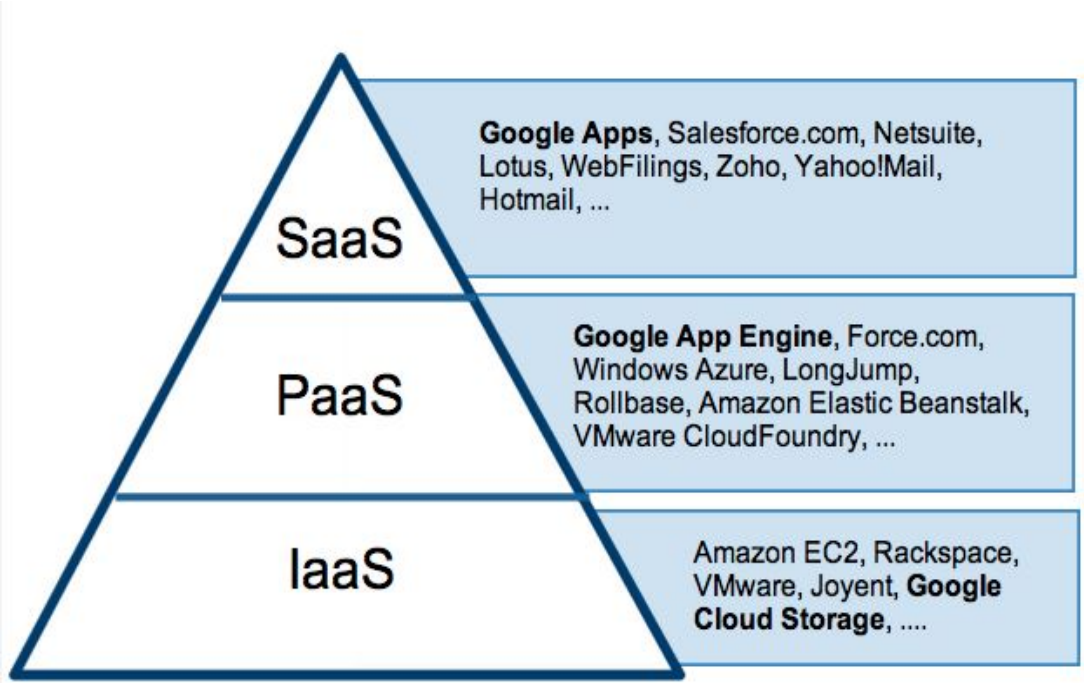
Şekil 2.2 Cloud Katmanları

IaaS katmanından sonra Platform katmanı (PaaS- Platform as a Service) bulunmaktadır. Hiyerarşide yukarı çıktığımızda PaaS' ın daha yüksek seviye programlama ve uygulama çalıştırma ortamları sağlamak için IaaS' in üzerine kurulduğunu görürüz. Bu katmandaki servislerin amacı kullanıcıyı endüstriyel standartlara sahip uygulamalarda yönetim ve yapılandırma işlemleriyle meşgul olmasından kurtarmaktır. Örnek olarak IaaS sisteminde inşa edilen dağıtık uygulamaların üzerinde bulunan Hadoop verilebilir [27].

Yazılım katmanı (SaaS-Software as a Service) Bulut yapısında en üst katmandır. Bu katman kullanıcılarına internet üzerinden son kullanıcı uygulamalarını direk olarak bilgisayarlarına indirip istifade etmelerini sağlar. Bu sayede kullanıcıların uygulamanın güncelleme ya da program yamaları gibi işlemleri ile meşgul olmasını ortadan kaldırır. Genellikle bu katmanda olan çalışmalarda tek bir tarayıcı uygulama gerekli olan tüm etkileşimler için yeterli olmaktadır. SaaS giderek yaygınlaşmakta olup Google gibi

endüstri devleri tüm kullanıcı uygulamalarını yükleme ihtiyacını ortadan kaldıracak hafif sıklet işletim sistemlerinin savunucusudur.

Bulut yapısının mimari katmanları ve bu katmanlarda çalışmalar yapan önemli teknoloji firmaları ise (Şekil 2.3) 'te belirtilmiştir.



Kaynak: Gartner AADI Summit

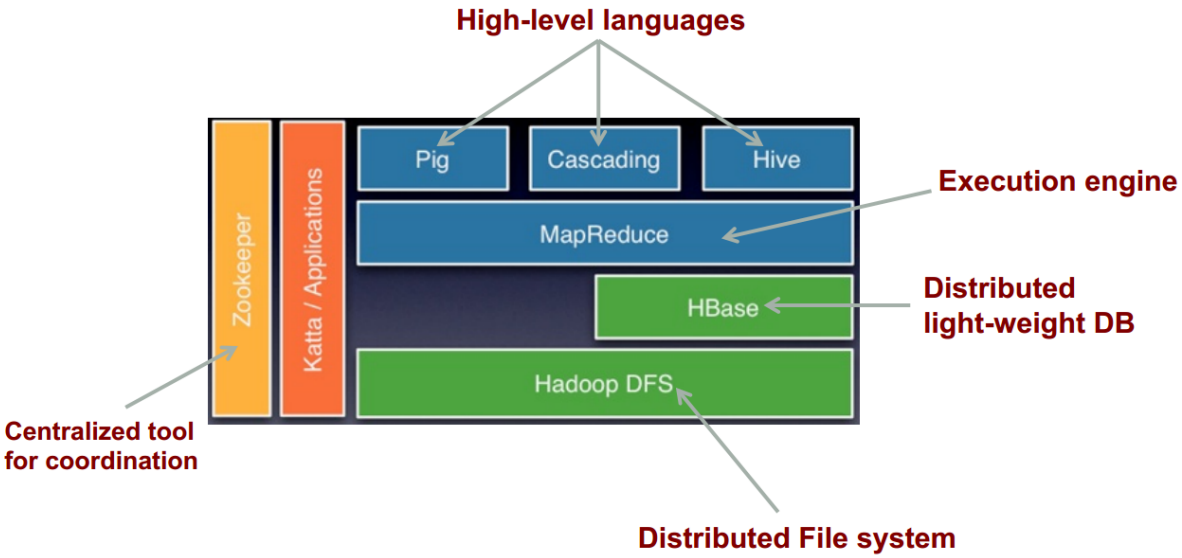
Şekil 2.3 Bulut yapıda çalışmalar yapan firmalar

2.3 Hadoop

Hadoop, genel amaçlı ve küme haline getirilmiş sunuculardan büyük ölçekli verileri işlemek, aramak amacıyla -HDFS (Hadoop Distributed File System) olarak isimlendirilen dağıtık dosya sistemiyle MapReduce yazılımının özelliklerinin bir araya getirilerek-oluşturulmuş bir altyapı kütüphanesidir. Tekil sunucularda veya kendine ait işlemci ve hafıza birimleriyle çok sayıda sunucuyu barındırabilen bir küme üzerinde de çalışabilecek esnek bir yapıya sahiptir. Java programlama diliyle geliştirilmiş açık kaynak kodlu bir altyapıdır. Apache projesi olarak ele alınmış ve özellikle ölçeklenebilir dağıtık

yapı işlemleri için geliştirilmiştir. Hadoop genel mimarisi Şekil 2.4'te gösterilmiştir. Bu kapsamda, Hadoop'un üç temel yazılım bileşeninden oluştuğu söylenebilir:

1. Hadoop Common
2. HDFS
3. MapReduce

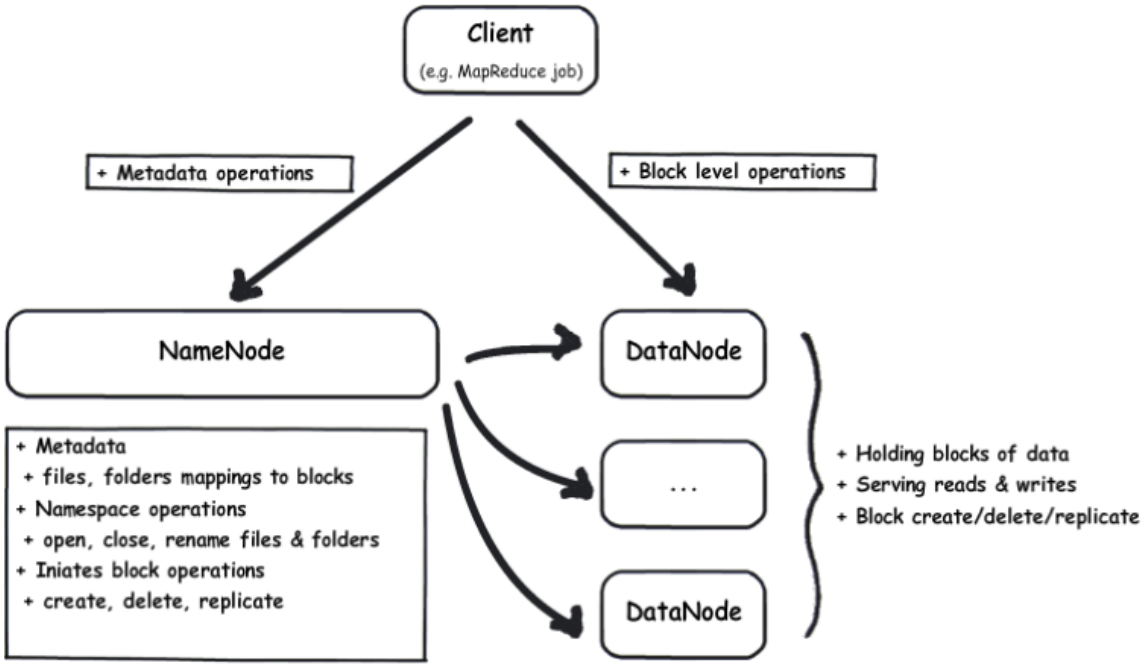


Şekil 2.4 Hadoop Mimarisi

HDFS (Hadoop Distributed File System) Hadoop için dağıtık yapıyı sağlayan dosya sistemidir. HDFS kullanılarak genel amaçlı sunucuların diskleri bir araya getirilerek büyük, tek bir sanal disk gibi davranması sağlanmaktadır. Büyük boyuttaki dosyalar bu sistem içinde depolanabilir. Dosyalar 64 Mb veya 128 Mb boyutlara sahip bloklar halinde birden çok farklı sunucu üzerine dağıtılarak yazılmaktadır. Bu yönüyle RAID yapısına benzemekte ve veri kaybı riskini en aza indirgemektedir. HDFS üzerindeki blokların üç kopyası varsayılan olarak saklanmaktadır. Her veri bloğu farklı düğümlere dağıtılır. HDFS'i iki temel süreçte inceleyebiliriz: NameNode Süreci ve DataNode Süreci.

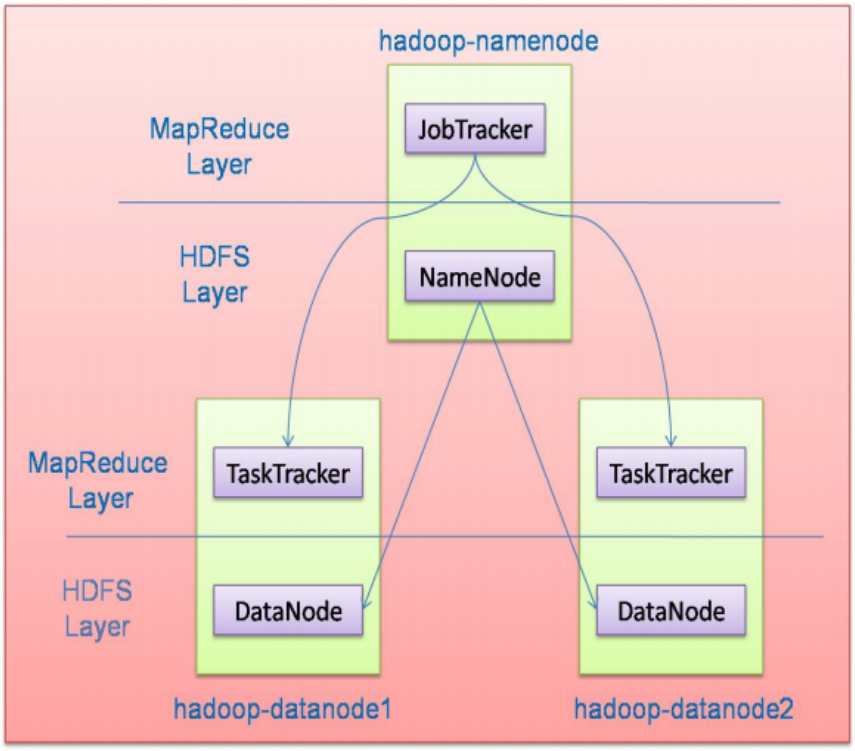
NameNode süreci bu sistemde ana süreç olarak tanımlanabilir. Veri bloklarının ilgili sunucularda dağıtılma, açılma, kapanma, isimlendirilme, oluşturulma, silinme, yeniden oluşturulma (sorun durumunda) ve dosya erişimine dönük işlemleri kapsamaktadır. Bu süreçte verilerden ziyade metadata (veri hakkında veri) yer almaktadır.

DataNode sürecinde ise temel amaç veri bloklarını saklamaktır. Bu süreçte istemciler tarafından talep edilen ve NameNode sürecine giren işlemleri yerine getirmekle sorumludur. Gerçek veriler DataNode'da tutulur. Şekil 2.5'te NameNode ve DataNode süreçlerinin özellikleri verilmiştir.



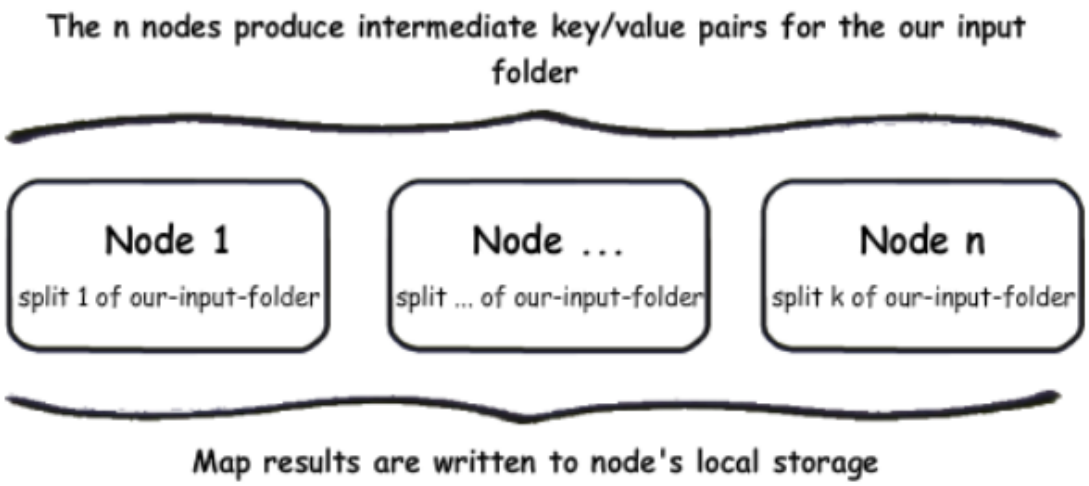
Şekil 2.5 DataNode NameNode İş Süreci

Hadoop kapsamındaki diğer ana bileşen olan MapReduce ise bir programlama modelidir. Google firması tarafından geliştirilmiş olup büyük ölçekli veriler üzerinde hızlı işlem yapabilmek için tasarlanmıştır. Veriler çok sayıda bilgisayar üzerinde dağıtılarak MapReduce teknolojisi kullanılarak iş dağılımı yapılmaktadır. Hadoop teknolojisi geliştirilirken bu yapı esas alınmıştır. Şekil 2.6'da MapReduce teknolojisinin genel mimarisi gösterilmiştir.

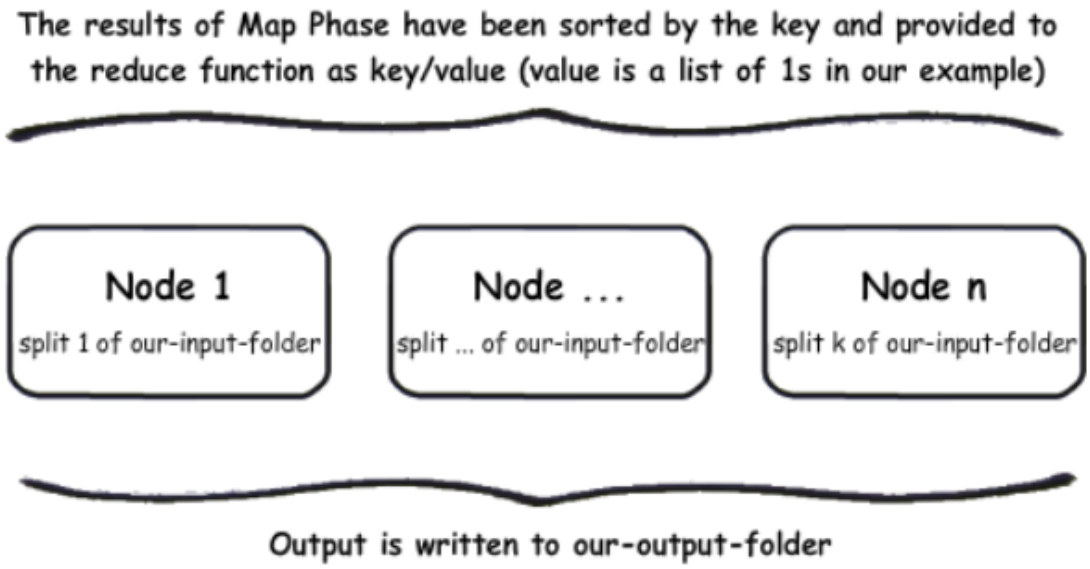


Şekil 2.6 Mapreduce Mimarisi

MapReduce teknolojisi temelde veri filtreleme amaçlı olarak Map işlevi, filtrelenen verilerden indirgenmiş sonuç alınması için de Reduce işlevlerini kullanmakta ve Hadoop Commons'a tümleşik olarak çalışmaktadır. Map ve Reduce işlevlerine ait sürece ilişkin görseller Şekil 2.7 ve Şekil 2.8'de verilmiştir.

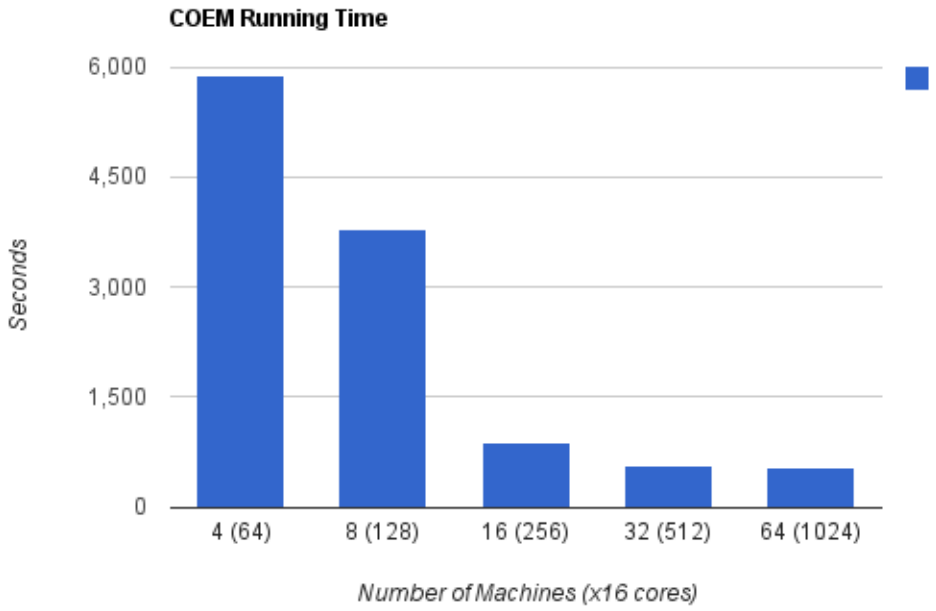


Şekil 2.7 Map İşlevine Ait Süreç



Şekil 2.8 Reduce İşlevine Ait Süreç

Hadoop, ağ trafiğine girmeden yerel diskte yer alan düğümden okuyarak, hızlı veri erişimi sağlamaktadır. Ayrıca birden fazla işlemi eş zamanlı olarak ele almakta ve doğrusal bir ölçeklendirme yapabilmektedir. Şekil 2.9'da da görüleceği üzere Hadoop üzerindeki node sayısı arttıkça performansın da arttığı söylenebilir.



Şekil 2.9 Hadoop Performans Tablosu

Hadoop hali hazırda büyük firmalar tarafından yoğun olarak kullanılmaktadır. Örnek olarak; eBay, Facebook, Linkedin, Yahoo, Amazon, Twitter şirketleri verilebilir. Ek olarak, büyük ölçekli verilerin işlenebilmesi için MongoDB, Neo4j, iş zekâsı sistemleri (Hive), yapay öğrenme sistemleri (Mahout), dağıtık uygulamalara yönelik olarak geliştirilen altyapılar (Zookeeper) gibi mevcut projeler için de kütüphanesinden faydalanılmaktadır. Sektörde aşağıda verilen alanlar için de kullanılmaya yoğun şekilde başlanmıştır:

- Risk Analizi
- Sahtekârlık ve Güvenlik Analizi
- Sağlık Hizmetleri
- Öneri Motoru
- Reklam Analizleri
- Olay/Süreç Analizi
- Kredilendirme Analizi
- Tedarik Zinciri Analizi
- Gelir ve Fiyat Dengeleme Analizleri

2.4 Azure

Windows Azure, Microsoft firmasının sunmuş olduğu bir bulut servis platformudur. Windows Azure'u, bulut yapılar altında incelemek mümkündür. Microsoft Online Services hizmetinin ardından kullanıma sunulan Azure Microsoft firmasının bulut bilgi işlemede kullandığı bir platform olarak nitelenebilir.

Bulut bilişime ve yapılarına ait bilgiler önceki bölümde detaylı olarak verilmişti. Bu bölümde bulut bilişim türleri kısaca tanıtıldıktan sonra Windows Azure platformunun bu genel tanımdaki yeri ve özellikleri verilecektir.

Bulut bilişimi Özel (private), Halka açık (public), Müşterek (community) ve Karma (hybrid) olmak üzere dört temel başlıkta incelemek mümkündür:

1. **Private Cloud:**

İlgili organizasyon/şirket bünyesinde oluşturulan sunucular ile verilen bulut hizmeti olarak tanımlanabilir.

2. **Public Cloud:**

İnternet üzerinde barındırılan sunucular ile verilen cloud hizmetidir.

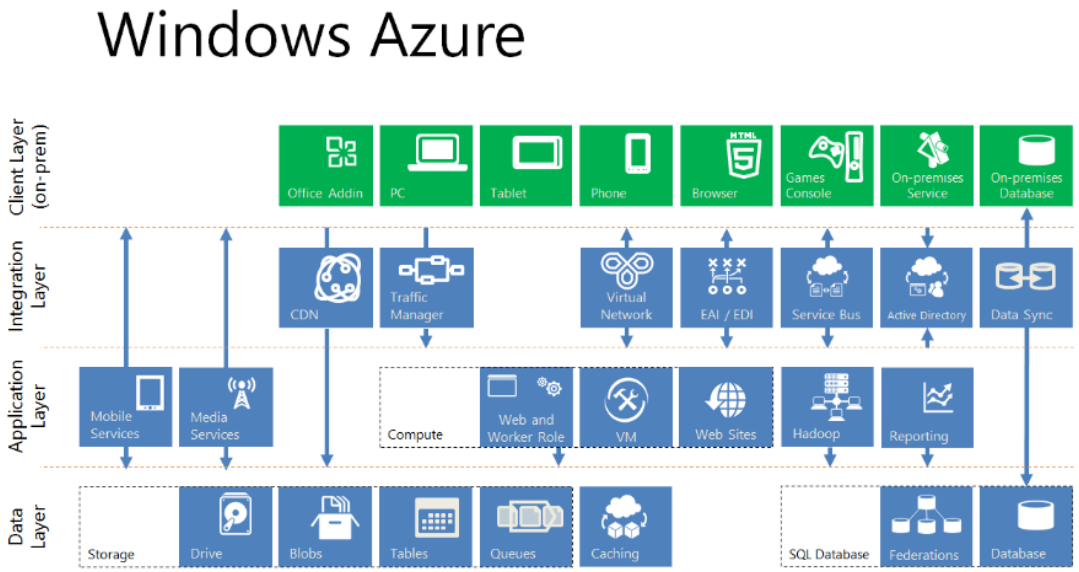
3. **Community Cloud :**

Bulut bilişim altyapısı belirli bir kurum ya da organizasyonla ortak hareket eden ya da benzerlik gösteren kuruluşlarla paylaşılmaktadır. İlgili topluluk üyeleri hizmeti verilen uygulama ve verilere erişim hakkına sahiptir.

4. **Hybrid Cloud :**

Windows Azure ile birlikte ortaya çıkan bir bulut bilişim türüdür. Burada private cloud ile public cloud türleri aradaki "site to site" vpn teknolojsi ile birlikte kullanılabilmektedir.

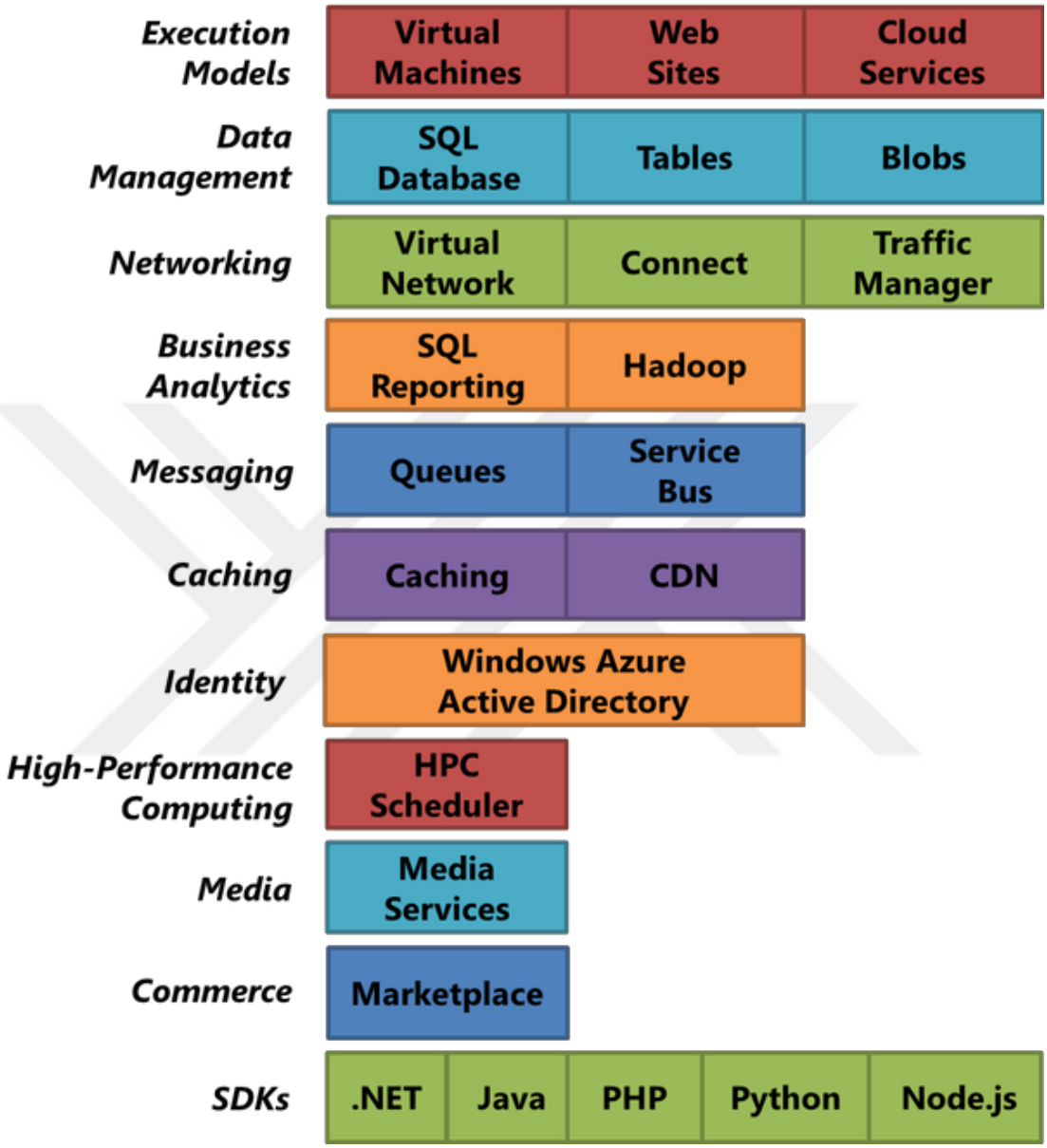
Windows Azure platformunun bileşenleri ve sunduğu hizmetlerin özetlendiği görsel Şekil 2.10'da verilmiştir.



Şekil 2.10 Azure Platform

Windows Azure kullanılarak Şekil 2.11'deki görselde belirtilen tüm hizmetler alınabilmektedir. Bu bağlamda Windows Azure platformunun üç temel hizmeti

sağladığı görülmektedir. (1) Hesaplama, (2) veri hizmetleri ve (3) uygulama hizmetleri. Bunlar kısaca takip eden bölümlerde açıklanmaktadır.



Şekil 2.11 Azure iş süreçleri

Windows Azure aylık %99,95 oranında bir hizmet seviye anlaşması SLA (Service-level agreement) sunabilmekte ve ilgili organizasyonların altyapıya odaklanmadan erişilebilir uygulamalar geliştirip test edilmesini sağlayabildiğini ifade etmektedir. Geliştirilen uygulamaların esnek olarak sürüm yükseltmesine olanak tanıyabilmekte, işletim

sistemi güncelleme, hizmet düzeltme ek geliştirmelere sağlayacak bir dağıtım modeli sunmaktadır.

Iaas, PaaS ve SaaS hizmetlerini sunabilen Windows Azure, ilgili organizasyona yönelik uygulama geliştirmek için herhangi bir dili, çerçeveyi veya aracı kullanabilecek ortam sunabilmektedir. Bu ortam sunulurken REST protokolleri kullanılmaktadır. Windows Azure bulut sisteminde birden fazla programlama diline ve altyapısına destek sunabilmekte ve açık kod lisansı kapsamında ele alınmakta ve GitHub'da barındırılmaktadır.

Uygulamaların ölçeklendirilmesi ve kaynak kullanımı konusunda tam otomatik bir self hizmet sunan Windows Azure platformu ihtiyaçlara bağlı olarak esnek bir şekilde çözümler üretebilmektedir.

Veritabanı süreçleri için de benzer şekilde ilişkisel SQL tabloları ve veritabanlarını, NoSQL tablo depolarını ve yapılandırılmamış blob depolarını kullanarak verileri saklayabilir. Ayrıca, isteğe bağlı olarak verileri analiz edip raporlama yapabilmek için iş zekâsı ürünleri kullanılabilir. Verileri aramak için de Hadoop altyapısı sisteme dâhil edilebilir [28]. Genel olarak, Windows Azure bulut yapısı istenilen programlama dili altyapısına destek olabilen, veritabanı süreçleri yönetebilen, kaynakların ölçeklendirilmesi ve sürüm güncellemeleri için uygun bir platform sunduğu söylenebilir.

2.5 Geniş Ölçekli Sistemlerde Yük Dengeleme ve Performans

Verilerin Geniş Ölçekli Dağıtık sistemlere paylaştırılması sırasında iyi seçilmiş bir parçalama sütunu (Sharding Column) performans ve verilerin erişimi açısından hayati öneme sahiptir [29]. Parçalama sütunu seçilmesi verinin karakteristiğine bağlı bir seçimdir ve yükün düğümlere dağılımını belirlemede hayati bir argümandır.

Öte yandan, düğümlere dağıtılmış yüklerin düğüm içerisindeki kaynakları etkin kullanabilmesi de performans için temel tasarım yaklaşımlarındandır [30].

Thread yönetimi temelde 1960'lı yıllardan bu yana endüstride kullanılagelen ve Little Kanunu olarak bilinen Bağıntı (1)'deki formüle dayanır.

$$L = \lambda.W \qquad (2.1)$$

Bu bağıntıda, L, sistemde bulunan anlık ortalama müşteri sayısını belirler. λ birim zamanda hizmet almak için gelen müşteri sayısını gösterirken W ise bir müşterinin ortalama hizmet alma süresidir.

Welsh de tersinden bir yaklaşımla [31], iş parçacıklarına kaynak ayırmak için her bir işin bekleme süresine bağlı bir formül kullanmaktadır. Bahsi geçen çalışmada, kuyrukta bekleyen iş sayısından SEDA sisteminin her bir aşamasındaki iş yükü hakkında kestirimde bulunmaktadır. Kuyruktaki iş sayısı arttığında kuyruğun önceden tanımlı eşik değerine bağlı dinamik olarak daha fazla thread çalıştırmakta, kuyruk boşaldığında da bu thread'leri geri düşürmektedir.

Bu yaklaşımlar, kuyruk için eşik değerlerin nasıl ayarlanacağına dair bir tanımda bulunmaz, ayrıca thread ekleme ve çıkarma işlemleri sırasında dalgalanmalar oluşmakta ve bu dalgalanmalar az bir miktarda bile olsa ek bekletme sürelerine ihtiyaç duymaktadır. Örneğin adı geçen çalışmada, 2 worker thread'den 3'e geçtildiğinde bekleme gecikmeleri %35'in üzerine çıkarak dramatik bir şekilde arttığından söz edilmiştir.

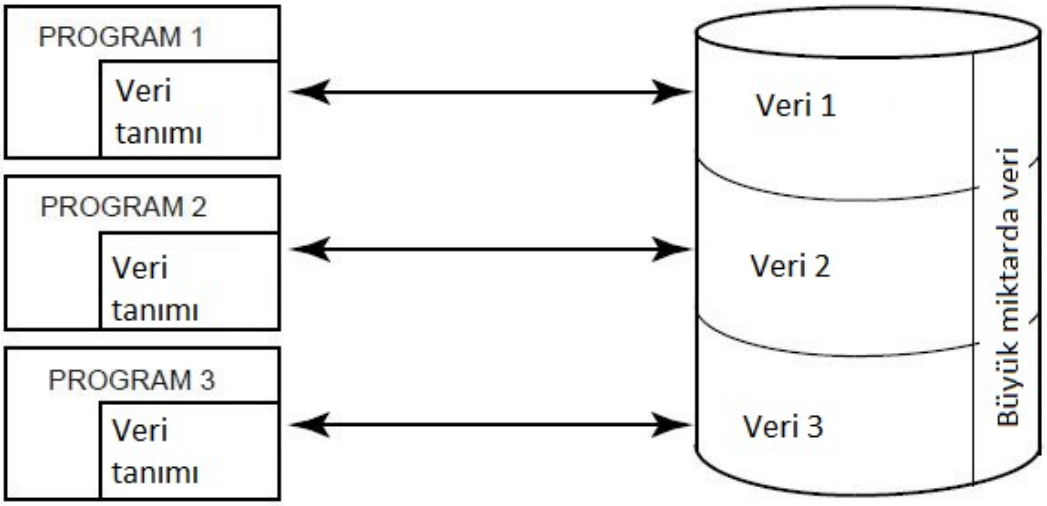
Çalışmanın bir devamı olan başka bir çalışmada [32] gecikme optimizasyonlu thread sayısının peryodik olarak yeniden hesaplanmakta ve bu anlık olarak sistem yükünün bir göstergesi gibi davranmaktadır. Bu yöntem ise sistem yükünün anlık değerlerine doğrudan odaklanırken, yükün yönü hakkındaki davranışları gözardı etmektedir. Örneğin, anlık bir yük artışına sistem agresif bir tepki gösterebilmektedir.

Bu çalışmada bu iki yaklaşımdan farklı olarak, kuyrukta bekleyen iş sayısı ve thread sayısı arasında anlık değerler ve geçmişten gelen değerler birlikte kullanılarak ortalama değer hesaplanmış ve ayrıca doğrudan yeni bir thread sayısına sıçramak yerine göreceli bir yönelim fonksiyonundan yararlanılarak olağanüstü dalgalanmaların önüne geçilmiştir.

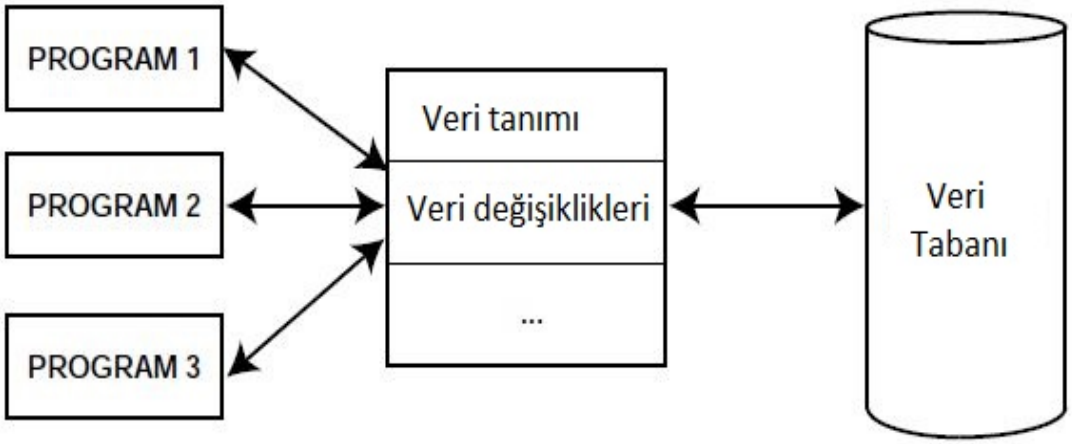
BÖLÜM 3

DAĞITIK VERİ TABANI SİSTEMLERİ

Dağıtık veritabanı sistemleri, veritabanı sistemleri ve bilgisayar ağları yaklaşımlarının birleşmesiyle oluşan bir teknolojidir. Veritabanı sistemleri bizi her bir uygulamanın kendi ile ilişkili olan veriye ulaştığı veri işleme paradigmasından (Şekil 3.1) tüm verinin tek bir merkezden yönetildiği ve tanımlandığı (Şekil 3.2) paradigmaya götürmüştür.



Şekil 3.1 Geleneksel veri işleme



Şekil 3.2 Veritabanı işlemleri

Bu yeni yaklaşımda veri bağımsızdır ve uygulamalar veri organizasyonundaki fiziksel ve mantıksal tüm değişikliklere bağımlıdır. Bu yeni tasarımın ortaya çıkış nedenlerinden en öne çıkanı operasyonel verilerin ticari ve geliştirme yönlerinden tek bir merkezde toplanması isteğidir.

İki farklı teknoloji ve sistemin güçlü ve tek bir sistem oluşturmak için nasıl bir araya geldiğini anlamak oldukça zor olsa da veritabanı teknolojilerindeki en önemli fikir ya da nesne "merkezileştirme" değil tümleştirmedir [33]. Burada dikkat edilmesi gereken nokta bu iki terimin birbirlerini ima etmediğini anlamadır. Yani merkezileştirme olmadan tümleştirme yapılabilmektedir ve bu da dağıtık veritabanı teknolojilerinin amacıdır.

Dağıtık veritabanı sistemleri bilgisayar ağları üzerinde çeşitli mantıksal yöntemlerle bir araya getirilmiş dağıtık veritabanları şeklinde tanımlanabilmektedir [34]. Dağıtık veritabanı sistemleri bir bilgisayar ağında her biri farklı düğümlerde depolanan veriler topluluğu değildir. Dağıtık veritabanı sistemlerinde mantıksal ilişkilendirmenin yanında veriler arasında yaygın bir arayüz aracılığı ile oluşturulmuş bir yapı olması gerekmektedir. Bazı bakış açıları fiziksel dağılımı gerekli görmeyip dağıtık veritabanı sistemlerini aynı bilgisayar sistemi içinde yer alan farklı veritabanları gibi düşünse de verilerin fiziksel yönden dağılması birçok sorunla karşılaşılmasına neden olmuştur. Fiziksel dağılımdan kasıt tamamıyla farklı coğrafik noktalar olmayıp aynı ortamda yer

alan birbirlerine bilgisayar ağlarıyla bağlı fakat disk, bellek ve işlemci gibi üniteleri paylaşmayan yapılar olmasıdır.

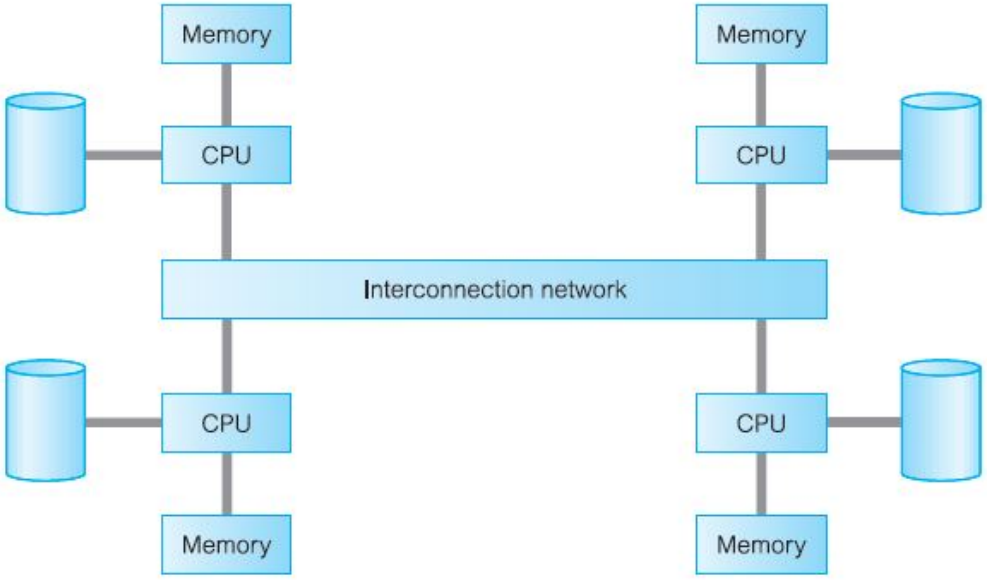
Dağıtık veritabanı sistemlerinde dikkat edilmesi gereken diğer bir nokta ise çok işlemcili sistemler ile dağıtık sistemlerin karıştırılmamasıdır. Çok işlemcili sistemler simetrik özdeş işlemci ve bellek ünitelerinden kurulu her bir işlemci parçacığının alacağı görevlerin aynı işletim sistemi tarafından belirlendiği sistemlerdir [33]. Fakat bu durum dağıtık veritabanı sistemlerinde böyle değildir. Hem işletim sistemi hem de işlemci ve disk gibi üniteler heterojen şekilde dağılmıştır. Ayrıca literatürde çok işlemcili sistemler üzerinde çalışmakta olan veritabanları paralel veritabanı sistemleri olarak adlandırılmaktadır.

3.1. Mimari Yapı

Dağıtık veritabanı sistemlerinde mimari yapı genel anlamda bilgisayarlar ya da sistemler arası kurulan ağ mimarisinden farklı olmayıp farklı bir ağda bulunan veri ve programların birbirleri ile iletişim kurması mantığı ile tanımlanabilir. Temel anlamda ise maliyet performans avantajlarına göre farklı mimarilerin olması sağlanmıştır. Bellek ve disk gibi temel ünitelerin paylaşılmasındaki farklılığa göre dağıtık veritabanı sistemlerinde 3 farklı mimari yaklaşım mevcuttur.

3.1.1. Paylaşımsız Mimari

Paylaşımsız mimari paralel dağıtık veritabanı sistemi olarak tanımlanabilir. Bu yapıda her bir çalışma parçacığı (process) ana belleğe ve disk ünitelerine özel erişim hakkına sahiptir [35] (Şekil 3.3). Bu tip mimarı yapıda işlemciler dağıtık yapıda bulunan diğer işlemcilere bağlıdır. Her sunucunun kendi diskleri mevcuttur. Düğümlerde bulunan diskler hiçbir şekilde paylaşılmaz ve diğer işlemciler tarafından kullanılmaz. Her işlemci ve bellek farklı işletim sistemlerinin kontrolündedir.



Şekil 3.3 Paylaşımsız Mimari

Paylaşımsız mimaride hızlı veri yollarının kullanılmasıyla çok geniş yapıda dağıtık sistemler elde edilebilir. İşlemciler ya da sunucular arası bağlantının diskler ya da bellekler arası bağlantıdan daha düşük maliyetli olması sayesinde paylaşımsız mimari daha düşük maliyetli olabilmektedir. Ayrıca maliyetin düşük olması ve düğümlerin bağımsız olması nedeniyle daha kolay genişleyebilir bir mimaridir. Bu avantajlara karşın sistemin hızlı ve geniş ölçekli büyümesi sistemin yönetimi ve yük dengelemeyi zorlaştırmaktadır.

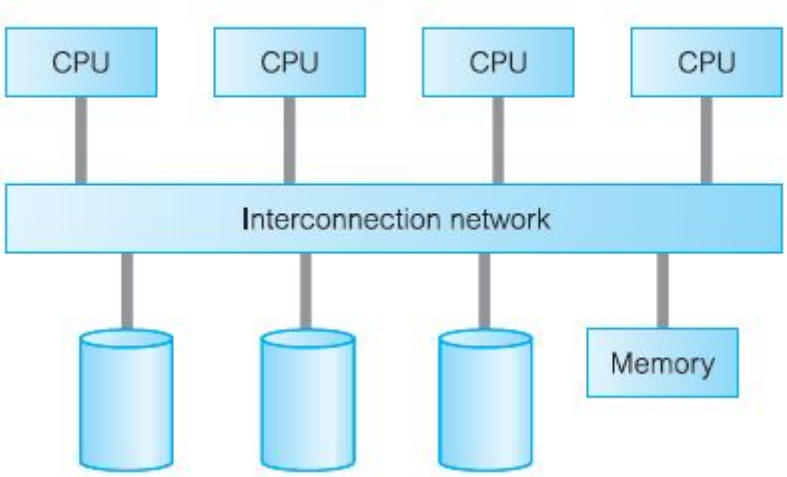
Paylaşılmayan, sıklıkla yoğun-paralel işlemler (MPP) olarak bilinen, her bir işlemcinin kendi hafıza ve disk alanı ile tüm sistemin bir parçası olduğu çoğul işlemci mimarisidir. Veritabanı, tüm diskler içinde parçalara ayrılmış, her bir sistemin veritabanı ile ilişkili ve veri tüm sistem üzerinde tüm kullanıcılara şeffaf bir şekilde açıktır. İstenen veri lokal olarak saklanıyorsa, performans optimumdur.

Paylaşılmayan tanımı dağıtık sistemleri zaman zaman kapsıyor olsa da paralel DBMS'lerdeki verinin dağılımı sadece performans sebeplerine bağlıdır. Paralel DBMS'nin düğümleri tipik olarak aynı bilgisayarda ya da sunucuda olmasına rağmen,

Dağıtık DBMS'nin düğümleri coğrafi olarak dağılımlıdır, ayrı ayrı yönetilir ve daha yavaş bir ara bağlantı ağına sahiptir. Paralel teknoloji tipik olarak muhtemelen terabaytlarla ölçülen çok büyük veri tabanları için kullanılır ya da her saniye binlerce işlemi gerçekleştirmek zorunda kalan sistemler için kullanılır. Bu sistemler çok büyük ölçekli verilere erişirler ve sorgulamalara zamanlı tepkiler sağlamalıdırlar. Paralel tarama kullanan karmaşık işlem performansını geliştirmek için bir paralel DBMS alt mimari kullanabilir ve işlem yükünü paylaşmak amacıyla otomatik olarak çoklu işlemci noktalarına imkân sağlayan teknikleri çoğaltabilir. Bütün büyük DBMS üreticileri veri tabanı motorlarının paralel sürümlerini üretirler.

3.1.2. Bellek Paylaşımlı Mimari

Paylaşımlı mimari modelinde iş parçacıkları herhangi bir bellek ya da disk ünitesine doğrudan bağlıdır (Şekil 3.4).



Şekil 3.4 Bellek paylaşımlı mimari

Paylaşılan bellek sistem hafızasını paylaşan tekil sistem içerisindeki çoklu işlemcilerin bir araya getirildiği mimaridir. Paylaşımlı mimaride tüm iş parçacıkları tek bir işletim sistemi tarafından yönetilmektedir. Mevcut anabilgisayarların (mainframe) tasarımı ve

simetrik çok işlemcili sistemler bu yaklaşım üzerine geliştirilmiş olup XPRS, DBS3, DB2 ilk örnekleridir [33].

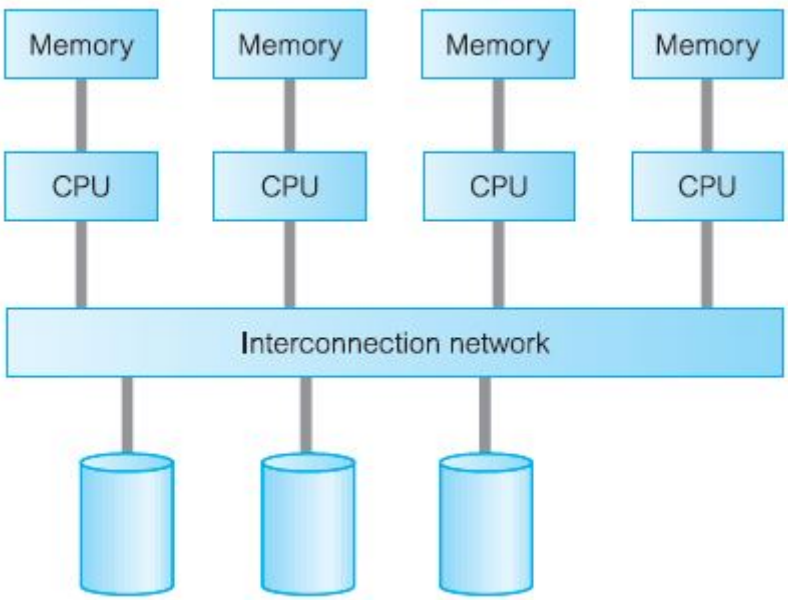
Paylaşılan bellek mimarisi ayrıca simetrik çoğul işlem (SMP) olarak da bilinir. Bu yaklaşım, birkaç mikro işlemcinin paralel desteklendiği kişisel iş istasyonlarından, büyük indirgenmiş komut takımıyla hesaplama (RISC) tabanlı makinelere kadar yol boyunca tüm büyük bilgisayarlarda, birçok platformda popüler olmuştur [34]. Bu mimari sınırlı sayıdaki işlemci için yüksek hızlı veri erişimi sağlar. Fakat bu yaklaşık 64 işlemcinin üzerinde ve bağlı ağın darboğaza girdiği zamanlarda ölçeklendirilebilir değildir [34].

Paylaşımlı mimarinin iki önemli avantajı basitlik (Simplicity) ve yük dengelemedir (Load Balancing). Dizin bilgileri ve kontrol anahtarlarının tüm iş parçacıkları tarafından paylaşılabilir olmasıyla birlikte veritabanında işlem yapma ile tek işlemcili bir bilgisayarın arasında fark kalmamıştır. Bu özellik basitlik avantajına örnek olmaktayken herhangi bir sorgu ya da işlemin paralel şekilde yürütülebilmesi de yük dengeleme avantajına örnek olarak verilebilmektedir.

Paylaşımlı mimarinin dezavantajlarını ise sistemde sayısı artmakta olan işlemcilerin bellek birimine ayrı ayrı bağlantı kurması ve işlemci ile bellek arasındaki bu bağlantıyı sağlayacak olan donanımın daha karmaşık yapıda olması sebebiyle yükselen maliyet, artan işlemci sayısı ile bellekte kullanılabilir alanın azalması olarak görülebilir.

3.1.3. Disk Paylaşımlı Mimari

Paylaşılan disk, doğal olarak merkezileşmiş ve yüksek geçerlilik ve performans gerektiren uygulamalar için optimize edilmiş mimaridir. Disk paylaşımlı mimaride işlemciler tüm disklerle ayrı ayrı bağlantı kurabilirken sistemde bulanan tüm belleklere erişimleri yoktur (Şekil 3.5). Bu yapıda işlemciler disklerde bulunan bilgileri kendi gruplarında olan belleğe aktarırlar. Disk birimlerinin ortak kullanıldığı bu mimarinin oluşmasıyla birlikte dağıtık sistemlerde veri kilit yönetimi (Distributed Lock Manager) tekniğinin gelişmesini sağlamıştır.



Şekil 3.5 Disk paylaşımlı mimari

Her bir işlemci tüm disklere direkt olarak ulaşabilir fakat her birinin kendi özel hafızası vardır. Paylaşılmayan mimari gibi, paylaşılan disk mimarisi de paylaşılan hafıza performansındaki darboğazlarını ortadan kaldırır.

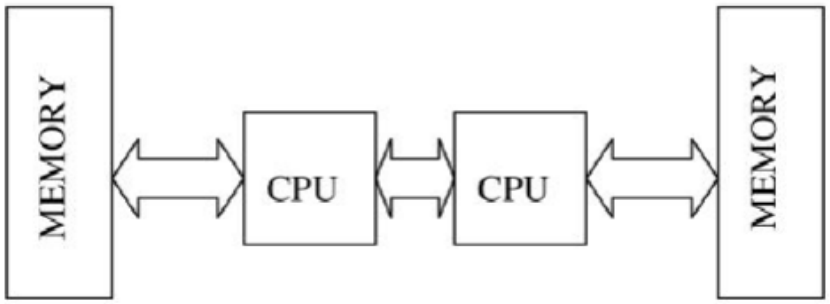
Disk paylaşımlı mimarinin birçok avantajı bulunmaktadır. Bunların başlıcaları düşük maliyet, kolay genişleyebilirlik (extensibility), yük dengeleme (Load balancing) ve devamlılıktır (availability). Zaman içerisinde bilgisayarlarda donanımlar arası iletişimi sağlayan veri yollarının (BUS) gelişmesi ve maliyetlerinin düşmesi ile birlikte disk paylaşımlı mimari bellek paylaşımlı mimariye göre daha düşük maliyette tasarlanması sağlanmıştır. Her işlemciye yeterli miktarda bellek verilmesi nedeniyle mimarinin daha kolay şekilde genişlemesi sağlanmıştır. Belleklerin işlemciler için özel olması bellekte meydana gelecek hatalardan ya da sorunlardan diğer düğümlerde bulunan işlemcilerin bu hatalardan meydana gelecek sorunlardan etkilenmemesi sayesinde sistemin devamlılığı sağlanmıştır.

Disk paylaşımlı mimarinin avantajlarının yanında dikkat edilmesi gereken önemli bir dezavantajı ise performanstır. Disk paylaşımlı mimaride dağıtık veritabanı yönetim protokollerinden olan dağıtık kilitleme (Distributed locking) ve iki taraflı onaylama (Two

phase commit) özelliklerinin kullanılması gerekmektedir (Özsu & Valduriez, 2011). Bu gereklilik de çeşitli performans sorunlarına neden olabilmektedir.

3.1.4. Hibrit Mimari

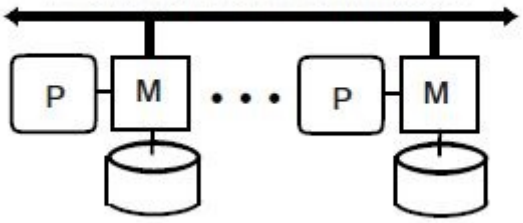
Hibrit mimari dağıtık veri tabanları için tasarlanan 3 farklı mimarinin avantajlarını kullanarak elde edilmek istenen bir mimaridir. Bu mimarilerden başlıcası Düzensiz Bellek Erişimi (NUMA)'dir [36]. NUMA'nın tasarımı bellek erişim zamanlarının belleğin işlemci ile bağlantısıyla ölçüldüğü bir mimaridir. NUMA'nın temel amacı kilit çakışmalarını azaltmak ve çalışan iş parçalıklarına ayrılan bellek miktarını maksimuma çıkarmaktır [36]. Simetrik çok işlemcili (SMP) mimarilerde bellek ve işlemci trafiğinin tek bir yere gitmesi sistemin verimli şekilde genişlemesine imkân vermiyordu [37]. NUMA paylaşımlı bellek mimarisiyle gelen avantajların yanında genişleyebilen dağıtık bellek mimarisi ile gerekli olan bu genişlemeyi etkin bir biçimde sağlamıştır (Şekil 3.6).



Şekil 3.6 NUMA mimarisi

NUMA'nın mimarisinde her bir işlemciye ayrı bir bellek düşmektedir. Bu sayede bellek çakışmaları önlenmiştir.

NUMA modelleri arasında en öne çıkanı Önbellek tutarlı düzensiz bellek erişimidir (CC-NUMA) [38]. Bu mimaride ana bellek düğümler boyunca paylaşımsız mimari ya da disk paylaşımlı mimaride olduğu gibi fiziksel olarak dağıtılmıştır (Şekil 3.7). Buna rağmen iş parçacıkları diğer tüm iş parçacıklarının bellek parçalarına erişebilmektedir.

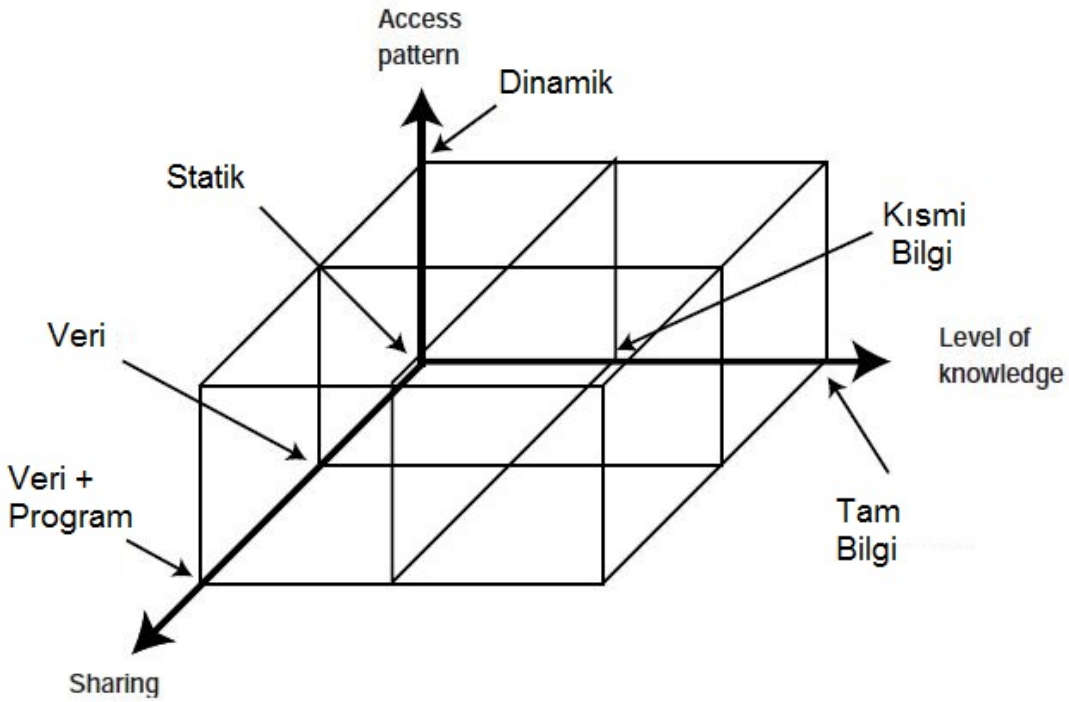


Şekil 3.7 Önbellek öncelikli düzensiz bellek erişimi

3.2. Dağıtık Veri Tabanı Tasarımı

Dağıtık veri tabanı sistemlerinde tasarım konusunda iki önemli unsur bulunmaktadır. Bunlardan ilki dağıtık veritabanı sistemlerindeki yönetim yazılımı diğeri ise dağıtık sistemde çalışmakta olan uygulama programlarıdır.

Dağıtık sistemler organizasyonu ilk defa 3 boyutlu ortogonal bir yapı şeklinde incelenmiştir [39] (Şekil 3.8).



Şekil 3.8 Dağıtık Sistemler Organizasyonu

Dağıtık sistemlerin organizasyon yapısının incelendiği bu 3 boyutlu sistemde üç farklı konu ele alınmıştır. Bunlardan ilki paylaşım derecesidir (Sharing). Paylaşım derecesi boyutunda 3 farklı ihtimal bulunmaktadır. Bunlardan ilki paylaşımın olmaması durumudur. Bu durumda her program kendi verisine erişmekte olup başka bir uygulama ya da programın verisine ihtiyaç duymamaktadır. Dolayısıyla paylaşım yoktur. Bu durum ağ altyapısının yeterince yaygın olmadığı dönemlerde kullanılmıştır.

Daha sonraki paylaşım durumu, verilerin paylaşıldığı durumdur. Bu durumda çalışacak tüm programlar sitede ya da uygulamada birebir şekilde kopyalanmış olarak mevcut bulunmaktadır. Fakat veriler tek bir noktadadır. Ağda önemli veriler ilgili yerlere gitmektedir.

Son paylaşım durumu ise veri ve programın birlikte paylaşıldığı durumdur. Bu durumda herhangi bir program merkezi alandaki veriyi paylaştığı gibi özel bir servis ile ağda çalışmakta olan farklı bir programdan da talepte bulunabilmektedir. Levin Morgan [39] planındaki veri paylaşımı ile hem verinin hem programın paylaşıldığı durum homojen ve heterojen dağıtık sistemler arasındaki yapıya örnek verilebilir. Bu planın açıkça bize gösterdiği şey heterojen sistemlerde bir programı farklı bir donanım ya da işletim sisteminde çalıştırmanın oldukça zor olduğudur. Bu zorluğu ancak veriyi ağda ihtiyaç olan noktaya taşıyarak ortadan kaldırabilmektedir.

Yapıdaki ikinci boyut olan erişim deseni (Access Pattern) de iki farklı ihtimal bulunmaktadır. Kullanıcı talepleri erişim deseninin statik olduğu durumda zamanla değişime uğramaz. Dinamik kullanıcı talepleri ile değişime açıktır. Açık bir şekilde anlaşılmaktadır ki statik değişkenlerin olduğu sistemlerin yönetimi ve planlaması daha kolaydır. Fakat statik değişkenler ile yürütülmekte olan sistemlerde plan ve yönetim kolay olduğu halde bu şekilde çalışan sisteme gerçek hayatta rastlamak oldukça zordur.

Organizasyon yapısının anlatıldığı şemadaki son boyut bilgi seviyesidir (Information of Knowledge). Bu boyut erişim deseninin davranış şekli ile neredeyse aynıdır. İhtimallerden biri sistemi tasarlayanların, kullanıcıların veritabanına nasıl erişeceği konusunda bilgisinin olmamasıdır. Bu seçenek teorik olarak muhtemel olsa da gerçek hayatta neredeyse imkânsızdır. Dağıtık veritabanı tasarımcıları ve yöneticileri bu zorluk ile mücadele etmektedir. Bilgi seviyesinin diğer seçeneği tam bilgiye sahip olunduğu

durumdur. Dağıtık veri tabanı tasarım problemleri bu genel çatı çerçevesinde ele alınmalıdır.

3.3. Dağıtık Veri Tabanlarında Yönetim İncelemesi

Dağıtık veri tabanları yönetim sistemleri herhangi bir yazılım sistemi ile dağıtık veri tabanlarının yönetim iznini kullanıcılara şeffaf bir biçimde veren sistemdir. Bu sayede kullanıcılar sadece yönetmekle sorumlu olduğu verileri bilmekte arka planda sistemin dağıtık olduğunu bilmemektedir.

Dağıtılmış (Distributed) bir bilgisayar ağı üzerinde fiziksel olarak gösterilmiş, paylaşılan bilginin mantıksal ilişkili olduğu koleksiyondur. Dağıtılmış (Distributed) verilerin yönetimine izin veren yazılım sistemi DBMS veritabanı ve sistemi kullanıcılara şeffaf hale getirir.

Dağıtık Veritabanı Yönetim Sistemi (DDBMS) parçalara bölünmüş tek bir mantıksal veri tabanından oluşur. Her bir parça, ayrı DBMS'lerin kontrolündeki ve iletişim ağıyla birbirine bağlanmış bir veya daha fazla bilgisayarda saklanır. Her bir düğüm kullanıcı isteklerini bağımsız bir şekilde işleme yeteneğine sahiptir. Ayrıca ağdaki diğer bilgisayarlar üzerinde saklanan verileri de işleme yeteneğine sahiptir.

Kullanıcılar dağıtılmış veri tabanına; diğer düğümlerden bilgi gerektirmeyen (yerel uygulamalar) ve diğer düğümlerden bilgi gerektiren (evrensel uygulamalar) çeşitli uygulamalar yoluyla ulaşırlar. Bu yüzden bir DDBMS şu özelliklere sahiptir:

-Mantıklı ve ilişkili paylaşılmış bir bilgi topluluğu;

-Birkaç parçaya ayrılmış veri;

-Kopya edilebilir parçalar;

-Parçalar/kopyalar alanlara ayrılmıştır;

-Alanlar iletişim ağı ile ilişkilendirilmiştir;

-Her bir düğümdeki bilgi DBMS'nin kontrolü altındadır;

-Her bir düğümdeki DBMS yerel uygulamaları bağımsız bir şekilde idare edebilir;

-Her bir DBMS en az bir evrensel uygulamaya katılır [34]. Sistemdeki her bir düğümün kendi yerel veri tabanına sahip olması gerekmemektedir

DDBMS'in tanımından, sistemin dağılımı kullanıcıya görünmez hale getirmesi beklenir. Bunun için, dağıtılmış veri tabanının farklı bilgisayarda saklanan veya kopyalanmış bölümlere ayrılması gerçeği kullanıcıdan gizli tutulmalıdır. Şeffaflık hedefi, dağıtımlı sistemi merkezi sistem gibi göstermektir. Bu DBMS'in temel prensibi olarak adlandırılır. Bu gereksinim son kullanıcıya önemli fonksiyonellik sağlar.

Çoklu işlemcilerin tekil bir veri tabanına ortak erişimlerini sağlamak için, paralel DBMS paylaşılmış kaynak yönetimini sağlamak zorundadır. Hangi kaynaklar paylaşıldığı ve bu paylaşım nasıl yapıldığı direk olarak sistem ölçeklendirilebilirliğini ve performansını etkilemektedir.

Dağıtık sistemlerin ve uygulamaların geleneksel merkezileştirilmiş veritabanı yönetimine göre avantajları ve dezavantajları vardır.

3.3.1. DDBMS'in Avantajları

3.3.1.1. Organizasyonel Yapıyı Yansıtır

Birçok organizasyon çeşitli bölgelerde doğal dağılımlıdır. Mesela firma çeşitli kentlerde birçok ofise sahiptir. Böylesi bir uygulamada kullanılan veri tabanlarının bu yerlere dağıtılmış olması doğaldır. Firmanın her ofisinde o noktada çalışan personel, kiralanacak ya da satılacak ürünler ve bu ürünleri almak ya da kiralamak isteyen ya da onlara sahip olan müşterilerle ilgili ve bu çeşit şeylerin detaylarını ihtiva eden veri tabanları tutabilir. Şubedeki personel veri tabanına yerel girişler yapacaktır. Şirket yöneticileri belirli bir sayıda ya da bütün şubelerdeki girdileri sorgulamak isteyebilirler.

3.3.1.2. Paylaşım ve Yerel Otonomi

Bir organizasyonun coğrafi dağılımı verinin dağılımına da yansıtılabilir; Bir noktadaki kullanıcılar bir başka noktadaki veriye erişebilirler. Veri, onu normal olarak kullanmak

isteyen kullanıcılara yakın bir bölgeye yerleştirilebilir. Böylelikle kullanıcılar veri üzerinde yerel kontrole sahip olabilirler ve bu verinin kullanımı ile ilgili yerel politikalar oluşturup bunları geliştirebilirler. Küresel bir veri tabanı yöneticisi (DBA) bütün sistemden sorumludur. Genel olarak, yerel DBA yerel DBMS'yi yönetebilsin diye bu sorumluluğun bir parçası yerel seviyeye devredilir.

3.3.1.3. Erişilebilirlik

Merkezi bir DBMS'de bilgisayar çökmesi DBMS'nin işlevlerini sonlandırır. Ancak bir DDBMS'nin bir noktasındaki çökme ya da bazı noktaları erişilemez hale getiren iletişim bağlantısındaki bir çökme bütün sistemi çalışamaz hale getirmez. Dağılımlı DBMS'ler bu tarz çökmelere rağmen işleme devam etmek için tasarlanmıştır. Eğer bir nokta çökerse sistem başarısız noktanın istemlerini bir başka bölgeye yönlendirebilir.

3.3.1.4. Güvenilirlik

Bir bölgeden daha fazla bölgede var olabilmesi için bir veri çoğaltılabilir. Bir noktanın çökmesi ya da iletişimin kesilmesi veriyi erişilemez hale getirmez.

3.3.1.5. Performans

Bir veri en fazla talebin olduğu yere yakın konuşlandırıldığı için ve dağılımlı DBMS'lerin daha önceki paralelliğine verildiğinden dolayı veri tabanına erişim hızı uzak merkezi veri tabanından daha hızlı olabilir. Dahası her bölge toplam veri tabanının sadece bir parçasına müdahale ettiği için işlemci açısından aynı işleme gerek duyulmayabilir.

3.3.1.6. Ekonomiklik

1960'larda işlem gücü, donanım maliyetlerinin karesine göre hesaplanırdı. Üç kat maliyet, dokuz kat güç sağlamalıydı. Bu Grosch Yasası olarak bilinirdi [34]. Ancak şimdilerde tek bir büyük bilgisayarın gücüne eşit, birçok daha küçük bilgisayardan oluşan bir sistem kurmak genellikle daha az bedele mal olur.

Bu aynı zamanda bütün sistemi güncellemektense zaten var olan bir ağa yeni iş istasyonları eklemeyi daha uygun maliyetli hale getirir. Maliyet azaltmanın diğer bir

yararı da verinin coğrafi olarak birbirine uzak olduğu durumlarda ve dağıtılmış veriye erişim gerektiren uygulamaların kullanıldığı durumlarda ortaya çıkar. Bu tür durumlarda, ağ boyunca iletilen verinin maliyetine bağlı olarak uygulamayı bölümlerine ayırmak ve işlemi her bölgede ayrı ayrı gerçekleştirmek çok daha ekonomik olabilir.

3.3.1.7. Modüler Büyüme

Dağıtık bir ortamda genişlemeye müdahale etmek çok daha kolaydır. Yeni bölgeler, diğer bölgelerin işlemlerine müdahale etmeksizin, ağa eklenebilirler. Bu esneklik bir organizasyonun göreli olarak kolaylıkla genişleyebilmesine imkân sağlar. Veri tabanı büyüklüğünü artırmak, sisteme işleme ve saklama gücü ekleyerek gerçekleştirilir. Fakat merkezi DBMS'lerde büyüme hem donanıma hem de yazılıma değişiklikler getirmeyi zorunlu kılabilir.

3.3.1.8. Entegrasyon

Dağıtık veritabanı yönetim sistemleri organizasyonların mevcut durumda kullanmakta oldukları sistemlerini yeni sistemlere tümleştirmelerinde önemli bir rol oynar. Organizasyonların iş hacimlerinin büyümesi ile birlikte yönetmeleri gereken veri miktarları da büyüyecektir. Bu da sistemlerin dağıtık yapıda olmasını zorlayacaktır.

3.3.2. DDBMS'lerin Dezavantajları

3.3.2.1. Kompleks yapı

Dağıtık veri tabanı sistemleri büyük ölçekli verilerin verimli şekilde yönetilmesi amacıyla geliştirilmiştir. Hızlı ve kolay şekilde büyüyen yapı beraberinde karmaşıklıkları da getirir. Merkezi veri tabanlarının yönetimine göre dağıtık veri tabanlarının yönetimi daha karmaşıktır. Bu da yönetime zorluk getirir. Yönetimin karmaşık olmasının yanında yönetilecek veritabanının ilk defa tasarlanacak olması da merkezi veri tabanlarına göre daha karmaşıktır.

3.3.2.2. Maliyet

Merkezi veri tabanlarına göre dağıtık veri tabanlarının bakım ve tedarik maliyetleri daha fazladır. Bunun yanında dağıtık veri tabanları düğümler ya da dağıtık yapıdaki sunucular arasında gerekli olan ağ bağlantıları için ekstra donanıma ihtiyaç duymaktadır. Sistemlerin yönetimi için daha fazla emek ve iş gücü harcanacağı için bu da maliyeti artıran etmenler arasındadır.

3.3.2.3. Güvenlik

Merkezi veri tabanlarının yönetimi tek bir noktada olacağı için sistemin güvenliği dağıtık yapıda olan sistemlere göre daha güvenilir ve kolay olacaktır. Dağıtık veri tabanlarında sadece veri tabanı güvenliği değil aynı zamanda ağ güvenliğinin üzerinde de ciddi şekilde çalışmalar yapılmalıdır.

3.3.2.4. Bütünlük

Bütünlük veritabanı sistemlerinde verinin geçerliliği ve tutarlılığı anlamına gelir. Dağıtık yapıda olan veritabanlarında verilerin farklı düğüm ya da bölgelerde yedeklerinin olacağı, bu yedeklerin birbirleri ile tutarlı olması gerekliliği üzerinde oldukça dikkatli durulması gereken bir konudur. Merkezi veritabanlarının olduğu sistemlerde verilerin geçerliliğini ve tutarlılığını sağlamak daha kolayken dağıtık sistemlerde bu, sorunlara neden olabilecek bir konudur.

3.3.2.5. Standart Eksikliği

Dağıtık veritabanlarının yönetildiği sistemlerde düğümler ya da sunucular arası bağlantı oldukça önemlidir. Veri erişim protokolleri ve standart iletişim araçları dağıtık veritabanı sistemlerinde henüz kullanılmaya başlamıştır.

3.4. PostgresSQL Üstünde Dağıtık Veritabanı Uygulamaları

Farklı sunuculara dağıtılmış PostgresSQL verilerini okumak için temelde 3 farklı yöntem öne çıkmaktadır. Bu yöntemler, (i) Üçüncü parti yazılımlar ve Özelleşmiş Postgres Sürümleri, (ii) Yabancı Veri Sarmalı (Foreign Data Wrappers) ve dblink uzantıları ile

veri tabanı seviyesinde sunucuların birbirine bağlanması ile Özel kodlanmış bir Middleware (Screen Tracker Back-end üstünde) ile bu işlemin sağlanması şeklindedir.

3.4.1. Üçüncü Parti Uygulamalar

Üçüncü parti uygulamalardan ticari, yarı ticari ve açık kaynak birçok araç öne çıkmaktadır. Çalışma kapsamında incelenen üçüncü parti uygulamalar bu bölümde özetlenmiştir

3.4.1.1. Postgres-XC

Dağıtık veri tabanını ölçeklemek üzere Postgres üstünde göze çarpan çalışma Suziki ve arkadaşları [40] tarafından yapılan çalışmalar ve Postgres-XC [1]olarak adlandırdıkları bir üründe bir araya getirmişlerdir. Postgres-XC kısaca şu şekilde özetlenmektedir: Paylaşımsız mimariye uygun bir DDBMS olarak Postgres üstüne kodlanmıştır. Birden fazla PostgreSQL sunucusunu bir arada çalıştırmayı hedefler. Sıradan sunucular ekleyerek ölçeklemeyi hedefler. Birden fazla düğüm ekleyerek Yazma/Okuma ölçeklemesi yapabilecek ayarlamaları mevcuttur. Senkron olarak çalıştırıldığında düğümlerden birine yazılan verileri diğer düğümlere yansıtmaktadır. Uygulama verilerin dağılımından soyutlanmıştır. Ancak çaba gerektiren ürün seviye(production) karmaşıklığındaki veri yönetimleri için yeterli olgunlukta bir sürümü bulunmamaktadır.

3.4.1.2. Postgres-XL

Postgres-XL[2] de Postgres-XC'ye benzer şekilde Postgres üstünde çalışan üçüncü parti yazılım olup Postgres-XC'nin OLAP desteği eklenerek geliştirilmiş yeni ve başka bir sürümüdür. Master-Slave olarak ayarlanabilmektedir. Aralık Hash ve Round Robin ile paralel parçalama yapabilmektedir. ANSI SQL 2008 desteği sağlamayan bir çözümdür. İlişkisel Veri Tabanları için temel kurallar olan ACID (Atomiklik, Tutarlılık, Izolasyon, Sağlamlık) desteğini garanti eden yazılım çok versiyonlu eşzamanlılık (Multi Version

[1] https://sourceforge.net/projects/postgres-xc

[2] https://www.postgres-xl.org/

Concurrency Control-MVCC) kullanır. Bu nedenle de diğer 2-Aşamalı kilitleme (2 Phase Locking) yaklaşımlarına göre daha iyi performans sağlar.

Yapılan bir çalışmaya göre [41] Grupsal fonksiyonlarda (Aggregate functions) Postgres-XC geleneksel DBMS üstünde çalıştığından Map&Reduce temelli yaklaşımlara göre daha performanslı çalışırken, diğer sorgularda da belirgin bir performans sergilemektedir. Bu temelde Porsgres-XL'in başarımı gibi görünse de RDBS olarak Postgres'in halen ilişkisel veriler için performans açısından Map&Reduce'e göre belirgin bir performans sağlayabileceğini göstermektedir.

3.4.1.3. CitusDB

Postgres üzerine geliştirilmiş yatay ölçeklenebilir bir veritabanı olan CitusDB[3] büyük paralel analitik sorgu ve gerçek zamanlı okuma- yazma desteği mevcuttur. SQL'in gücünü kullanır. Yatay olarak ölçeklendirilebilir olan gerçek zamanlı büyük veri sorunlarını çözmek için PostgreSQL'i cluster çalışma ve yüksek oranda paralel sorgu işleme yeteneklerine genişletir [42]. CitusDB sütun bazlı Postgres OLAP uygulamaları için de kullanılabilmektedir [43].

3.4.1.4. Greenplum

Greenplum[4], Bizgres'in değiştirilmesiye elde edilmiş, Büyük Veri(Big Data) odaklı bir Postgres'in geliştirilmiş farklı açık kaynak sürümüdür [44]. Kendine özgü maliyet temelli sorgu iyileştiriciye sahiptir bu iyileştiricisi sayesinde petabayt ölçeğindeki verilerin analizini yapabilmektedir. Greenplum dünyanın en gelişmiş [42] maliyet temelli sorgu iyileştiricisi tarafından desteklenmekte ve büyük veri hacimlerinde yüksek analitik sorgu performansı sağlamaktadır. Özellikle standart ANSI-SQL uyumlu olması ve OLAP ve raporlama kaynaklı iş yüklerine SQL desteği sağlaması avantajlarındandır.

---

[3] https://www.citusdata.com/

[4] http://greenplum.org

3.4.1.5. Presto

Presto[5] Apache tarafından açık kaynak olarak desteklenen bir Dağıtık SQL Sorgu motorudur [42]. Hive, Cassandra, ilişkisel veri tabanları hatta sıradan veri kaynaklarını bile etkileşimli olarak sorgulamaya izin vermektedir. Genel amaçlı bir dağıtık büyük veri uygulaması olarak dikkat çekici olsa da, sadece Postgres verilerinin dağıtımı için çok amaçlılığından gelen karmaşıklığı nedeniyle Screen Tracking uygulaması arka planı için çok etkin görünmemektedir.

3.4.2. Dblink ve Yabancı Veri Sarmalları ile DB Seviyeli Bağlantılar Üçüncü Parti Uygulamalar

Postgres DBMS Dblink ve Yabancı Veri Sarmalları ile de diğer Postgres veri tabanı sunucularına her ne kadar erişim olanağı sağlasa da bu geniş ölçekli verilerin dağıtımı için yeterli performansı sağlamamaktadır.

3.4.3. Screen Tracker'a Özel Yaklaşım

Bu çözüm, 4. Bölümde ayrıntılarıyla anlatılmıştır. Her bir kullanıcının profil verisinin hangi düğüme yazılacağı ve hangi düğümden okunacağı Dizin Sunucusu'nda tutulur. Dizin sunucusunda kullanıcılar bağlı oldukları firmalara göre gruplanmışlardır. Sistemde kayıtlı bir firma için dağıtık sunuculardan birindeki veritabanı üzerinde bir şema oluşturulmakta ve bu firmaya ait tüm çalışanlara ait veriler bu şema içerisine kaydedilmektedir. Böylece okuma ve yazma işlemleri farklı sunuculara yönlendirilerek yük dengelemesi sağlanmıştır. Ayrıca her bir sunucu içerisindeki kaynakların etkin kullanımı için dinamik thread yönetim algoritması geliştirilmiştir.

Bütün bu verilerin ve yüklerin dağılımına ilişkin bileşenler, Screen Tracking uygulamasının Arka Uç (Back-end) bileşeni içerisinde kodlandığından, genel amaçlı karmaşık iş rutinlerinden arındırılmış ve amaca yönelik olarak tampon(cache) işlemleri de bu bileşen tarafından yönetilmektedir.

---

[5] https://prestodb.io/

Çizelge 3.1 Farklı yöntemlerin esneklik ve sürdürülebilirlik açısından kıyaslanması

Yöntem	***Esneklik***	***Sürdürülebilirlik***
Postgres-XC	Alt seviye konfigürasyona ihtiyaç duymakta	Açık kaynak akademik kod. Ticari bir ürüne dönüştürülmemiş
Postgres-XL	Alt seviye konfigurasyon	Ticari seviyede değil
CitusDB	Basit kurulum	Ticari seviyede
Greenpulm	Özelleşmiş bir Postgres sürümü olduğundan, zayıf	Ticari seviyede olsa da uzun vadede projenin Postgres DBMS dışına taşınmasını gerektirebilir.
Apache Presto	Genel amaçlı yazılım	Genel amaçlı ürün olduğundan adapte edilmesi daha fazla çaba gerektirmektedir.
Dblink ve Foreign Data Wrapper	Kısıtlı yetenek	Postgres ile birlikte geliştirilmesi devam etmekte
Özel Yaklaşım (Screen Tracking Back End)	Amaç için özel geliştirilmiş olması nedeniyle çok esnek	Ürünle birlikte geliştirilebilir.

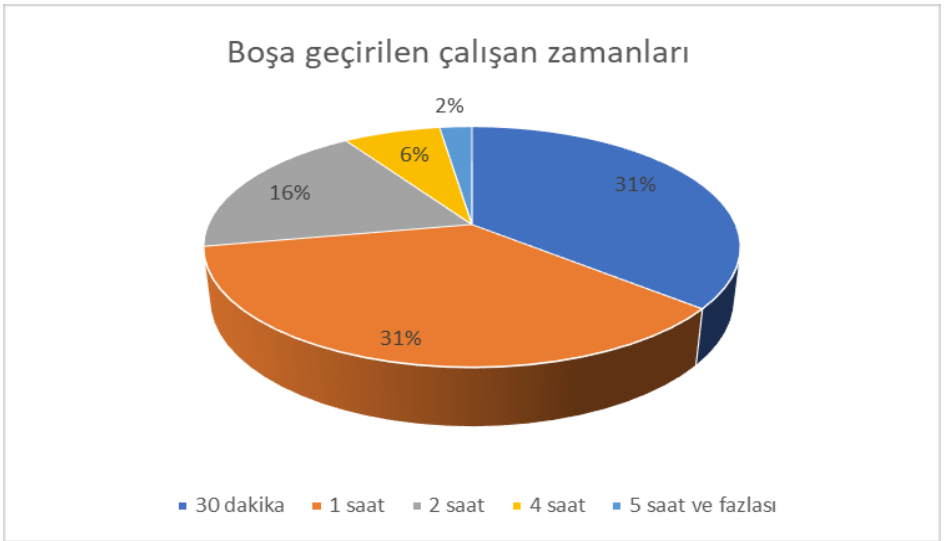
BÖLÜM 4

SCREEN TRACKER SİSTEMİ

Önceki bölümlerde paralel mimarilerin, dağıtık mimarilerin performans, maliyet, yönetilebilirlik, güvenlik vb. gibi yönlerden detaylı irdelemesi yapıldı. Bu bölümde ise dağıtık mimariye sahip pratik bir uygulamadan söz edilecek ve uygulamanın gereksinimleri gösterilerek sistemin artıları ve eksileri üzerinde durulacaktır.

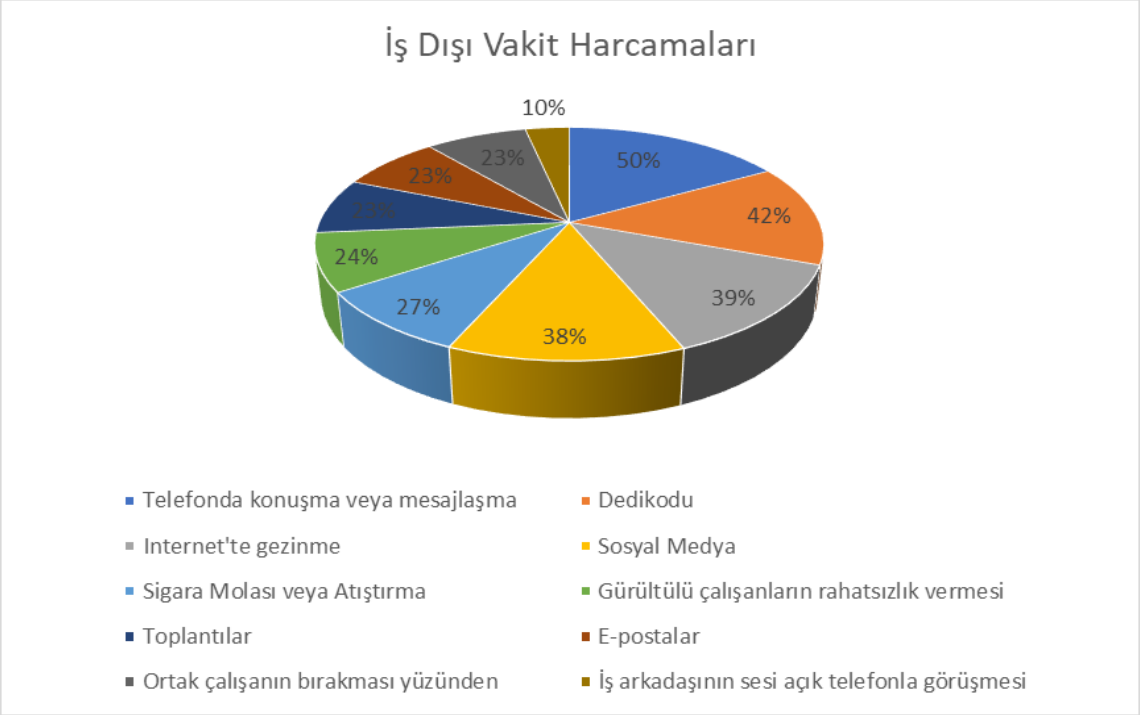
Ofis çalışanlarının veya ofis dışında çalışanların (evden çalışanlar, uzaktan yazılım geliştirenler vb.) mesai saatleri içerisinde oluşturdukları belirli profil verileri (Aktif kullanılan programlar, bağlanılan internet siteleri, network kullanım miktarı, boşta geçen zaman, ekran görüntüleri vb. gibi) raporlayarak iş verimliliğini geliştirmeyi hedefleyen bir yazılımdır.

Forbes dergisinde yayınlanan bir çalışmaya göre [45] ofiste çalışanların %60'dan fazlası bile işyerinde en az 1 saatini (Şekil 4.1) boşa geçirmektedir.



Şekil 4.1 Çalışanların yüzde olarak boşa geçirdikleri süre

Aynı çalışmaya göre çalışanların iş dışında vakit ayırdıkları konular Şekil 4.2 ‘de verilmiştir.



Şekil 4.2 İş dışında vakit harcanan konular

Screen Tracker, bütün bu konu dışı vakit harcamalarının minimize edilmesini hedefleyerek geliştirilmiş bir yazılımdır.

4.1. Genel Sistem Mimarisi

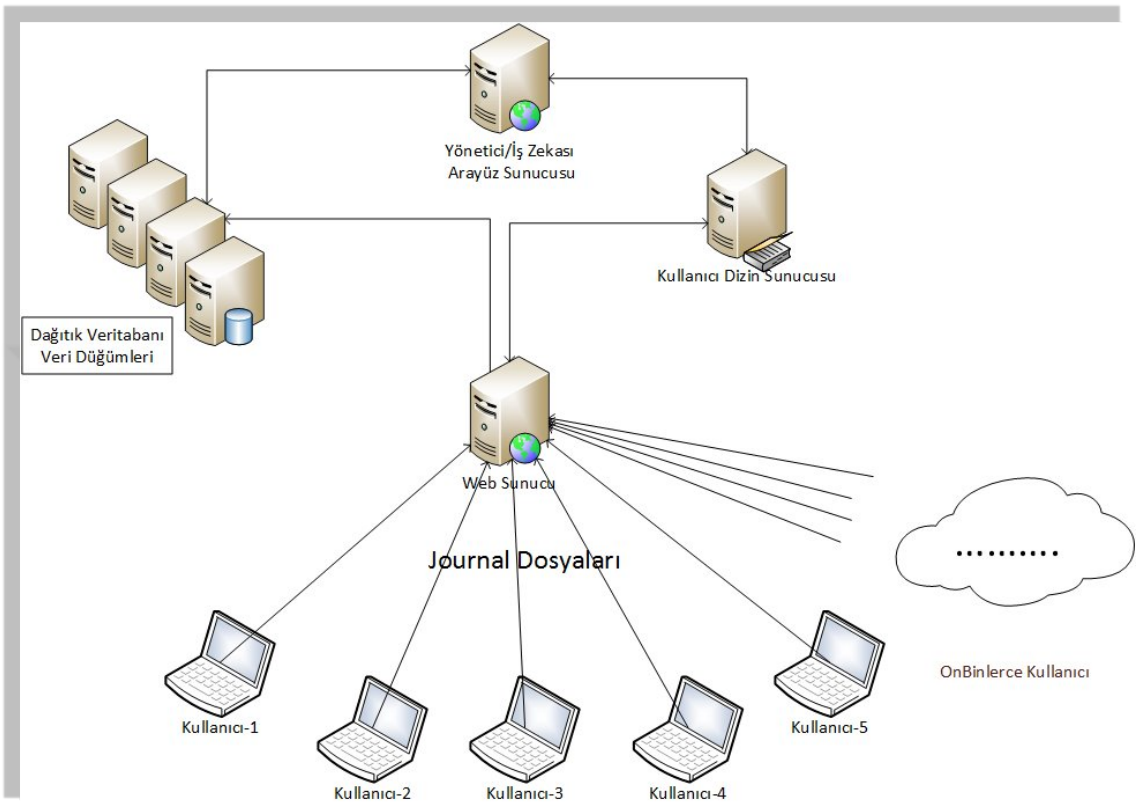
Yazılım istemci (Client) taraf, arka taraf (Back-End) ve raporlamalar olmak üzere üç temel bileşenden oluşur.

Çalışan bilgisayarına kurulan bir istemci yazılım, belirli aralıklarla çalışanların etkinliklerini bilgisayarında açtığı journal dosyalarına kaydeder. Oluşan bu journal dosyaları daha sonra merkezdeki sunuculara belirli aralıklarla aktarılır.

Bu aşamadan sonra arka taraf bileşeni devreye girer. Merkeze aktarılan journal dosyalarını işleyerek sorgulanabilir hale getirir ve dağıtık yapıdaki veritabanlarına kaydedilir.

Üçüncü bileşen ise dağıtık olarak yazılmış olan bu verileri raporlayarak kullanıcıların (veya çalışanların) verimlilikleri ve vakitlerini ne şekilde geçirdiklerine dair raporlar üreterek çalışan verimliliğinin gelişmesine katkıda bulunur.

Sistemin genel mimarisi Şekil 4.3'te gösterilmiştir.



Şekil 4.3 Screen Tracker Genel Sistem Mimarisi

Her kullanıcının oluşturduğu profil verisinin hangi veri düğümüne yazılacağı ve hangi düğümden sorgulanacağı bilgisi Dizin Sunucusunda bulunmaktadır. Dizin sunucusunda kullanıcılar bağlı oldukları firmalara göre gruplanmışlardır. Sisteme kayıtlı her bir firma için dağıtık veri tabanlarından birisi üzerinde bir şema oluşturulmakta ve firmada çalışan tüm kullanıcı verileri bu şema içerisine kaydedilmektedir. Bu şekilde dağıtık veritabanı mimarisi oluşturulmuştur, firmalar farklı veri düğümlerindeki farklı şemalara yönlendirilerek firmalar altındaki kullanıcı verilerinin dağıtık bir şekilde yazılması sağlanmaktadır. Verileri okurken de yine okuma istekleri farklı veri düğümlerine yönlendirildiği için dağıtık veri tabanı mimarisinin sağlamış olduğu performans artışından faydalanılabilmektir.

Sistemde kayıtlı aktif veri düğümünün disk kapasitesi dolmaya başladığında veya performans değerleri azalma eğilimine girdiği durumda sisteme aktif olarak yeni bir veri düğümü tanımlanmaktadır. Mevcut aktif düğüm ise saklama (retention) düğümü olarak işaretlenmektedir. Firmalardaki kullanıcıların eski profil verileri saklama düğümlerden sorgulanabilirken yeni veriler aktif düğüm sunucusundan sorgulanabilir. Dizin sunucusu firmaların tarih bazlı verilerinin hangi düğüm sunucularda olduğu bilgisini tutmaktadır. Aplikasyonlar, verilerin bulunduğu sunucu bilgilerini öğrendikten sonra bu veritabanı sunucularından firma/kullanıcı düzeyinde sorgulama yapabilirler.

Verilerin bu şekilde düğüm sunucularına dağıtılması sorgulamalarda en iyi başarımı sağlamaktadır.

4.2. Sistemde Tutulan Veriler

Screen Tracker uygulamasında, kullanıcı sayısına bağlı olarak ölçeklemeye ihtiyaç duyan temel veri tiplerine burada yer verilmiştir. Bu veri tipleri sırasıyla kullanıcı hareketlerinin kaydedildiği activity tablosu, kullanıcının ağ kullanım durumlarının kaydedildiği band_width tablosu ile kullanıcının ekran fotoğraflarının kaydedildiği screen_shot_header ve screen_shot tablolarıdır.

Bu tablolara işlenecek verileri istemci program journal dosyaları içerisinde sunucuya göndermekte, arka uç (back end) yazılımı da bu dosyayı açarak uygun şekilde tablolara verileri dağıtmaktadır. Her bir verinin detayları şu şekildedir:

Activity tablosu: Kullanıcının hangi saniyeler arasında hangi aktiviteleri gerçekleştirdiği bilgisi bu tabloda tutulmaktadır. Kullanıcının çalıştırdığı programlar (MS Visual Studio, notepad, vb.), bağlandığı siteler (docs.google.com, facebook.com, vb.), hiçbir iş yapmıyorsa veya ekran kilitli durumda ise idle bilgisi bu tabloya aktivite olarak kaydedilir (Şekil 4.4).

Şekil 4.4 Activity tablosu

NetworkUsage (Bandwidth) tablosu: Journal dosyası içeriğinde kullanıcının o dakika içerisinde network bağlantısı sonucunda oluşan upload ve download veri miktarları bu tabloya byte cinsinden kaydedilir (Şekil 4.5).

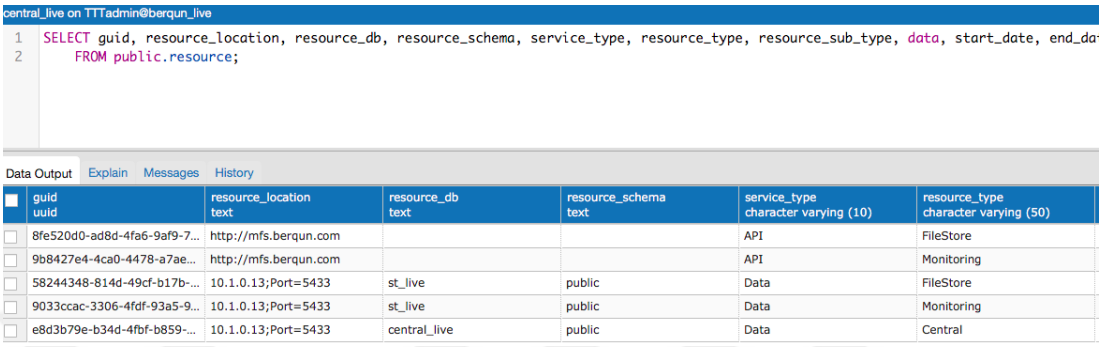
Şekil 4.5 Network kullanım miktarı tablosu

ScreenshotHeader ve Screenshot tabloları: Journal dosyası içeriğinde binary olarak bir veya birkaç screen shot verisi olabilir (birden fazla ekran kullanma senaryosu veya bir dakika içerisinde birden çok screen shot alma durumu). ScreenshotHeader tablosu master kayıt, mesela 2017-05-08 17:43'te 2 screen shot kaydı yapıldığını 1 kayıt olarak tutar. Screenshot tablosu ise bu master kayda ait 2 detay kaydı tutar. Bu kayıtta screen shot dosyalarının guid'leri, boyut ve disk üzerindeki konum bilgileri tutulur.

4.3. Sistem Tabloları

Firmalara ait veritabanı bilgileri (hangi firmanın veritabanı hangi sunucu veya sunucularda) merkezi veri tabanı dediğimiz central_live veri tabanı içerisinde bulunmaktadır. Örnek olarak sistemde kayıtlı 100 bin firma (hesap) var ise merkezi veritabanı içerisindeki firmalarla ilgili tabloda 100 bin kayıt bulunmaktadır. Bu veri

tabanı içerisindeki resource tablosu içerisinde istemcideki aplikasyonların bağlantı yapacağı aplikasyon sunucuları ve profil verilerinin yükleneceği düğümlere ait bağlantı bilgileri yer almaktadır. Resource tablosunun yapısı ve örnek içeriği Şekil 4.6'te gösterilmiştir.


central_live on TTTadmin@berqun_live

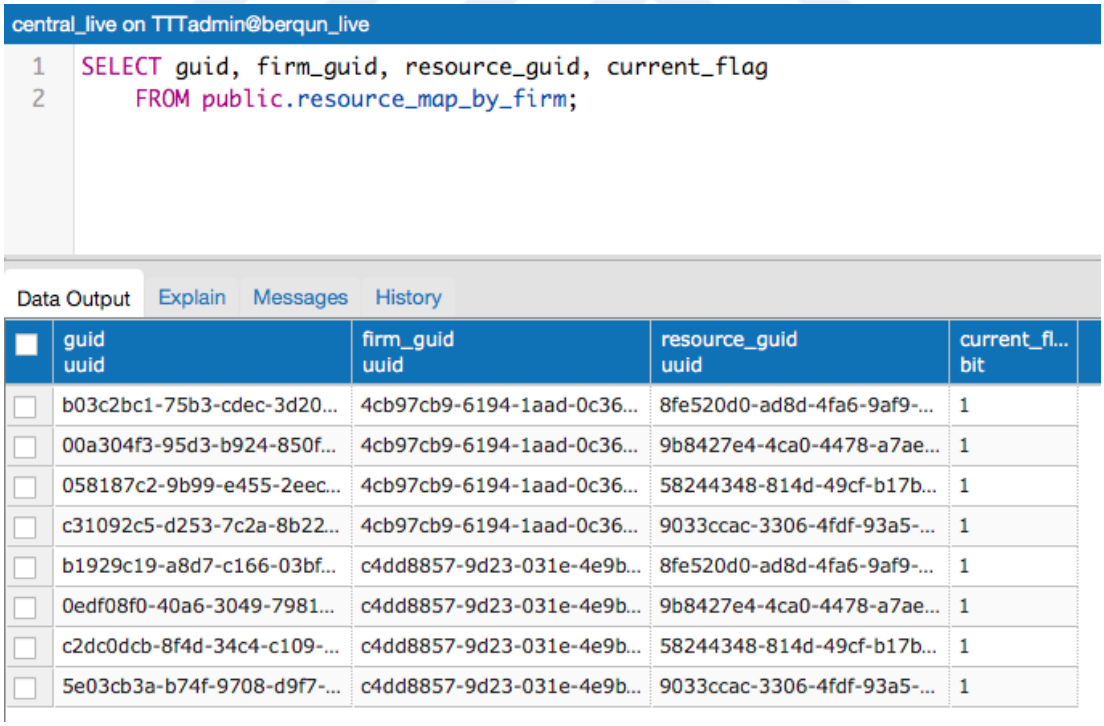
```
SELECT guid, resource_location, resource_db, resource_schema, service_type, resource_type, resource_sub_type, data, start_date, end_da
    FROM public.resource;
```

Data Output | Explain | Messages | History

guid uuid	resource_location text	resource_db text	resource_schema text	service_type character varying (10)	resource_type character varying (50)
8fe520d0-ad8d-4fa6-9af9-7...	http://mfs.berqun.com			API	FileStore
9b8427e4-4ca0-4478-a7ae...	http://mfs.berqun.com			API	Monitoring
58244348-814d-49cf-b17b-...	10.1.0.13;Port=5433	st_live	public	Data	FileStore
9033ccac-3306-4fdf-93a5-9...	10.1.0.13;Port=5433	st_live	public	Data	Monitoring
e8d3b79e-b34d-4fbf-b859-...	10.1.0.13;Port=5433	central_live	public	Data	Central

Şekil 4.6 Resource tablo yapısı

Hangi firmaya hangi resource'ların atandığı bilgisi resource_map_by_firm tablosu içerisinde yer almaktadır (Şekil 4.7).


central_live on TTTadmin@berqun_live

```
SELECT guid, firm_guid, resource_guid, current_flag
    FROM public.resource_map_by_firm;
```

Data Output | Explain | Messages | History

guid uuid	firm_guid uuid	resource_guid uuid	current_fl... bit
b03c2bc1-75b3-cdec-3d20...	4cb97cb9-6194-1aad-0c36...	8fe520d0-ad8d-4fa6-9af9-...	1
00a304f3-95d3-b924-850f...	4cb97cb9-6194-1aad-0c36...	9b8427e4-4ca0-4478-a7ae...	1
058187c2-9b99-e455-2eec...	4cb97cb9-6194-1aad-0c36...	58244348-814d-49cf-b17b...	1
c31092c5-d253-7c2a-8b22...	4cb97cb9-6194-1aad-0c36...	9033ccac-3306-4fdf-93a5-...	1
b1929c19-a8d7-c166-03bf...	c4dd8857-9d23-031e-4e9b...	8fe520d0-ad8d-4fa6-9af9-...	1
0edf08f0-40a6-3049-7981...	c4dd8857-9d23-031e-4e9b...	9b8427e4-4ca0-4478-a7ae...	1
c2dc0dcb-8f4d-34c4-c109-...	c4dd8857-9d23-031e-4e9b...	58244348-814d-49cf-b17b...	1
5e03cb3a-b74f-9708-d9f7-...	c4dd8857-9d23-031e-4e9b...	9033ccac-3306-4fdf-93a5-...	1

Şekil 4.7 Firma-resource ilişki tablosu

Central_live olarak adlandırılan merkezi veri tabanı, sistemin beyni konumundadır. Dolayısıyla sistemde hayati bir öneme sahip olan bu veritabanının birkaç tane yedeği

olacaktır. Böylelikle olası bir teknik arıza durumunda zarar görebilecek olan merkezi veri tabanının görevleri yedek veritabanları üzerinden sorunsuz olarak yürütülebilecektir.

4.4. Veri Büyüklüğü

Screen tracker uygulaması ile izlenen firma sayısı ve kullanıcı sayısı arttıkça oluşan profil verisi miktarı da hızla artacaktır. Gün içerisinde milyonlarca kayıt oluşması beklenmektedir. Ekran çıktılarının da sistemde depolandığı göz önüne alındığında sistemde disk yoğun çok sayıda işlem olacaktır. Dağıtık veri tabanı mimarisinin önemi veri miktarı arttıkça net bir şekilde ortaya çıkacaktır.

4.5. Sunucu Miktarı

Veri miktarı arttıkça doğal olarak kullanılan sunucu miktarı da artacaktır. Bir yıllık veriyi saklayabilmek için onlarca sunucu ihtiyacı olacaktır. Kullanıcı verilerinin yedeklenmesi planlandığı durumda sunucu sayısı 2 veya 3 katına çıkabileceği öngörülmektedir.

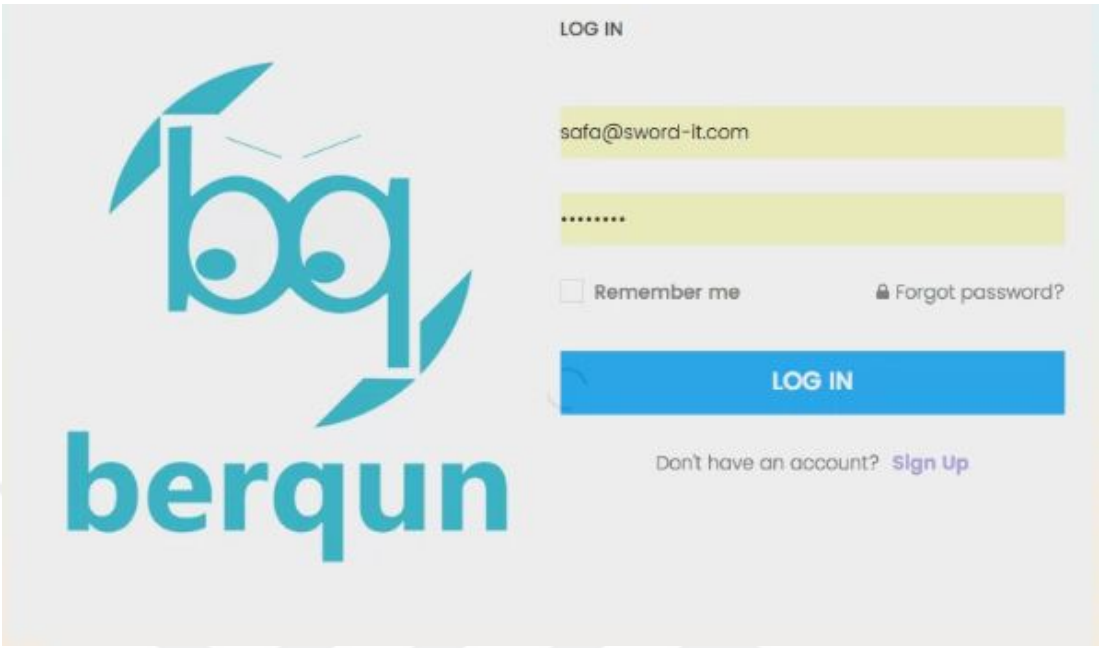
4.6. Disk Analizleri

Dağıtık veritabanı sisteminde tüm sunucularda yapılması gereken kontrollerden en önemlisi disk doluluk oranlarının tespitidir. Büyük verilerin yönetilmesinde kullanılmakta olan sistemlerden biri olan Hadoop'ta benzer kontroller mevcuttur. Hadoop sisteminde NameNode adı verilen merkezi sunucuya DataNode denilen ve verilerin kopyalarının bulunduğu sunuculardan belli aralıklarla kalp atışı mesajları (Heartbeat messages) denilen kontrol bilgileri gitmektedir [46]. Belirli aralıklarla veri düğümlerindeki disk doluluk oranları kontrol edilmeli böylece sistemde yaşanabilecek performans ve diğer problemlerin önüne geçilmelidir. Ayrıca bu sunucular üzerinde çalışan PgSQL veritabanı servisinin sürekli ayakta olduğu ve düzgün tepki verip vermediği kontrol edilmelidir.

4.7. Web Arayüzü ve Analiz Ekranları

Firmalar, çalışanlarının durumunu web sayfası üzerinden takip edebilmektedir. Veritabanına kaydedilen veriler web arayüzü vasıtasıyla sorgulanabilir ve çeşitli

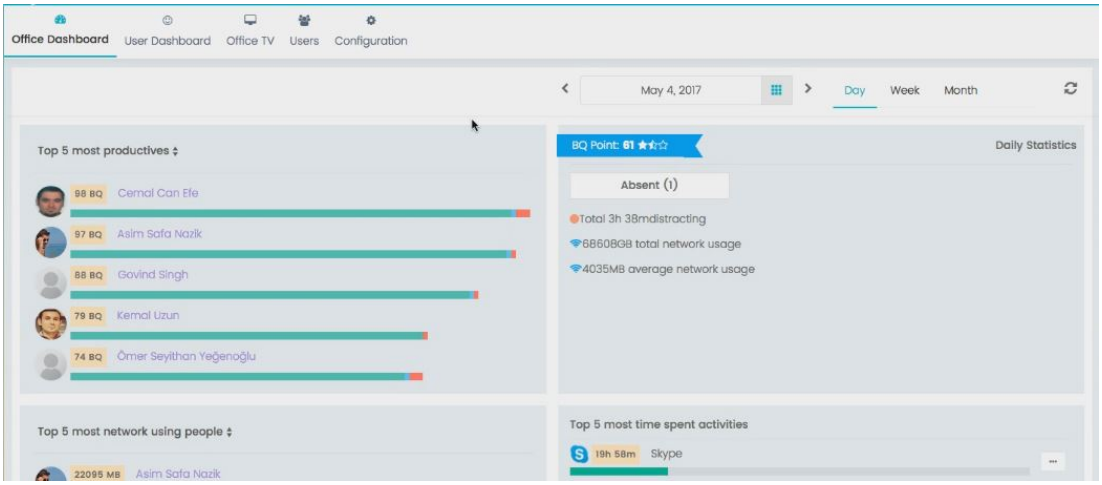
analizler yapılabilir. Örnek bir giriş ekranı ve analiz sayfası Şekil 4.8 ve Şekil 4.9'de görülmektedir.



Şekil 4.8 Screen Tracker giriş ekranı

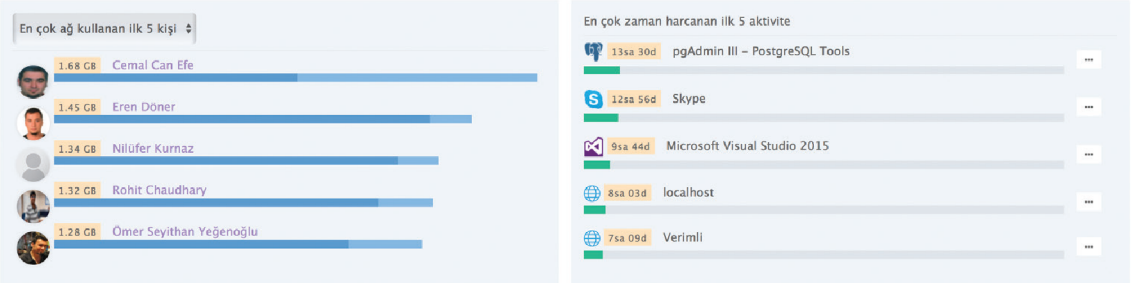
Her bir iş yöneticisine açılan hesap üstünden kullanıcı adı ve şifre girilerek, analiz raporlarına ulaşılmaktadır.

Screen Tracker analiz ekranı (Şekil 4.9) en üretken çalışanları listelemektedir.



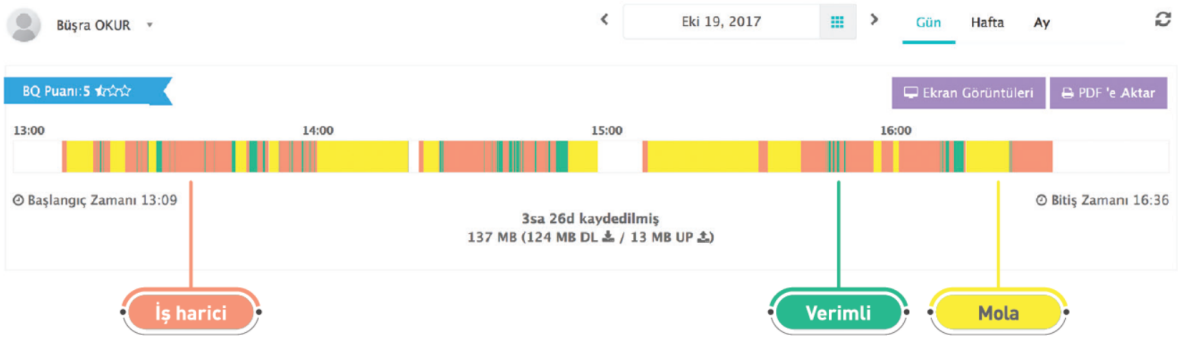
Şekil 4.9 Screen Tracker analiz ekranı

Benzer şekilde, en çok network trafiği tüketen çalışanlar ve kullandıkları uygulamalar listelenebilmektedir (Şekil 4.10).



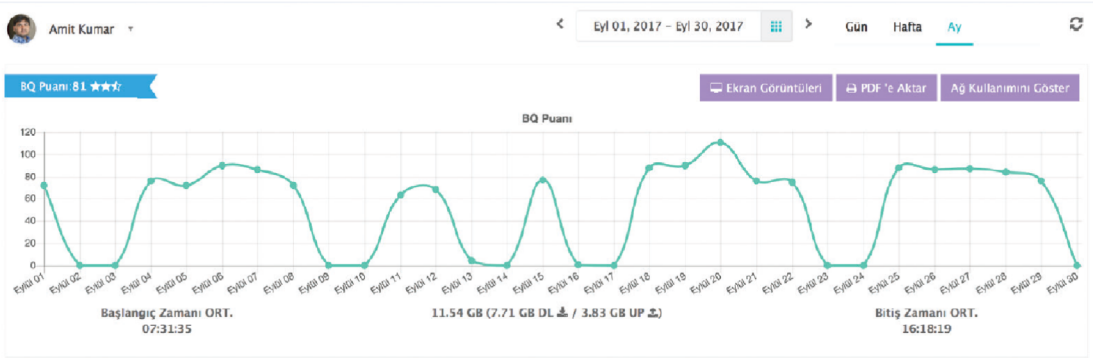
Şekil 4.10 En çok ağ kullananlar ve kullandıkları uygulamalar

Herhangi bir kullanıcının gün içerisinde nelere ne kadar vakit ayırdığı veya ne kadar iş harici zaman harcadığı görüntülenebilmekte, böylece verimliliği düşmüş çalışanları tespit edebilen yöneticisi isterse çalışanın motive edilmesine yönelik önlemlere başvurabilmektedir (Şekil 4.11).



Şekil 4.11 Kişi bazlı günlük verimlilik raporu

Kişi bazlı aylık verimlilik raporları alınarak, bir kişinin uzun vadedeki verimliliği de gözlemlenebilmektedir (Şekil 4.12).



Şekil 4.12 Kişi bazlı aylık verimlilik raporu

Görüldüğü üzere birçok raporlama üretilmektedir. Bu nedenle dağıtık yapı yazma yoğun olduğu kadar okuma yoğun da bir sistem olarak tasarlanmıştır. Kullanıcı sayısı arttıkça eşzamanlı yazma ve okuma sayılarının da dağıtık yapı ile ölçeklenmesi sağlanmıştır.

BÖLÜM 5

GERÇEKLEME VE DENEYSEL ÇALIŞMALAR

Bu bölümde ScreenTracker sistemi üstünde gerçekleştirilen çeşitli testler, kodlanmasında kullanılan yaklaşımlar ve sistemin mimarisi açısından yapılan bazı seçimlerden söz edilmiştir.

Screen Tracker sisteminin altyapısında kullanılan PgSQL ile tasarlanmış olan yapının artı ve eksi yönlerini görebilmek amacıyla back-end olarak adlandırılan verilerin saklandığı arka taraftaki yapıyı alternatif bir sistem ile kurgulanıp bir test ortamı oluşturulmuştur. Oluşturulan Java temelli bu sistem, çok iş parçacıklı (multi-threaded) olarak yük testleri amacıyla yeniden tasarlanmış ve böylece sanal yüklerle, eş zamanlı kullanıcıların sayısı deneysel olarak artırılıp azaltılabilmiştir. Bu amaçla, ilk sistem (Sistem-1) sadece tek bir sunucu üstünde çalışan bir veri tabanı ile çalışırken, ikinci sistem (Sistem-2) veri tabanının birden fazla sunucuya dağıtılarak okuma ve yazma süreçlerinin hızlandırılması hedeflenmiştir. Böylece geliştirilen Sistem-2'nin Sistem-1'e nazaran sağladığı fayda kıyaslanmıştır. Ayrıca Sistem-2'de yapılan bazı optimizasyonlarla dağıtık kaynakların etkin kullanımı hedeflenmiş ve böylece tasarlanan Sistem-3 diğer iki sistem ile başarım açısından kıyaslanmıştır.

5.1. Mimari Tercihler

Yazılım mimarisi için arka planda platform bağımsız Java, uç uygulama içinse C# ve .NET platformu kullanılmıştır. Sistemin ölçeklenebilir olması temel problem olduğundan, öncelikli olarak sistemin dağıtık hali tasarlanmıştır. Dağıtık haldeki veriler zamana bağlı aktivite verileri için genellikle ölçeklemede başvurula gelen yegâne ölçekleme yaklaşımlarındandır. Veriler genel olarak ilişkisel karakteristikte olduğundan ölçekleme

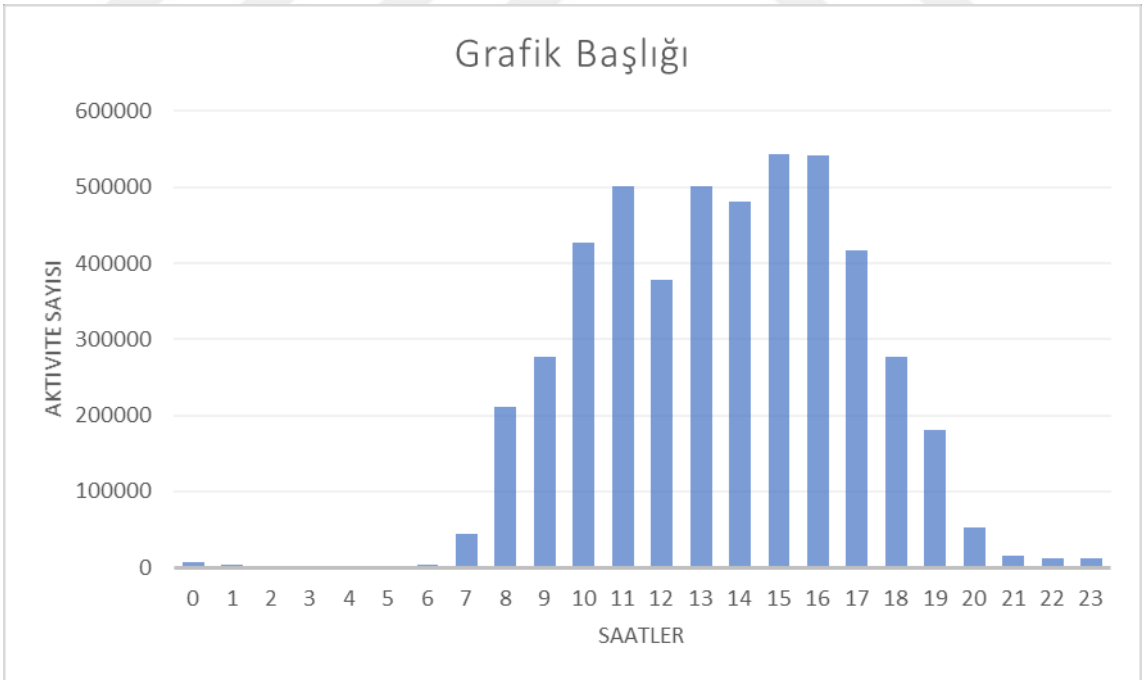
sırasında PostgreSQL VTYS kullanımından ödün verilememiştir. Son olarak, farklı sunuculara dağıtılan verilerin sunucu içerisindeki kaynakları daha etkin kullanabilmesi için de çeşitli optimizasyonlar yapılmıştır.

5.2. İş Yükünün Modellenmesi

Bir önceki bölümde de bahsedildiği gibi sistem iki temel bileşenden oluşmaktadır: Ön Uç ve Arka Uç. Bu iki bileşenin sisteme bindirdikleri yük ayrı ayrı modellenmiştir. Böylelikle modellenen yükün farklı mimarideki sistemler üstündeki performans değerleri kıyaslanabilmiştir.

5.2.1. Arka Uç Yük Karakteristiği

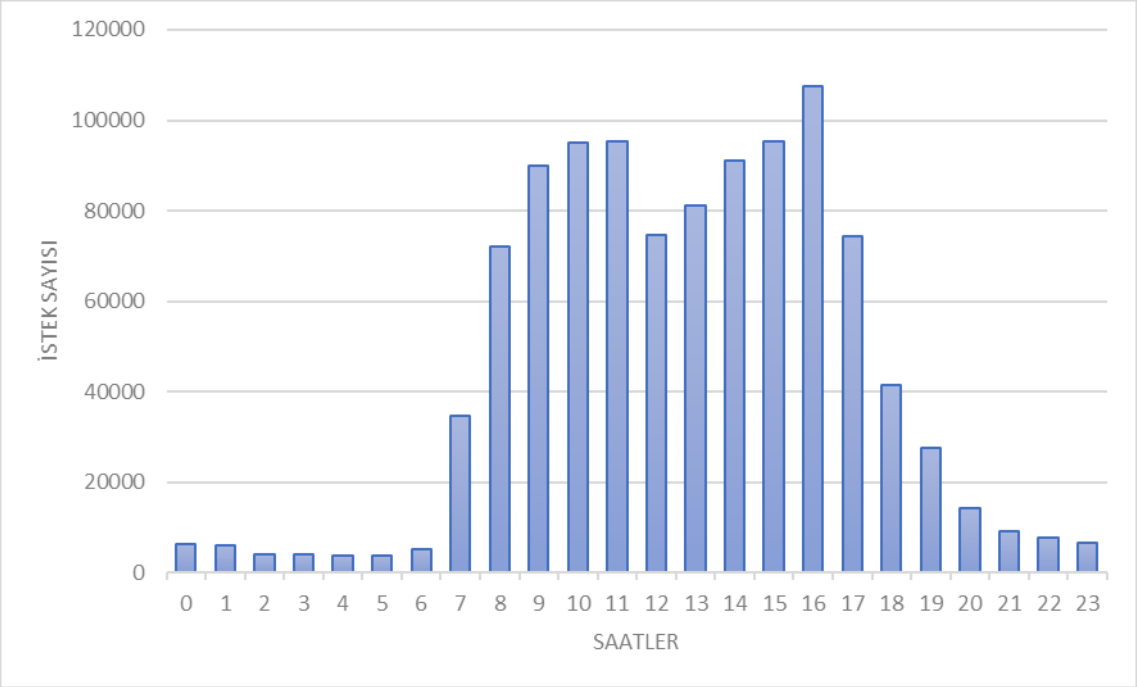
Arka uç yük karakteristiği, offline olduğu için bir süre bekletilebilir, ancak müşteriler bir an önce verileri sorgulamak isteyeceklerinden dolayı kuyruk modeli daha uygundur. Kullanıcı sayısı ile doğrudan ilişkilidir, mesai saatleri ile ciddi bir pozitif korelasyon göstermektedir (Şekil 5.1).



Şekil 5.1 Yazma yükünün gün içi saatlere dağılımı

5.2.2. Ön Uç Yük Karakteristiği

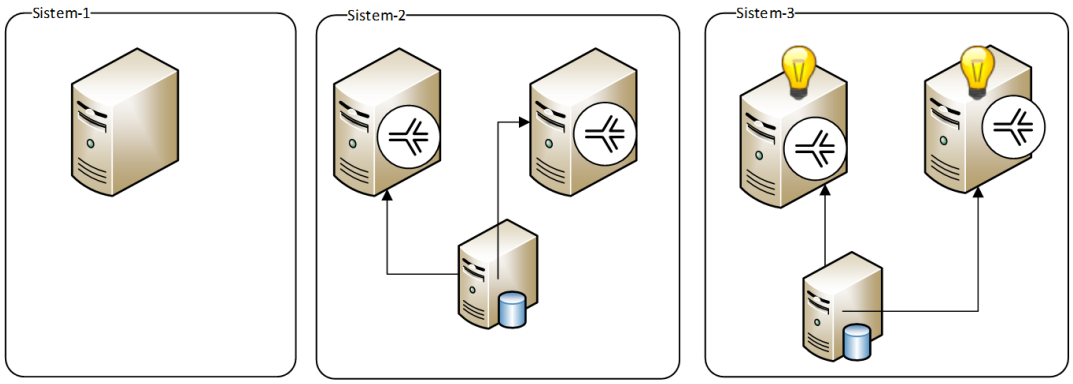
Okuma bileşenlerinin oluşturduğu yük daha kestirilemez durumdadır. Gene mesai saatleri ile yakın bir ilişki içerisindedir (Şekil 5.2) ancak kullanıcı sayısı ile doğrudan ilişkili değildir. Yazma sırasında bir bekleme zaten söz konusudur ancak okuma sırasında kullanıcı isteğinin göreceli eşzamanlı (veya kısa bir süre içerisinde) karşılanması gerekir. Okuma işlemleri için caching (ön bellekleme) yaklaşımı kısa süre içerisinde tekrarlayan okumalar için daha etkin bir yük azaltma stratejisi olarak öne çıkmaktadır [47].



Şekil 5.2 Okuma yükünün gün içi saatlere dağılımı

5.3. Ayarlamalar ve Konfigürasyonlar

Yük dağılımını yapabilmek için sisteme AWS (Amazon Web Services) üzerinden bir naming node ve iki retention node eklenmiştir. Her bir düğüme gelen journal dosyaları yazma amaçlı işlenebilmekte ve PostgreSQL'lere yüklenmektedir. Daha sonra gelen okuma istekleri de Dizin Sunucusu tarafından karşılanarak, ilgili düğümlere yönlendirilebilmektedir. Buna göre tasarlanan ağ topolojisi Şekil 5.3 şekilde tasarlanmıştır.



Şekil 5.3 Test işlemi için tasarlanan sistemler

Buna göre her bir sistemin ayrıntıları aşağıda sıralanmıştır:

- Sistem-1'de tek bir sunucu üstünde gelen bütün journal dosyalarını işlemek üzere arka plan sistemi çalıştırılmıştır.
- Sistem-2'de bir Dizin sunucusu ve 2 adet veri sunucusu birlikte çalıştırılırken CPU sayısı kadar(8 adet) ile sınırlı ve sabit, paralel thread ile iki sunucuya dağıtılarak journal dosyaları işlenmiştir.
- Sistem-3'de da Sistem-2 ile aynı konfigürasyon kullanılmış ancak adaptif thread sayısı fonksiyonu ile kuyruktaki iş sayısına göre thread sayısında dinamik yönlendirme yapılmıştır.

5.4. Performans

Her bir şirkete ait çalışanlara ait kullanıcıların aynı sunucuda olmak üzere, farklı sunuculara dağıtımı için Dizin Sunucusu'nda ayarlamalar yapılmıştır. Burada, daha önce de belirtildiği gibi sharding fonksiyonu, aynı şirkette çalışanların aynı sunucuda kalması şartıyla şirket çalışanlarının sunuculara eşite yakın dağıtılması ve aynı zamanda farklı time-zone bilgisine sahip şirketlerin (veya kullanıcıların) aynı sunuculara dağıtılması amcına yönelik bir fonksiyon düzenlenmiştir.

Şirket ve kullanıcı sayısının çok fazla artması halinde, bu türden bir fonksiyon yerine genetik algoritma veya tavlama benzetimi gibi yöntemlerle hangi şirketlerin hangi

sunuculardа tutulmasının daha faydalı olacağı hesaplanabilir. Bu konuda kasa yerleştirme problemi için yapılmış çalışmalar [48] bu probleme de uygulanabilir, ancak yapılan çalışmada sunucu sayısının bu kadar fazla artmayacağı varsayılmış ve çalışmada yükün düğümlere dağıtılmasından öte düğüm içi kaynakların etkin kullanımına odaklanılmıştır.

5.4.1. Yükün Düğümlere Dağıtımı ile İlgili Optimizasyonlar

Verilerin dağıtık sistemlere paylaştırılması sırasında iyi seçilmiş bir parçalama sütunu (Sharding Column) performans ve verilerin erişimi açısından hayati öneme sahiptir [29].

Okuma yükü ile kıyaslandığında yazma yükü çok daha belirgin maliyette ve öngörülebilir nitelikte olduğundan sharding işlemi için yazma yükü yanında okuma yükü ihmal edilebilir durumdadır. Yazma yükü, kullanıcı sayısının bir fonksiyonu olduğundan parçalama (sharding) kullanıcıların şirket bilgilerine göre yapılabilir durumdadır. Böylece, her bir şirkete ait kullanıcıların aynı düğümde kalması kısıtı çerçevesinde düğümlere kullanıcı sayılarına göre dağıtılmasının en etkin yük dağılımı sağlayacağı hesaplanmıştır. Ayrıca, her ne kadar farklı farklı timezone bilgisi kullanan müşteriler bulunmasa da timezone bilgisi birbirinden en uzak kullanıcıların aynı düğümde olmaları, ters yük etkisi nedeniyle avantaj sağlayacaktır (Şekil 5.1 ve Şekil 5.2).

5.4.2. Düğüm İçi Kaynakların Etkin Kullanımı ile İlgili Optimizasyonlar

İş yükü ile ilgili yapılan analizlerde (Şekil 5.1) iş yükünün düzenli olarak belirli bir seviyede olmadığı aşikardır. Bu durum, herhangi bir düğüm üstündeki yükün de sürekli sabit olmaması anlamına gelmektedir. Günümüz ortalama sunucuları onlarca işlemciye ve gigabyte'larca belleğe sahiptir. Bütün bu şartlar, çoklu iş parçacığı (multi-threading) yaklaşımını zorunlu kılmaktadır.

Çoklu iş parçacığı kullanan yapılarda, iş sayısı (job) ile işleyen parçacık (thread) sayısının birbirine yakın değerlerde kalması kaynakların etkin kullanımı için kritik öneme sahiptir. Belli sayıda iş parçacığını (thread) en baştan çalışır tutmak, yükün az olduğu

zamanlarda sistem kaynaklarının israfı anlamına gelecektir. Aynı şekilde, çok fazla yük varken, çok az sayışa iş parçacığı (thread) ile çalışmak da kaynak israfı anlamına gelir. Tersi olarak, sistemde çok fazla yük olduğu bir zamanda, iş parçacığı sayısını sistem kaynaklarının karşılayabileceği seviyelerin üstüne çıkartmanın, sistemin tamamen iş yapamaz hale gelmesine neden olduğu deneysel çalışmalarda gözlemlenmiştir [32].

Önceki paragrafta değinilen kısıtlamalar, sunuculardan her birinin kendi üstündeki yük için dinamik bir iş parçacığı sayısı yönetimi yapabilmesine ihtiyaç duyar. Bu nedenle, yük ve iş parçacığı sayılarını uyumlu bir hale getiren dinamik bir fonksiyon tasarlanmıştır. Bu fonksiyon, belli periyotlarla bir hesaplanan ortalama kuyruk değerine göre dinamik olarak yön almaktadır. Doğrudan dinamik bir değere anlık olarak ayarlanmamasının nedeni anlık dalgalanmalara bağlı performans kayıplarının önüne geçmektir. Ayrıca kuyrukta bekleyen iş sayısı ve thread sayısı arasında anlık değerler ve geçmişten gelen değerler belli oranda (α katsayısıyla belirlenen) birlikte kullanılarak olağanüstü dalgalanmaların [32] ve bu dalgalanmalara bağlı fazladan kuyruk beklemelerinin oluşması önlenmiştir.

Ortalama bekleyen yük değeri hesaplama fonksiyonu Bağıntı(1)'de verilen Little kanunundan hareketle şu şekildedir:

$$f(avg) = \alpha * avg_{ti-1} + (1- \alpha) * Q_{ti} \quad (5.1)$$

(5.1) denkleminde;

α : 0.25 alınmıştır sabit bir katsayıdır. Bir önceki ortalamanın yeni ortalamadaki ağırlığını belirlemektedir.

avg_{ti-1} : Kuyrukta bekleyen ve bitirilmesi gereken iş yükünün büyüklüğüne ilişkin hesaplanan i-1 zamanındaki (bir önceki iterasyon) ortalama yük değeri

Q_{ti} : kuyrukta son hesaplama iterasyonu sırasında bekleyen toplam iş sayısı

Ortalama yük Bağıntı (5.1)'de verilen fonksiyondan hareketle hesaplandıktan sonra gerekli olan thread sayısına yön veren f(wc) (3)'te verilen formül ile şu şekilde hesaplanır.

$$
f(wc) = \begin{cases} sharpDecrease(n), & time >= max\_waiting\_time \\ sharpIncrease(n), & avgs >= 2500 \\ slowDecrease\ (n), & avg < 100 \\ sharpDecrease(n), & avg < 10 \\ slowIncrease(n), & avg > 500 \end{cases} \tag{5.2}
$$

Bağıntı (5.2)'de geçen ve Thread sayısını artırıp azaltmaya yarayan fonksiyonların detayları (Şekil 5.4)'te verilmiştir.

```
private void slowIncrease() {
    setWorkerCount(manager.runningWorkerCount + 2);
}
private void sharpIncrease() {
    setWorkerCount(manager.runningWorkerCount * 2);
}
private void slowDecrease() {
    setWorkerCount(manager.runningWorkerCount - 2);
}
private void sharpDecrease() {
    setWorkerCount(manager.runningWorkerCount / 2);
}
```

Şekil 5.4 Thread sayısı yönlendirme fonksiyonlar

Bütün bu ayarlamaların Screen Tracking uygulaması üstündeki başarımını ölçmek için Sistem-1, Sistem-2 ve Sistem-3 olmak üzere üç farklı konfigürasyon oluşturulmuştur.

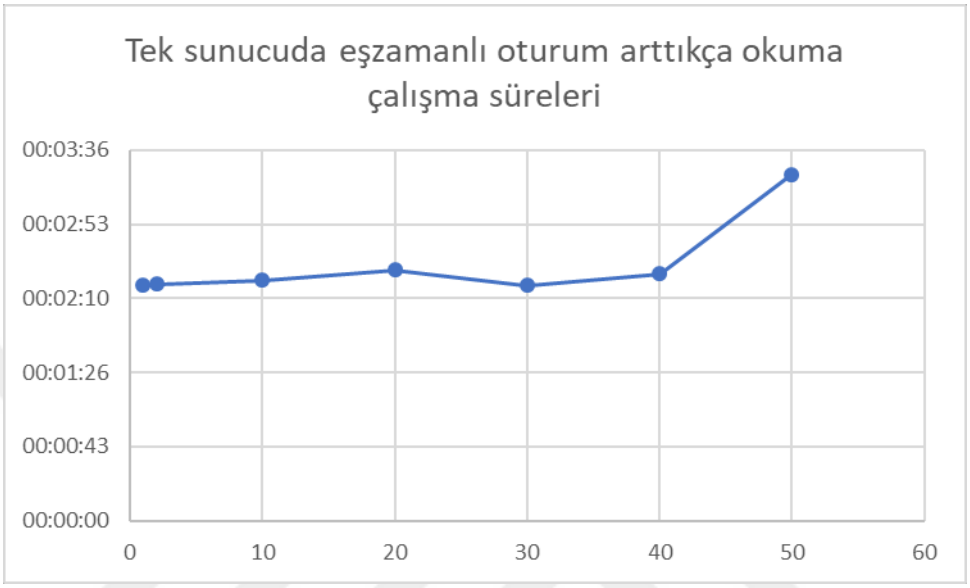
5.5. Deneysel Çalışmalar

Okuma yükündeki olağanüstü artışların, diğer okuma yüklerine bekletme açısından bir etkisinin olup olmadığı testi yapılmıştır. Yapılan çalışmalarda, okuma yükünün 50 eş zamanlı sorgulamalara kadar herhangi bir performans darboğazı oluşturulmadığı gözlemlenmiştir.

Yazma testlerinde, uygun bir parçalama fonksiyonu ve dinamik kaynak kullanımı amacıyla fonksiyon kullanmanın performansa katkısı incelenmiştir.

5.5.1. Okuma testleri

Okuma işlemlerinin eşzamanlı artımının gecikmeye neden olma ihtimali için yapılan testlerde, 50 eşzamanlı raporlamaya kadar sunucuya çok ciddi yük oluşturmadığı ortaya çıkartılmıştır. Yapılan testlere ilişkin sonuçlar Şekil 5.5'te yer almaktadır.



Şekil 5.5 Eşzamanlı okuma sayısının çalışma süresine etkisi

5.5.2. Yazma Yükü ile İlgili Testler

Deneysel çalışmalarda 3 farklı sistem konfigürasyonu kullanılmıştır. Sistem-1'de tek bir sunucuda tek başına journal dosyalarını açıp veri tabanına yükleyen bir proses çalıştırılmış ve iki farklı sanal şirkete ait Screen Tracking uygulamasından gelen her bir dosyanın ortalama bekleme süresi ölçümlenmiştir. Aynı yük Sistem-2'de farklı iki sunucuya dağıtılmış ve dosyaların ortalama süreleri yeniden ölçümlenmiştir. Sistem 3'te ise Sistem-2'den farklı olarak her bir sunucudaki yerel kaynakların daha etkin kullanımı için dinamik thread yönelim fonksiyonundan yararlanılmıştır.

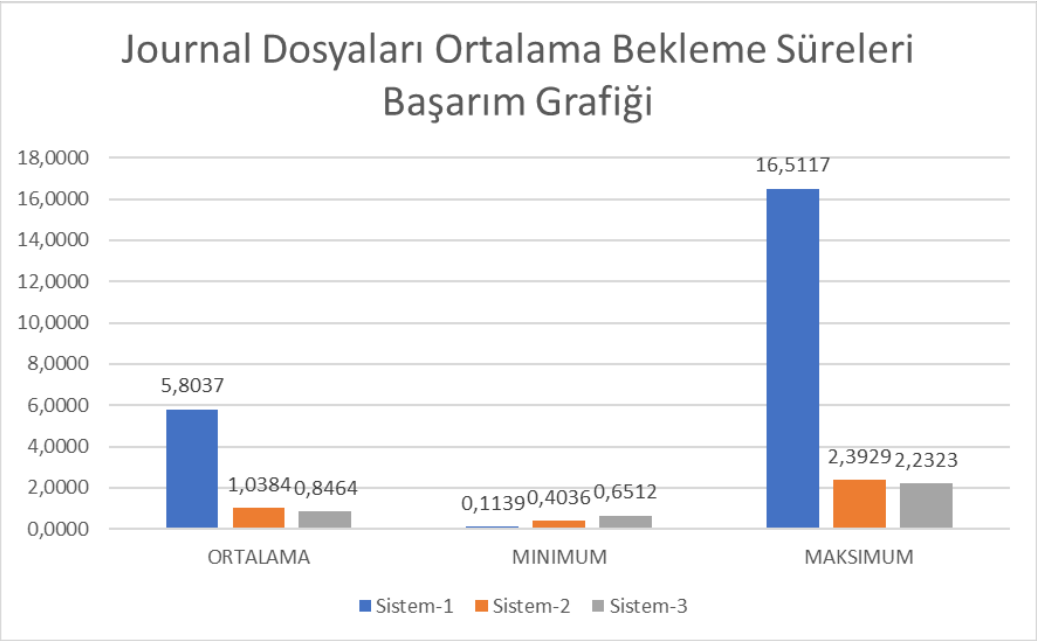
Her bir sistemde 20 eş zamanlı kullanıcının 1dk aralıklarla sunucuya journal dosyaları göndermeleri test yükü olarak kullanılmıştır. Bu amaçla bir stres testi istemci uygulaması tasarlanmış ve sunucuya düzenli olarak journal dosyaları gönderilmesi sağlanmıştır. Her bir sisteme ait ortalama, minimum ve maksimum bekleme süreleri

Çizelge GERÇEKLEME VE DENEYSEL ÇALIŞMALAR.2'de gösterilmiş ve Şekil 5.6'da kıyaslanmıştır.

Çizelge 5.1 Farklı konfigürasyondaki sistemlerde yazma performansının kıyaslanması

	Sistem-1 Performansı	**Sistem-2 Performansı**	**Sistem-3 Performansı**
Thread Sayısı	1	Cpu sayısı(8) kadar	Dinamik Thread Yönlendirme Fonksiyonu
Sunucu Sayısı	1	2 Saklama Sunucusu 1 Dizin Suncusu	2 Saklama Sunucusu 1 Dizin Suncusu
Kuyrukta Ortalama Bekleme Süresi (sn/dosya)	5,8037	1,0384	0,8464
Minimum Bekleme Süresi(sn)	0,1139	0,4036	0,6512
Maksimum Bekleme Süresi(sn)	16,5117	2,3929	2,2323

Buna göre, Sistem-2 ve Sistem-3'de görüldüğü üzere dağıtık olarak çalışma başarılı bir şekilde performans artışı sağlamıştır. Sistem-3'de ayrıca kullanılan adaptif thread yönelim fonksiyonu %19 oranında bir sistem iyileşmesi daha sağlamıştır.



Şekil 5.6 Farklı sistemlerde paketlerin bekleme süreleri

Özellikle düşük yüklerde Sistem-1 oldukça hızlı tepki verebilirken(Minimum bekleme süresi 0,1139 sn ile) sisteme gelen anlık yük arttığında, Sistem-3'ün bariz bir şekilde kaynakları etkin kullanarak bekleme süresinde iyileşme sağladığı ölçümlenmiştir.

BÖLÜM 6

SONUÇ VE ÖNERİLER

Mevcut sistem ile alternatif sistemin karşılaştırılması yapılarak elde edilen sonuçlar karşılaştırılmış ve öneriler sunulmuştur. Buna göre, parçalama sütunu olarak şirket bilgisi kullanılmış böylece verilerin aynı sunucudan sorgulanarak cevaplanabilmesi sağlanmıştır. Bir sunucuda birden fazla şirket tutulacaksa, yük dağılımının yanı sıra, birbirine en uzak zaman dilimi (timezone) bilgisine sahip şirketlerin aynı sunucuya konulmasının performans açısından olumlu katkı sağlayacağı ortaya çıkartılmıştır.

Round Robin ile sunuculara dağıtılan işlerin performans için tek başına yerel kaynakların faydalı kullanımı için yeterli olmayacağı ortaya konulmuştur. Bu nedenle de her bir yerel sunucuda işlerin beklemede geçirdiği süre, thread sayısı ve kaynaklar dikkate alınarak dinamik bir şekilde sunucuda çalışan thread sayısını artırıp azaltmak için bir formül önerilmiş ve formülün, Round Robin dağıtımına göre sağladığı performans her bir işin ortalama bekleme süreleri kıyaslanarak ortaya konulmuştur.

Sonuç olarak, tek bir sunucu üstünde çalışmak üzere tasarlanmış olan Screen Tracking uygulaması için dağıtık olarak uygun performans altında çalışabileceği şartlardan en elverişli olanı gerçeklenmiştir. Bu yöntem özellikle bulut sistemlerde işlemci kullanımı başına ücretlendirme gibi politikalardan dolayı bir maliyet optimizasyonu da sağlamaktadır. Yerel sistemlerde çalıştırıldığında da daha az sistem kaynağı(CPU) ile daha fazla yükün karşılanmasına olanak sağlamaktadır.

Her ne kadar bu çalışmada, çok fazla sayıda (yüzlerce) sunucu ya da şirketin bulunmadığı varsayılmışsa da özellikle sunucu ve şirket sayısının çok fazla artması halinde, sunuculara şirketlerin dağıtımı hakkında genetik algoritma ya da tavlama benzetimi gibi yapay zekâ tekniklerinden faydalanacak çalışmalar yapılabilir.

KAYNAKLAR

[1] Erl, T., Puttini, R., Mahmood, Z. (2013). Cloud Computing: Concepts, Technology & Architecture, Pearson Education, New York.

[2] Berka, T., Vajteršic, M. (2011). "Fast Information Retrieval In The Open Grid Service Architecture", Serdica Journal of Computing, 5:207-236.

[3] Zhao, J., Yang, K., Wei, X., Ding, Y., Hu, L., Xu, G. (2016, Şubat 1). "A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment", IEEE Transactions on Parallel and Distributed Systems, 305-316.

[4] Zappi, D. C., Filardi, F. L. R., Leitman, P., Souza, V. C., Walter, B. M., Pirani, J. R.,Forzza, R. C. (2015). "Growing knowledge: an overview of Seed Plant diversity in Brazil", Rodriguésia, 66(4):1085-1113.

[5] Krzymińska, S., Szczuka, E.,Kaznowski, A. (2012). "Staphylococcus Haemolyticus Strains Target Mitochondria and Induce Caspase-dependent Apoptosis of Macrophages", Antonie Van Leeuwenhoek, 102(4):611-620.

[6] Kleyman, B., The Big Data Battleground: Analyzing the Big Picture, http://www.datacenterknowledge.com/archives/2012/09/12/a-look-into-the-big-data-battleground-analyzing-the-market/?utm-source=feedburner&utm-medium=feed&utm-campaign=Feed%3A+DataCenterKnowledge+%28Data+Center+Knowledge%29, 12 Ocak 2018

[7] Meredith, M., Carrigan, T., Brockman, J., Cloninger, T., Privoznik, J., Williams, J. (2003). "Exploring Beowulf clusters", J. Comput. Small Coll, 268–284.

[8] Meeker, R. D. (2005). "Comparative System Performance for a Beowulf Cluster", J. Comput. Small Coll, 114–119.

[9] Capello, F., Herault, T., Dongarra, J. (2007). "Recent Advances in Parallel Virtual Machine and Message Passing Interface", 14th European PVM/MPI User's Group Meeting, 30 Ekim 2007 - 3 Ekim 2007,Paris, Fransa.

[10] Thain, D., Tannenbaum, T., Livny, M. (2005). "Distributed Computing in Practice: the Condor Experience: Research Articles", Concurrency and Computation: Practice and Experience, 323–356.

[11] Barak, A., Guday, S., Wheeler, R. (1993). The MOSIX Distributed Operating System: Load Balancing for UNIX, Springer-Verlag.

[12] Haddad , I., Paquin, E. (2001). “MOSIX: A Cluster Load-balancing Solution for Linux”, Linux journal, 6.

[13] Morin, C., Lottiaux, R., Valee, G., Gallard, P., Margery, D., Berthou, J.-Y., Scherson, I. (2004). “Kerrighed And Data Parallelism: Cluster Computing on Single System Image Operating Systems”, Proceedings of the 2004 IEEE International Conference on Cluster Computing, Computer Society, 20-23 Eylül 2004, San Diego, Kaliforniya,ABD, 277–286

[14] Lottiaux, R., Boissinot, B., Gallard, P., Vallée, G., Morin, C. (2005). “OpenMosix, OpenSSI and Kerrighed: a Comparative Study”, Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on, 27-30 Eylül 2005, Boston, Massachusetts, ABD, 1016 - 1023

[15] Foster, I., Kesselman, C. (1999). “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann Publishers Inc.

[16] Foster, I., Kesselman, C., Tuecke, S. (2001). “The Anatomy of The Grid: Enabling Scalable Virtual Organizations”, Int. J. High Perform. Comput. Appl, 200–222.

[17] Casanova, H. (2002). “Distributed Computing Research Issues in Grid Computing”, SIGACT, 50–70.

[18] Foster, I. (2002). “What is the Grid? - A Three Point Checklist”, GRIDtoday, 1(6).

[19] Foster, I. (2006). “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, Springer-Verlag, 2 – 4 Kasım 2006, Tokyo, Japonya, 2-13.

[20] Aloisio, G., Cafaro, M., Epicoco, I., Fiore, S. (2001). “Analysis of the Globus Toolkit Grid Information Service”, Lecce Italy.

[21] Weiss, A. (2007). “Computing in the clouds”, Networker, 11(4):16-25.

[22] Vaquero, L., Rodero-Merino, L., Caceres, J., Lindner, M. (2009). “A Break in The Clouds: Towards a Cloud Definition”, SIGCOMM Comput, 50-55.

[23] Buyya, R., Yeo, C., Venugopal, S., Broberg, J., Brandic, I. (2009). “Cloud Computing and Emerging It Platforms: Vision, Hype, and Reality For Delivering Computing as the 5th Utility”, Future Generation Computer Systems, 599–616.

[24] Lindner, M., Vaquero, L., Rodero, L., Caceres, J. (2010). “Cloud Economics: Dynamic Business Models for Business On Demand”, Int. J. Bus. Inf. Syst, 373–392.

[25] Lenk, A., Klems, M., Nimis, J., Tai, S., Sandholm, T. (2009). “What’s Inside The Cloud? an Architectural Map of the Cloud Landscape”, CLOUD ’09: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, 16-24 Mayıs 2009, Vancouver, Kanada, 23–31.

[26] Robinson, D. (2008). Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services Enables You to Reach Business Goals Faster, Emereo Pty Ltd.

[27] White, T. (2012). Hadoop: the Definitive Guide, O'Reilly, Pekin

[28] Qin, X. C. (2017). "The Performance of SQL-on-Hadoop Systems-An Experimental Study", In Big Data (BigData Congress), IEEE International Congress, 25-30 Haziran 2017, Honolulu, Hawaii, ABD, 464-471.

[29] Szczuka, M. S. (2014). "Using domain knowledge in initial stages of KDD: Optimization of compound object processing", Fundamenta Informaticae 129(4):341-364.

[30] Welsh, M. C. (2001). "An Architecture for Well-Conditioned, Scalable Internet Services", ACM SIGOPS Operating Systems Review, 230-243.

[31] Welsh, M. D. (2002). An Architecture for Highly Concurrent, Well-conditioned Internet Services(Doctoral dissertation), Berkeley: University of California.

[32] Newell, A. K. (2016). "Optimizing distributed actor systems for dynamic interactive services", the Eleventh European Conference on Computer Systems, 19-21 Nisan 2016 South Kensington, Londra, İngiltere, 38.

[33] Özsu, M., Valduriez, P. (2011). "Principles of Distributed Database Systems", Springer Science Business Media.

[34] Connolly, T., Begg, C. (2005). Database Systems A Practical Approach to Design,Implementation and Management, Addison-Wesley, İngiltere.

[35] Özsu, M. T., Valduriez, P. (1997). "Distributed and Parallel Database Systems", Handbook of Computer Science and Engineering, 1093-1111.

[36] Manchanda, N., Anand, K. (2012). Non-uniform Memory Access (NUMA), New York University.

[37] Nylund, J. (2011). Simulating Non-uniform Memory Access Architecture For Cloud Server Applications, Åbo Akademi University.

[38] Goodman, J., Woest, P. (1988). "The Wisconsin Multicube: a New Large-scale Cache-coherent Multiprocessor", ACM SIGARCH Computer Architecture, 422-431.

[39] Levin, K., Morgan, D. (1975). "Optimizing Distributed Data Bases: A Framework For Research", National Computer Conferance, 473-478.

[40] Suzuki, K., Paquier, M., & Bapat, A. (2012, May). "WriteScalable PostgreSQL Cluster", PostgreSQL, 15-19.

[41] Chihoub, H. (2016). "A Scalability Comparison Study of Data Management Approaches for Smart Metering Systems", in Parallel Processing (ICPP) 45th International Conference IEEE, 16-19 Ağustos 2016, Philadelphia, ABD, 474-483

[42] Pal, S. (2016). SQL on Big Data: Technology Architecture and Innovation, Apress,Wilmington, Massachusetss

[43] Goncalves, M., Mendoza, J. N. (2017). “A Physical Design Strategy for Datasets with Multiple Dimensions”, In Intelligent Multidimensional Data Clustering and Analysis, IGI Global, 1-27.

[44] Obe, R. O., Hsu, L. S.(2015). Postgresql: Up and Running: a Practical Guide to the Advanced Open Source Database, ABD, 15

[45] Conner, C. , Wasting Time At Work: The Epidemic Continues, https://www.forbes.com/sites/cherylsnappconner/2015/07/31/wasting-time-at-work-the-epidemic-continues/#10cc47221d94, 23 Ocak 2018

[46] Borthakur, D. (2007). The Hadoop Distributed File System:Architecture and Design, The Apache Software Foundation.

[47] Sarhan, A. E. (2014). “New web cache replacement approaches based on internal requests factor”, 9th International Conference In Computer Engineering & Systems (ICCES), 22-23 Aralık 2014, Kahire, Mısır, 383-389.

[48] Bortfeldt, A. (2006). “A Genetic Algorithm for the TwoDimensional Strip Packing Problem with Rectangular Pieces”, European Journal of Operational Research, 172(3):814- 837.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	Nurullah KILIÇ
Doğum Tarihi ve Yeri	17.07.1973 / GAZIANTEP
Yabancı Dili	İNGİLİZCE
E-posta	nurullah@sword.it

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Mühendisliği	Yıldız Teknik Üniversitesi	1999
Lise	Matematik	Vefa Lisesi	1990

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2007-2018	Sword IT	Yazılım Mimarı
2006-2007	Microsoft Canada	Veri tabanı danışmanı
2005-2006	Ajilon Canada	.NET mimarı
2003-2005	Gavel & Gown Canada	Veri tabanı uzmanı

2001-2003	Enter Bilgisayar	Yazılım Uzmanı
1999-2001	Kuveyt Turk	Yazılım Uzmanı