İLİŞKİSEL VERİ TABANLARININ UYGULANDIĞI ARAÇ TAKİP SİSTEMİNİN BÜYÜK VERİYE UYARLANMASI

Ferhat KOÇER
151402125

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Tezli Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Hüseyin Fehmi Selim BAYRAKLI

İstanbul
T.C. Maltepe Üniversitesi
Fen Bilimleri Enstitüsü
Ocak, 2019

JÜRİ VE ENSTİTÜ ONAYI

Ferhat KOÇER'in "Açıklama:İlişkisel Veri Tabanlarının Uygulandığı Araç Takip Sisteminin Büyük Veriye Uyarlanması" başlıklı tezi 25.01.2019 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Maltepe Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği"nin ilgili maddeleri uyarınca, Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans Doktora tezi **oy birliğiyle / oy çokluğuyla** olarak kabul edilmiştir.

	Unvanı. Adı ve soyadı	İmza
Üye (Tez Danışmanı)	: Dr. Öğr. Üyesi Selim BAYRAKLI	.....
Üye (Manisa Celal Bayar Üniversitesi)	: Dr. Öğr. Üyesi Fatih YUCALAR	.....
Üye	: Dr. Öğr. Üyesi Volkan TUNALI	.....

Prof. Dr. İlter BÜYÜKDIĞAN

Enstitü Müdürü

ŞEKİL ONAY SAYFASI

20../02../2019

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE.

Aşağıda bilgileri bulunan lisansüstü öğrencinin tezi şekil yönünden tarafımca incelenmiş ve Enstitüye teslim edilmesi uygun bulunmuştur.

Anabilim Dalı Başkanı
Dr. Öğretim Üyesi Ali AKMAN

ÖĞRENCİ BİLGİLERİ	
ADI SOYADI	Ferhat KOÇER
ÖĞRENCİ NUMARASI	151402125
ANABİLİM DALI	Bilgisayar Mühendisliği Tezli Yüksek Lisans
PROGRAMI	(X) YÜKSEK LİSANS ( ) DOKTORA ( ) SANATTA YETERLİK
DANIŞMANI	
TEZ BAŞLIĞI	İlişkisel Veri Tabanlarının Uygulandığı Araç Takip Sisteminin Büyük Veriye Uyarlanması
SAVUNMA TARİHİ	25/01/2019
e-posta	ferhatkocer25@gmail.com

İç Kapak	☒ Var ☐ Yok
Jüri Onay Sayfası	☒ Var ☐ Yok
Etik İlke ve Kurallara Uyum Beyanı	☒ Var ☐ Yok
İntihal Raporu	☒ Var ☐ Yok
Teşekkür Sayfası	☒ Var ☐ Yok

Hazırlayan İlgili Birim	Kalite Koordinatörü Dr. Öğr. Üyesi Şafak GÜNDÜZ	Kurumsal Yetkili Prof. Dr. Belma AKŞİT

	ŞEKİL ONAY SAYFASI	Doküman No	FR-105
		İlk Yayın Tarihi	20.12.2017
		Revizyon Tarihi	10.12.2018
		Revizyon No	01
		Sayfa	2/2

Öz (Başlık-Öz-Anahtar Sözcükler)	☒ Var ☐ Yok
Abstract (Title-Abstract-Key Words)	☒ Var ☐ Yok
İçindekiler	☒ Var ☐ Yok
Çizelgeler Listesi	☒ Var ☐ Yok
Şekiller Listesi (varsa)	☐ Şekil yok ☒ Uygundur ☐ Uygun Değildir
Kısaltmalar Listesi	☒ Var ☐ Yok
Tablolar Listesi (varsa)	☒ Tablo yok ☐ Uygundur ☐ Uygun Değildir
Ekler Listesi (varsa)	☒ Ek yok ☐ Uygundur ☐ Uygun Değildir
Özgeçmiş	☒ Var ☐ Yok
Sayfa Genişliği	☒ Uygundur ☐ Uygun Değildir
Yazı Tipi	☒ Uygundur ☐ Uygun Değildir
Referans Kullanımı	☒ Uygundur ☐ Uygun Değildir
Kaynakça Yazımı	☒ Uygundur ☐ Uygun Değildir
Ekler (varsa)	☒ Ek yok ☐ Uygundur ☐ Uygun Değildir

Dr. Öğr. Üyesi Erdal GÜVENOĞLU

Hazırlayan İlgili Birim	Kalite Koordinatörü Dr. Öğr. Üyesi Şafak GÜNDÜZ	Kurumsal Yetkili Prof. Dr. Belma AKŞİT

Revizyon Takip Tablosu

REVİZYON NO	TARİH	AÇIKLAMA
00	01.03.2018	İlk yayın.

ETİK İLKE VE KURALLARA UYUM BEYANI

05/02/2019

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarından bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilmeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; çalışmanın Maltepe Üniversitesinde kullanılan "bilimsel intihal tespit programı" ile tarandığını ve öngörülen standartları karşıladığını beyan ederim.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

Ferhat KOÇER

İLİŞKİSEL VERİ TABANLARININ UYGULANDIĞI ARAÇ TAKİP SİSTEMİNİN BÜYÜK VERİYE UYARLANMASI

ORİJİNALLİK RAPORU

%8	%7	%1	%3
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	%3
2	oranteknoloji.com İnternet Kaynağı	%3
3	dergipark.gov.tr İnternet Kaynağı	%1
4	Submitted to Beykent Universitesi Öğrenci Ödevi	<%1
5	ÖZTÜRK, Serdar and ATMACA, Hatice Ediz. "İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi", Gazi Üniversitesi Bilişim Enstitüsü, 2017. Yayın	<%1
6	KOCA, Başar and CEYLAN, Ayhan. "Uydu Konum Belirleme Sistemlerindeki (GNSS)	<%1

incelenmiştir
Dr. Öğr. Üyesi Selim Bayraklı

TEŞEKKÜR

Tez konusunun belirlenmesinden son aşamasına kadar tüm adımlarda bana kılavuzluğunu esirgemeyen, çalışmalarımın yönünü tayin eden ve tamamlanmasında büyük emeği olan tez danışmanım Sayın Dr. Selim BAYRAKLI' ya, her zaman destek olan motivasyonumu sağlayan ailem ve arkadaşlarıma teşekkür ederim.

Ferhat KOÇER

Şubat 2019

ÖZ

İLİŞKİSEL VERİ TABANLARININ UYGULANDIĞI ARAÇ TAKİP SİSTEMİNİN BÜYÜK VERİYE UYARLANMASI

Ferhat KOÇER
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans
Danışman: Dr. Öğr. Üyesi Hüseyin Fehmi Selim BAYRAKLI
Maltepe Üniversitesi Fen Bilimleri Enstitüsü, 2019

Teknolojideki gelişmeler iletişimin ve ulaşımın gittikçe daha kolay hale gelmesini sağlamıştır. Bu gelişmeler sayesinde, gerçekleşen olayların bulundukları coğrafyadan dünyaya hızlı bir şekilde yayılması kolaylaşmıştır. Gelişen teknolojiyle filo yönetim sistemlerinin her geçen gün artması sadece bu alanda bile bilginin kontrolünde ve işlenmesinde büyük zorluklara yol açmaktadır. Küçük ölçekli işletmelerden büyük ölçekli işletmelere kadar birçok firma araç takip hizmeti vermektedir. Ülkemizde ve dünyada her geçen gün araç takip sistemlerine talep artmaktadır. Araç takip uygulamaları kapsamında düşündüğümüzde, farklı tiplerde gelen verilerin gerçek zamanlı olarak alınması ayrıştırıp anlamlı hale getirilmesi gerekmektedir. Bu tezde büyük veriyi kullanarak verilerin gerçek zamanlı olarak alınması işlenerek anlamlı hale getirilmesi sağlanmıştır. Ayrıca verilerin geçmişe dönük raporlanması ve izlenmesi için tutarlı, dağıtık, dayanıklı ve ölçeklenebilir bir uygulama örneği büyük veri teknikleriyle geliştirilmiştir. Geliştirilen sistemin denenmesi için gerçek araçlardan oluşan cihazların sisteme yönlendirilmesi sağlanmıştır. Mevcut ilişkisel veri tabanları kullanan klasik yaklaşımlı araç takip sistemlerinin yetenekleri ile karşılaştırıldığında, önerilen sistemin; yeni yaklaşımlar ile çözümler sunması, mimarisinin gerçek zamanlı proaktif yapıda olması, hataya dayanıklı, ölçeklenebilir ve dağıtık sisteme sahip olması sağlanmıştır.

Anahtar Sözcükler: 1. Büyük Veri; 2. Araç Takip Sistemi; 3. Gerçek Zamanlı Veri İşleme; 4. Apache Spark; 5. Apache Cassandra; 6. Apache ActiveMQ

ABSTRACT

BIG DATA ADAPTATION OF THE VEHICLE TRACKING SYSTEM APPLIED BY RELATIONAL DATABASES

Ferhat KOÇER
Master Thesis
Computer Engineering Department
Computer Engineering (With Thesis)
Thesis Advisor: Dr. Öğr. Üyesi Hüseyin Fehmi Selim BAYRAKLI
Maltepe University Graduate School of Science Engineering, 2019

Developments in technology have made communication and transportation easier. These developments have increased the fact that the events taking place have independent global effects from the regions. The increasing number of fleet management systems with the developing technology leads to great difficulties in the control and processing of information even in this field. From small-scale enterprises to large-scale enterprises, many companies provide vehicle tracking services.In the context of vehicle tracking applications, it is necessary to differentiate the data from different types in real time and to make them meaningful. Using big data in this thesis; In order to obtain data in real time, to make it meaningful by processing, to provide historical reporting and monitoring of data, a consistent, distributed, durable and scalable application example has been developed with big data techniques. In order to test the developed system, it is ensured that the devices consisting of real vehicles are directed to the system.Compared to the capabilities of the classical approach vehicle tracking systems using existing relational databases, the proposed system; providing solutions with new approaches, real-time proactive architecture, fault-tolerant, scalable, distributed and early warning system.

Keywords: 1. Big Data; 2. Vehicle Tracking System; 3. Real Time Data Processing; 4. Apache Spark; 5. Apache Cassandra; 6. Apache ActiveMQ

İÇİNDEKİLER

ÇİZELGELER LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR

Kısaltma	Türkçe	İngilizce
AMQP	Gelişmiş Mesaj Kuyruğu Protokolü	Advanced Message Queuing Protocol
API	Uygulama Programlama Arayüzü	Application Programming Interface
ATS	Araç Takip Sistemi	Vehicle Tracking System
ATS	Araç Takip Sistemi	Vehicle Tracking System
CQL	Cassandra Sorgu Dili	Cassandra Query Language
GNSS	Global Yön Uydu Sistemleri	Global Navigation Satellite Systems
GPS	Küresel Konumlama Sistemi	Global Positioning System
HDFS	Hadoop Dağıtılmış Dosya Sistemi	Hadoop Distributed File System
HTTP	Hiper Metin Transfer Protokolü	Hyper Text Transfer Protocol
IDC	Uluslararası Veri Kuruluşu	International Data Corporation
IMEI	Uluslararası Mobil Cihaz Kodu	International Mobile Equipment Identity
IOT	Nesnelerin İnterneti	Internet of Thing
IP	İnternet Protokolü	Internet Protocol
JMS	Java Mesajlaşma Servisi	Java Messaging Service
MQTT	Mesaj Kuyruğu Telemetri Tranferi	Message Queuing Telemetry Transport
PC	Kişisel Bilgisayar	Personel Computer

RAM	Rastgele Erişimli Bellek	Random Access Memory
REST	Temsili Durum Transferi	Representational State Transfer
RFID	Radyo Frekansı ile Tanımlama	Radio Frequency Identification
SQL	Yapısal Sorgulama Dili	Structured Query Language
STOMP	Basit Metin Odaklı Mesaj Protokolü	Simple Text-Orientated Messaging Protocol
TCP	Geçiş Kontrol Protokolü	Transmission Control Protocol
TİD	Taşıt İzleme Donanımı	Vehicle Tracking Device

ÖZGEÇMİŞ

Ferhat KOÇER

Bilgisayar Mühendisliği Anabilim Dalı

Eğitim

Derece	*Yıl*	*Üniversite, Enstitü, Anabilim/Anasanat Dalı*
Ls.	2013	Süleyman Demirel Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
Lise	2009	Gebze Cumhuriyet Lisesi

İş/İstihdam

Yıl	*Görev*
2014 -	Yazılım Mühendisi - Armoli Teknoloji A.Ş
2013- 14	Yazılım Mühendisi - Labirento Digital Solutions

Kişisel Bilgiler

Doğum yeri ve yılı	: Afşin/1991	Cinsiyet: Erkek
Yabancı diller	: İngilizce	
GSM / e-posta	: +90 (505) 692 9125 / ferhatkocer25@gmail.com	

BÖLÜM 1. GİRİŞ

Günümüzde teknolojinin giderek gelişmekte olması yeni alt dalların oluşumunu arttırmış ve birçok alanda yenilikçi çözümlerin ortaya çıkması bu yenilikçi çözümlerin her alana uyarlanması adeta bir ihtiyaç haline dönüşmüştür. Kullanıcıların veriye hızlı, güvenilir, kolay ve düşük maliyetli bir şekilde ulaşması teknoloji yarışında olan bütün kuruluşlar için amaç olmuştur. Teknolojinin gelişmesiyle hayatımıza giren nesnelerde aynı oranda ivmeli bir şekilde artmıştır. Bu teknolojik donanım ve yazılımların artması Nesnelerin Internet'i (IOT) denilen bir kavramı hayatımıza sokmuştur. IOT'yi tanımlarsak internet üzerinden birbirleriyle bağlı neslerin haberleşerek ihtiyaca göre giriş ve çıkış birimlerinin uzaktan kontrol edilebildiği veya programlanabildiği ve geçmiş hareketlerinin kayıt altına alınıp izlenebildiği otomasyon sistemleridir [1]. Akıllı ev sistemleri ve Küresel Konumlama Sistemi (GPS) tabanlı araç takip sistemleri buna örnek olarak verilebilir. Bütün bu iletişim ağı nesnelerin Internet'e çıkmasının amaçlamıştır. Nesnelerin internet üzerinde aktifliğinin hızla artması başka problemler ortaya çıkarmıştır. Bu kadar yoğun verinin işlenme maliyeti bir problem oluşturmuş ve büyük veri kavramını hayatımıza sokmuştur. Makinelerin sürekli veri ürettiği böyle bir ekosistemde büyük verinin oluşması kaçınılmaz bir durum olarak karşımıza çıkabilmektedir. Mevcut klasik veri tabanları ile kurulan sistemler maliyetli ve yeteneklerinin sınırlı olması nedeniyle, büyük veri sistemleri alternatif bir çözüm olarak ortaya çıkmıştır.

Araçların takip edilmesi ve çevre birimlerinin bir ana merkezden izlenip raporlanabilmesi gibi büyük problemlerin çözümüne yönelik veri işlemlerinde;

- gerçek zamanlı olarak akan veri üzerinde, düşük gecikme ile verilerin anlamlı hale getirilmesi, düzenlenmesi ve temizlenmesi,
- veri giriş ve çıkışlarının hızlı olması,
- kullanıcılara gerçek zamanlı bilgi ve alarm verilmesi, geçmişe dönük verilerin raporlanması kesinlikle olması gereken özelliklerdir.

Bir arada çalışan farklı disiplindeki veri üreticileri, değişken veri formatlarını ve değişken zaman aralıklarında farklı büyüklükteki verileri içerisinde barındırabilmektedir. Ancak büyük hacimli verileri depolamak, hızlı akan verileri analiz edip yapılandırmak günümüzde çoğunlukla kullanılan veritabanı yönetim sistemleri ve mimarilerle tasarlanması oldukça zordur. Büyük veri verinin, ekonomik, anlamlandırılabilir, ölçeklenebilir bir biçimde depolanması ve okunması olarak açıklanabilir [2].

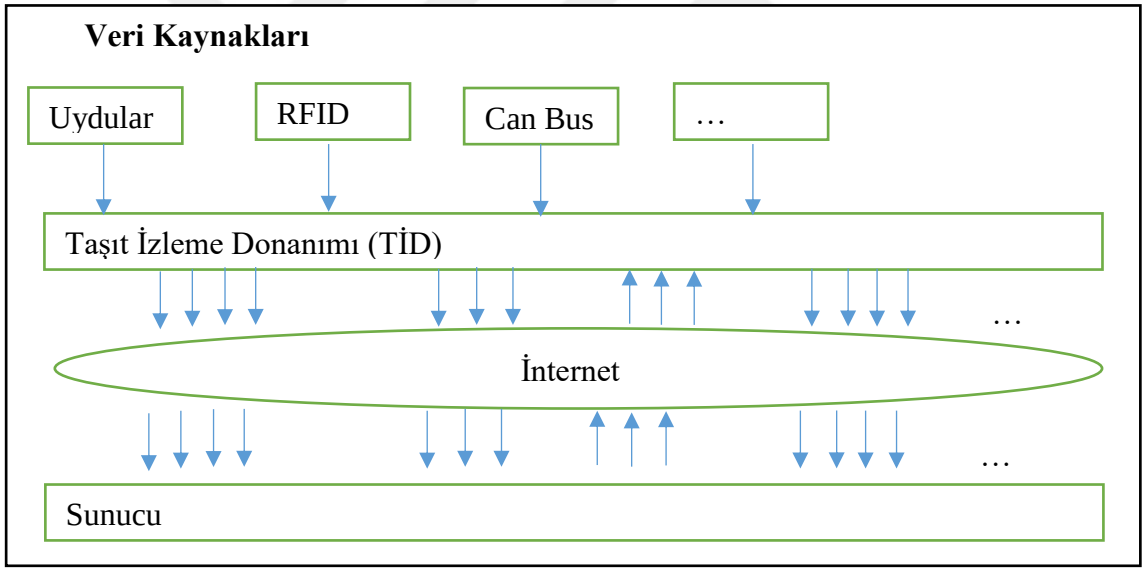
Araç takip sistemleri, otomobil, otobüs, kamyonet, motorsiklet, tekne gibi birçok alanda insanlara hizmet veren taşıtların bir elektronik cihaz aracılığıyla konumlarının izlenmesine olanak veren sistemlerdir. Takip cihazları enlem, boylam, zaman, hız ve yön bilgilerini ilgili uydu sistemlerinden alır ve konum bilgisini elde ederler bu konum bilgileri ile eğer varsa ivme ölçer, sürücü tanıma birimi için kullanılan Radyo Frekansı ile Tanımlama (RFID) kartları, panik butonu, araç içi navigasyon gibi harici cihaz ve sensörlerden aldıkları verileri birleştirerek sunucu tarafına TCP/IP protokolü ile gönderen cihazlardır.

Araç takip sistemleri, son yıllarda popülerliğini arttırmıştır. Araç Takip Sistemleri (ATS) genellikle Uydu Konum Belirleme Sistemleri (Global Navigation Satellite Systems-GNSS) tabanlı konum, hız, mesafe belirleme üzerine çalışan sistemlerdir. GNSS arama-kurtarma, hedef bulma, görüşün kısıtlı olduğu alanlarda ölçüm yapma, uçakların iniş kalkış yaptığı alanlarda, araç takip sistemlerinde, turizmde, tarım ve hayvancılıkta, spor, güvenlik, jeolojik teknonik fay hareketlerinin takibinde gibi birçok askeri ve sivil alanda kullanılmaktadır [3]. Bu tez çalışmasında araç takip sistemleri ile ilgili yapılan tez çalışmaları incelenmiş, bu sistemlerin üzerinde çalışan kişisel bilgisayar tabanlı navigasyon sistemlerinin alt yapısı, güncel teknolojiler araştırılıp uygun olanlar seçilerek sistem büyük veri ile tasarlanmıştır.

ATS lojistik şirketleri, araç veya filo kiralama firmaları, kamu kurum ve kuruluşları tarafından tercih edilmektedir. Bu kurumlar güvenlik, araç filo yönetimi, yakıt güvenliği, sürücü performans analizi, adli vakalar gibi bir çok alanda kullanılmaktadır. ATS ile aracın konumu, hızı, yönü, gittiği güzergah tükettiği yakıt miktarı, aracı kullanım şekli, hız limitlerine uyup uymadığının takibi cihazların uydulardan aldığı bilgiler ve

dahili sensörler ile takip edilebilmektedir. Bu sayede olası kazaların önüne geçilmesi, araçların daha iyi organize edilerek verimliliğin arttırılması, yakıt hırsızlığın önüne geçilmesi, araç hırsızlığında aracın kolayca bulunmasında kullanılabilmektedir.

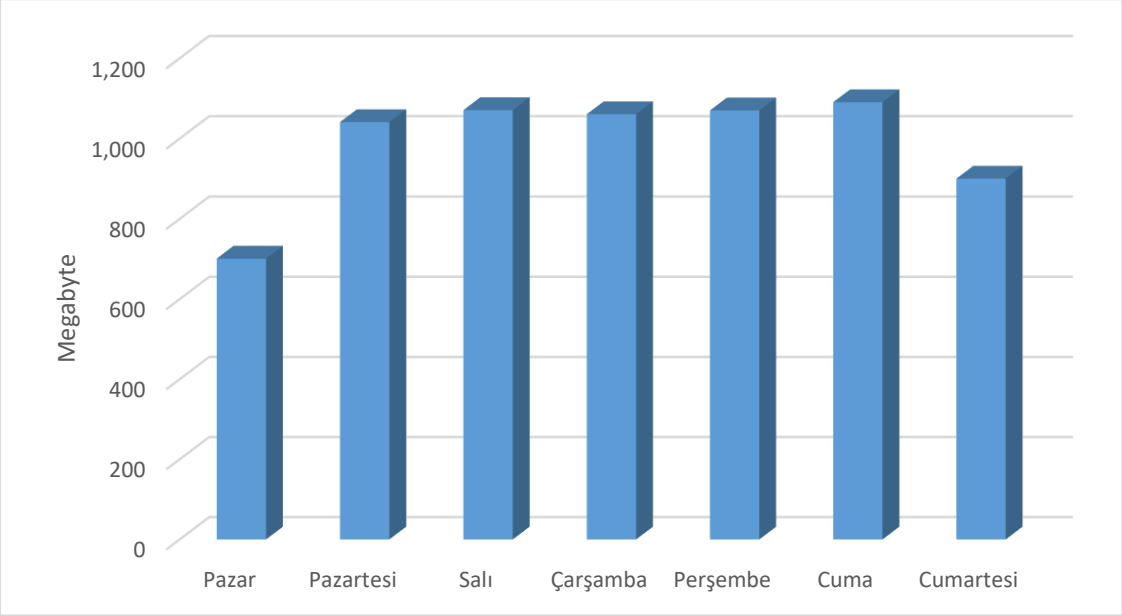
Şekil 1.1 ‘de görüldüğü üzere takip cihazları ile sunucu arasında çift yönlü haberleşme yapılmaktadır. Bu haberleşme cihazdan sunucuya veya sunucudan cihaza şeklindedir. Ayrıca harici sensörlerin bağlı olduğu cihazdan takip cihazına, takip cihazından sunucuya olabildiği gibi sunucudan cihaza cihazdan harici cihazdaki sensöre şeklinde veri iletimi olabilmektedir. Dolayısıyla araç takip sistemleri aslında birçok nesnenin birlikte haberleşebildiği bir ağ yapısına dönebilmektedir. Bu veriler birçok modülden toparlanır, sunucuya gönderilir. Sunucu tarafında veriler sürekli olarak günün 24 saati işlenir. ATS’ de çalışma zamanının (uptime) %100’e yakın olması beklenmektedir.



Şekil 1.1. Veri kaynakları ile sunucular arası iletişim

İlişkisel veri tabanları kullanılan, ölçeklenebilir tasarlanmayan bir ATS için sürdürülebilirlik ve maliyet artışı kaçınılmazdır. Eğer yapı iyi tasarlanmaz ise sistem için kaynakların tamamen tükenmesi veya kaynakların verimsiz kullanılması beklenilen bir durumdur.

Şekil 1.2 ‘de görüldüğü üzere araçların yoğun kullanımlarında ürettikleri veri miktarları aynı oranda artmaktadır. Bunun sebebi takip cihazları araçların kontak durumlarına göre programlanmasından kaynaklanmaktadır. Örneğin, araç hareketliyken (kontak açık) veri gönderim periyodu 3 sn ayarlanırken park halindeyken (kontak kapalı) 2 saat veya duruma göre 3 saat olarak ayarlanabilmektedir. Hatta enerji tasarrufu sağlanması amacıyla cihazlar kontak kapalıyken uyku moduna geçirilerek tamamen veri gönderimi de kesilebilmektedir. Bu şekilde tavsiye edilen periyot ayarlamalarıyla gereksiz veri tekrarının önüne geçilerek hem iletişim maliyeti düşürülmeye çalışılır, hemde sunucu tarafındaki kaynakların boşa kullanımı olabildiğince optimize edilmeye çalışılır. Sürekli tekrar eden verinin engellenmesi için alternatif çözümlerde bulunmaktadır. Bu çözümlerden birtanesi, konum uydularından alınan yön ve hesaplanan mesafe bilgilerine göre aracın programlanmasıdır. Örneğin, bir araç 25 derecelik yön değişiminde veya 100 metre mesafe gittiğinde konum göndersin şeklinde programlanırsa araç rölantide beklerken gereksiz veri gönderiminin önüne geçilinebilir ve bu sayede haritalar üzerinde yolu daha iyi takip eden bir rota çizgisi oluşturulabilmektedir.



Şekil 1.2. 2000 adet cihazın günlük veri miktarları

Takip cihazlarından gelen veri yoğunluğu Şekil 1.2’deki grafikte göründüğü gibi çalışma günlerinde hafta sonuna göre nispeten biraz daha fazla olmaktadır. Benzer şekilde günlük veri trafiğinin yoğun olduğu zamanlar sabah 07:00 ile 08:00, akşam 16:00 ile 19:00 arası olmaktadır. Bu sonuçlar veri miktarlarının genel olarak mesai saatleri ile

bağlantılı olduğunu göstermektedir. Dolayısıyla tasarlanacak olan sistemin gerçek zamanlı veri akışının en yoğun olduğu zamanları kaldırabilmesi gerekmektedir. Sistemde bulunan araç kapasitesinin de artacağı varsayılarak sistemin tasarlanması gerekmektedir.

Yoğun işlem yükünde olan ve sürekli büyüyen bir sistemin ölçeklenebilir olarak tasarlanması önemlidir. Mevcut sunucu sistemi tam kapasiteye ulaştığında sunucu kapasitesini arttırmanın iki ana yolu vardır. Bu yollar yatay ölçeklendirme (horizontal scaling) ve dikey (vertical scaling) ölçeklendirmedir [4].

1.1 Dikey Ölçeklendirme

Dikey ölçeklendirme ek performans değerlerinin tek sunucuda geliştirilemesi anlamına gelir. Bunun için bir sunucuya ek olarak RAM takılması, daha güçlü bir CPU veya birden fazla CPU takılması, daha yüksek kapasiteli ve performanslı Hard Disk takılması örnek olarak verilebilir [4].

1.2 Yatay Ölçeklendirme

Yatay ölçeklendirme daha fazla sunucunun alt yapıya eklenerek ölçeklenebilirliğin sağlanmasıdır.

Yatay ölçekleme dikey ölçeklemeden daha ucuz olabilir, ancak daha dikkatli bir planlama gerektirir. Yatay ölçeklemede eğer yazılım iyi tasarlanmışsa çalıştırılabilecek örnekler neredeyse sınırsızdır. Eğer mimari izin veriyorsa istendiği zaman küçültülüp büyütülmesi sayesinde maliyet tasarrufu da yapılabilmektedir. Ayrıca sunuculardan birisi arızalanırsa sistem yinede çalışmaya devam edebilmektedir [4].

Verileri tutmak ve daha sonra kullanıcı arayüzünde raporlayarak anlamlı veri haline getimek için genel olarak iki adet veri tabanı çözümü vardır. İlişkisel veri tabanı ve ilişkisel olmayan (NoSQL) veritabanı çözümüdür.

1.3 İlişkisel ve İlişkisel Olmayan Veri Tabanları

1.3.1 İlişkisel Veri Tabanları

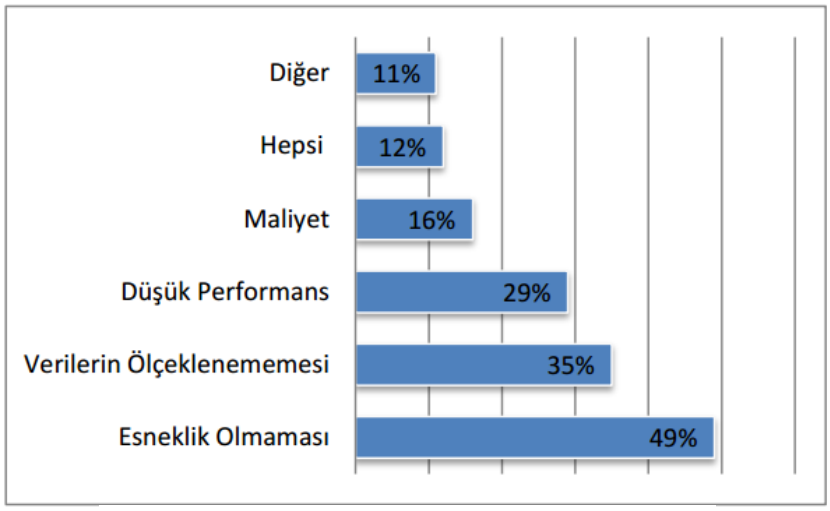
İlişkisel veri tabanları günümüzde en yaygın kullanılan veri tabanı sistemlerindendir. Satır ve sütunların oluşturduğu tablo yapılarıdır. Bu tablo yapıları arasında ilişki bulunur. Dolayısıyla ilişkisel olabilmesi için en az iki tablonun var olması ve bu tablolar arasında ilintili verilerin yer alması gerekmektedir [5].

Klasik ilişkisel veri tabanlarında olması beklenen temel özellikler (ACID) [6]:

- Bölünmezlik(Atomicity); yapının her türlü istenmeyen duruma karşı bölünmez yapıda olması garanti edilmelidir.
- Tutarlılık(Consistency); veri tabanına yazılan verilerin tanımlı kurallara uygun olarak kayıt altına alınmalıdır.
- Yalıtım(Isolation); aynı veri üzerinde yapılan işlemler birbirlerinden yalıtılmış olarak işlem görmelidir.
- Dayanıklılık(Durability); yapılan işlemlerde hata olması durumunda son verinin korunması sağlanmalıdır.

1.3.2 İlişkisel Olmayan (NoSQL) Veri Tabanları

İlişkisel olmayan (NoSQL) veri tabanı ilk ortaya 1998 yılında çıkmıştır. Carlo Strozzi tarafından başlatılan bir kavramdır. İlişkisel olmayan veri tabanları ilişkisel veri tabanlarına alternatif bir çözümdür. Yatay ölçeklendirilen veri tabanı sistemidir. İlişkisel veri tabanı kullanıcılarının neden ilişkisel olmayan veri tabanına geçmek istemelerinin nedenlari Şekil 1.3 üzerinde gösterilmiştir [5].



Şekil 1.3. Neden NoSQL gerekli ? [5]

Bu tezde yapılması planlanan uydu ile konum belirleme tabanlı araç takip sistemlerinin klasik ilişkisel veri tabanı çözümleri yerine büyük veri teknikleri kullanılarak daha esnek ve güçlü bir yapı üzerinde tasarlanmasıdır. Proje yapılırken kullanılan teknolojilere değinilmiş, bu teknojilerin projenin hangi aşamasında kullanıldığı belirtilmiştir. Klasik veri tabanlarında karşılaşılan problemler ve bu problemlerin büyük veri tarafındaki çözümlerine değinilmiştir. Büyük verinin avantajlarından ve dezavantajlarından bahsedilmiştir. Günümüzde birçok firma gerek kiralama gerek satın alma yoluyla elde ettiği araçların takibini yapmak istemektedir. Bu sayede sürüş optimizasyonu sağlanarak şirket içi giderlerin araç veya sürücü kaynaklı olmasının önüne geçilmesi amaçlanmaktadır. Küresel ölçekte düşünüldüğünde iyi tasarlanmış bir ATS yakıt tüketimini düşürerek ve araç ömrünü uzatarak çevre kirilliğinin araç kaynaklı olmasının büyük ölçüde önüne geçilebilinir. Araç takibinde hedeflenen temel amaçlar şunlardır;

- yakıt tüketimini en düşük seviyede tutmak,
- kaza oranını düşürmek,
- sürücüye gerçek zamanlı iş vererek zaman tasarrufu sağlamak,
- araçların yıpranmalarını düşürmek,
- şüpheli durumlarda aracı uzaktan kontrol edebilme ve zamanında müdahele edebilme,
- personelin çalışma sürelerinin hesaplanması
- adli vakalarda gerekli konum bilgilerini sunabilmek

- kullanıcı ile etkileşimi daha fazla, proaktif sistem tasarlanması

Bu özelliklerin iyileştirilmesi bazılarında direkt bazılarında dolaylı olarak küresel ve bölgesel anlamda hem ekonomiye, sürüş güvenliğine, hemde çevre kirliliğinin önüne geçmede olumlu etki etmektedir. Bu açıdan bakıldığında araç takip sektörünün gelişmesi birçok açıdan katkı sağlayacağı öngörülmektedir. Araç üreticileri bu sistemleri yavaş yavaş üretim standardı haline getirmektedir.

Bu tez çalışmasında, Büyük Veri teknolojisi kullanılarak takip cihazından verinin ilk çıktığı andan kullanıcı arayüzüne kadar olan bir araç takip sistemi tasarımı geliştirilmesi amaçlanmıştır. Gerekli teknolojiler araştırılmış ve bilgiler hakkında açıklamalar yapılmıştır.

Bu çalışmada;

- veri gönderiminde ve dağıtımında Apache ActiveMQ,
- verilerin gerçek zamanlı işlenmesinde Apache Spark,
- verilerin depolanmasında Apache Cassandra,
- verilerin işlendikten sonra kullanıcı arayüzüne yansıtılmasında NodeJS,
- Cassandra ile kıyaslama yapılabilmesi için MySQL veritabanı

kullanılmıştır. Yapılan araştırma ve geliştirme aşamalarında bu probleme benzer özellikteki problemlerin çözümüne olanak sağlayabilecek denenmiş bir örnek elde edilmiştir.

Günümüzde elektrikli araçlar giderek yaygınlaşmaktadır. Otomobil üreticileri, ürünlerini daha fazla geliştirebilmek için satış sonrasında dijitalleşmenin avantajlarını kullanarak ürünlerinden daha fazla ve anlık veri toplama ihtiyacı hissetmektedir. Araç takip sistemleri ile sürücülerinde sürüş karakteristiği ve yol durumları analiz edilerek kaza oranlarında azalma, yakıt tüketimini optimize etme, zaman kaybını önleme gibi hayati

öneme sahip konularda iyileştirme yapılabilmektedir. Araç takip sistemleri doğası gereği sürekli çevrimiçi çalışan gerçek zamanlı veri akışının çok yoğun olduğu sistemlerdir. Klasik çözümler ile yaklaşılan araç takip sistemlerinde belirli bir noktadan sonra tıkanıklık yaşanması mümkün olabilmektedir. Bu sistemlere entegre edilebilen cihaz tipleri teknolojinin gelişmesiyle sensörlerin gelişmesine, dolayısıyla etkileşim yöntemlerinin değişmesine neden olmaktadır. Dolayısıyla klasik yaklaşımlar ATS'nin veri trafiğini işleme konusunda yeni yaklaşımlara ihtiyaç duyabilmektedir. Bu noktada çözüm için yeni yaklaşımlarla esnek, dayanıklı, ölçeklendirilebilir olan büyük veri çözümü incelenmiştir.

Bu çalışmada yapılanlar:

- ATS ile ilgili geliştirilen çözüm için uygun teknikler araştırılıp detaylandırılmış, geliştirilen projede genel çözümün yapısına uygun metodolojiye yakın özel çözümler incelenmiş ve aralarından uygun olanlar belirlenmesi için yol haritası oluşturulmuş, bu tekniklerin avantaj ve dezavantajları açıklanmıştır.
- Yeni yaklaşımlarla ATS yapısı tasarlanmış, büyük verinin sisteme nasıl uyarlanabileceğinden bahsedilmiştir. Klasik ilişkisel veri tabanlarında yaşanabilecek zorluklar anlatılmıştır. Geliştirilen sistemde kullanılan araçlar sistemde rol aldığı görevleriyle belirtilmiştir.
- Tasarlanan yapıda elde edilen bulgulardan bahsedilmiş, bulgular hakkında yorum yapılmıştır. Klasik ilişkisel veri tabanı olan MySQL ile kolon bazlı dağıtık ilişkisel olmayan Apache Cassandra kıyaslanmış elde edilen sonuçlar açıklanmıştır.
- Geliştirilen proje sonuçları anlatılmış, yazılım geliştirilirken kullanılan yöntemler, karşılaşılan problemler ve problemlerin çözümleri ele alınmış, ayrıca projenin uygulanabileceği alanlardan bahsedilmiştir.

BÖLÜM 2. YÖNTEM

Bu tez çalışmasında farklı teknolojilerin birbirleriyle haberleşmesi için gerekli alt yapının kurulması sağlanmıştır. Bu haberleşme dili olarak çoğunlukla JSON (Javascript Object Notation) formatı seçilmiştir. JSON hafif bir veri değişim formatıdır. Okunup anlaşılması kolaydır. JSON tamamen programlama dillerinden ve platformlardan bağımsız bir veri tanımlama formatıdır [7]. JSON obje tabanlı yapıya sahiptir. Birçok programlama dili ile uyumludur. Ayrıca JSON alternatifi olan XML (Extensible Markup Language)'e göre daha hafif, daha esnek olmasından dolayı çoğunlukla tercih sebebidir. Araç takip cihazları ile sunucular arasında haberleşme dili için takip cihazının kendine özgü formatı kullanılmıştır. Bu formata özel yazılım geliştirilmiştir. İki yönlü haberleşme yapılmıştır. Bu tez çalışmasında bu haberleşmenin yapılabilmesi için literatürdeki çözümler hakkında bilgi verilmiş, iletişimin güvenliğinin nasıl sağlandığı olası aksaklıklarda alınabilecek önlem için çözüm önerileri incelenmiştir. Uygun görülen bir çözüm geliştirilen projede uygulanmıştır.

2.1 Büyük Veri

Her geçen gün teknoloji gelişmektedir ve bize yeni kolaylıklar sağlarken, bu kolaylıkların yanında da yeni zorluklar getirmektedir. Makinelerin akıllı olması beraberinde herşeyin Internet'e açılmasını getiriyor bu da veriyi depolama, yayınlama, ayrıştırma gibi problemleri ortaya çıkarıyor.

Geçmiş zamanlarda ihtiyaçlar doğrultusunda büyük veriye gerek duyulmuyordu. Günümüzde verilerin artması yeni bir probleme sebep olmuş ve büyük veriyle bu problem yavaş yavaş çözülmeye başlanmıştır. Büyük veri genel olarak 5V denilen beş ana bileşimden oluşmaktadır.

- *Volume (Hacim)*: International Data Corporation'nin (IDC) 2012 yılında yapılan bir araştırmaya göre 2020'ye kadar dünya üzerindeki her kadın, erkek ve çocuk için 5.200 GB toplamda 40 zetta bayttan fazla veri olacağı tahmin ediliyor. Bu veri miktarı dünyadaki tüm kum tanelerinin 57 katı kadar olduğu tahmin edilmektedir. 2020 yılına kadar olan her 2 yılda bir, veri büyüklüğü 2

katına çıkacaktır [8]. Bu şekilde büyüyen bir verinin depolanabiliyor olması gerekmektedir.

- *Variety (Çeşitlilik)*: Donanımların küçülmesi her yerde kullanılabilecek yeni cihazların ortaya çıkmasına sebep olmuş dolayısıyla yeni veri protokollerini ortaya çıkarmıştır. Bu da çeşitliliği arttırmıştır. Farklı tip verilerden beslenen sistemlerin bütünlüğü sağlanabiliyor olmalıdır.
- *Velocity (Hız)*: Hacimsel olarak büyüyen ve çeşitliliği artan verinin klasik ilişkisel veri tabanlarıyla sorgulanması zor ve maliyetlidir. Klasik yöntemler hacim ve çeşitliliğin yüksek olduğu verileri sorgulama konusunda iyi bir çözüm sunamazlar. Büyük verinin olduğu projelerde platformun hızlı olması beklenmektedir.
- *Veracity (Doğruluk)*: Okunan verilerin veritabanı sisteminde ilk yazıldığı haldeki gibi kalması ve bu verilerin dışardan müdahale edilebilecek hiçbir açığının olmaması için gerekli güvenlik önlemlerinin alınmış olması beklenmektedir.
- *Value (Değer)*: En önemli özelliktir, diğer özelliklerin ana hedefi bu özelliği desteklemektir. Elde edilen verilerin karar vermede ve sürekli güncel kalmasını sağlacak bir alt yapı gerekir.

Günümüzde yayılabilen dağıtık veri işleme ve depolama sistemlerine ihtiyaç duyulmaktır, veri boyununun üstsel olarak büyümesi bu şekilde bir çözüm oluşturmuştur.

Büyük veri ile oluşan yığın içerisindeki işe yarayacak anlamlı veriler elde edip bunları analiz edebilmek için, mevcut çözümlerin üzerinde yeni yaklaşımlar gerekmektedir. Bu yaklaşımların birçoğu, bilgisayar bilimleri, ekonomi, matematik ve istatistik gibi birçok alan ile kesişmiştir.

Büyük veriyi anlamlı veriler haline getirebilmek kullanılan araçlar üç guruba ayrılır; yığın işleme (batch processing tools), akış işleme (stream processing tools), interaktif analiz (interactive analysis tools) araçlarıdır.

2.1.1 Büyük Veri İşleme Araçları

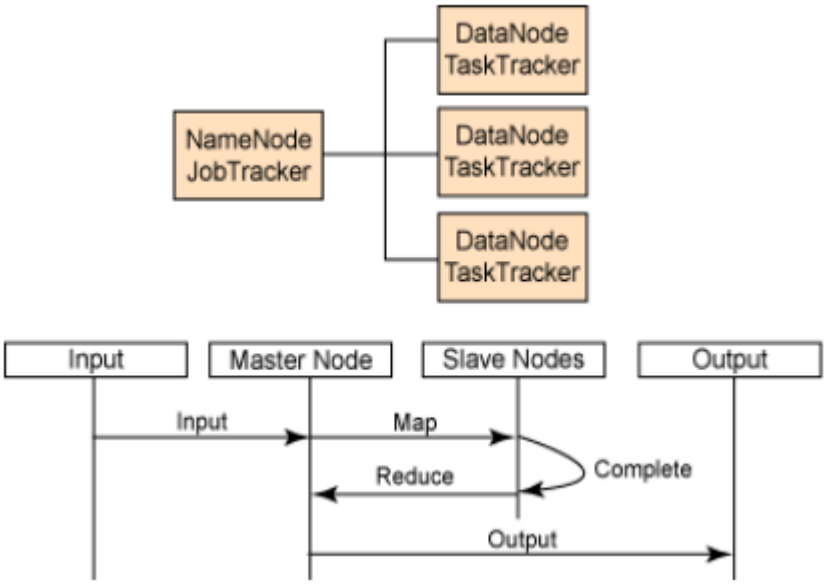
Popüler olarak tercih edilen büyük verileri işleyebilecek araçlardan birtanesi Apache Hadoop'dur. Birçok Büyük Veri uygulamaları Apache Hadoop temelini kullanır.

Apache Hadoop ve Map/Reduce

Apache Hadoop petabaytlarca veriyi işlemeye izin veren bir kütüphanedir. Çözümlenen verilerin yedeklerini elinde tutarak çalışan bir yapıya sahiptir. Herhangi bir düğümde bir hata oluşursa başka bir düğümde bulunan hatasız kopyasını tekrar kopyalar, verilerin güvenliğini bu şekilde sağlar [9].

Hadoop temelinde Hadoop Distrubuted File System (HDFS) [10] diye adlandırılan bir dağıtık dosya sistemi bulundurmaktadır.

Map/Reduce, Google'ın geliştirdiği bir büyük veri bloklarını işlemede kullanılan bir yazılım modelidir. Map/Reduce üç temel bölümden oluşur: Map fonsiyonu, Reduce fonksiyonu ve Ana fonksiyon. Map fonksiyonu işlevinde ana düğüm veri girişlerini alır, bunları küçük parçalarda işlemlere ayırır işçi düğümlerine dağıtır. İşçiler işlemi gerçekleştirir sonucu ana düğüme gönderir. Reduce işleminde ana düğüm parça işlemlerden aldığı cevapları Şekil 2.1 de görüldüğü gibi birleştirir. Ana düğüm iş listesini oluşturur ve alt işçi düğümlere kölelere görev dağırım yapar. Köleler gerekli işleri yerine getirir. İş akışı devam ederken bir işçi düğümden kaynaklı oluşan aksama durumunda ana düğüm görevi farklı düğümlere devreder.



Şekil 2.1 Hadoop kümesi ve MapReduce [11]

Apache Mahout

Apache Mahout Hadoop temelli Map/Reduce üzerinde çalışır. Büyük veri analizi ve ölçeklenebilir makine öğrenmesi yapılabilmesi amaçlanarak geliştirilen bir açık kaynak kodlu araçtır. Ölçeklenebilir bir yapay zeka algoritması yapmak beraberinde birçok problem getirmektedir. Bu problemlerin kolaylaştırılması için içerisinde çeşitli araçlar barındırmaktadır. Apache Mahout;

- 2008 yılında Sean Owen tarafından SourceForge'da başlatılan bir projedir,
- Map/Reduce ile kümeleme implementasyonları; k-Means [12], fuzzy k-Means [13], Canopy, Dirichlet ve Ortamala Kaydırma (Mean Shift),
- evrimsel programlama için dağıtılmış esnek fonksiyon kapasitesi ,
- matris ve vektör kütüphaneleri,

başta gelen özellikleridir [14].

Apache Spark

Apache Spark oldukça hızlıdır, hem yığın işleme hem de anlık akış işleme özelliğine sahiptir. Map/Reduce modeli üzerinde kurulmuş böl ve yönet mantığında çalışmaktadır. Bellek üzerinde işlemleri yönetebilir, bu sayede çok hızlıdır. Master slave yapısı üzerine kurulmuştur. Spark sıkça Hadoop'un HDFS veri saklama dosya sistemi üzerinde çalışır. Ancak diğer popüler veri saklama sistemleriyle de entegrasyonu eşit düzeyde sağlamıştır. Bunlar HBase [15], Cassandra, MapR-DB, MongoDB [16] ve Amazon S3'dür. Spark'ı seçme nedenleri olarak şu özellikler sayılabilir [17];

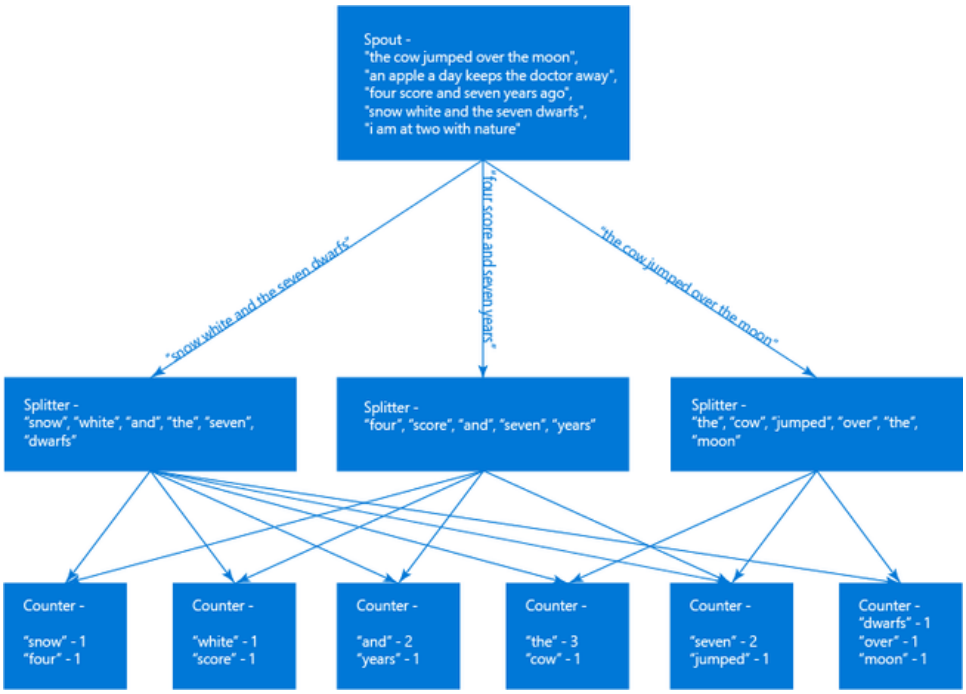
- Basitlik: Spark'ın kabiliyetlerine sunmuş olduğu zengin Uygulama Programlama Arayüzü (API) kümesi üzerinden erişilebilir , bunlar özellikle veri üzerinde interaktif işlemleri hızlı ve basit şekilde sağlayabilen ve ölçümlenebilir bir mimari şeklinde tasarlanmıştır. Veri bilimcilerin ve uygulama geliştiricilerin Spark'ı sahada hızlıca uygulamaları için API dokümantasyonu düzgün, anlaşılır ve kaliteli bir dile sahiptir.
- Hız: Spark'ın tasarımı bellek ve disk üzerinde çalışacak şekilde ve hız esas alınarak yapılmıştır. 2014'de yapılan Daytona Gray Sort yarışmasında Spark, SSD üzerinde tutulan 100 TB'lık veriyi sadece 23 dakikada sıralayarak birincilik ödülünü kazanmıştır. Düzenlenen bir önceki yarışmada Hadoop aynı veriyi 72 dakikada sıralayarak ödülün sahibi olmuştur. Yarışma statik veri kümelerinin işlenmesi üzerinedir. Fakat bellek içi interaktif sorguların bulunduğu bir ortamda Spark'ın Hadoop MapReduce'dan 100 kat daha hızlı olduğu görülmektedir.
- Destek: Spark bir çok programlama dilini desteklemektedir. Bunlar arasında Java [18], Python [19], R ve Scala bulunmaktadır. Hadoop'un HDFS veri depolama sistemiyle çoğu zaman bağlantılı olsa da Spark, Hadoop'la veya ötesinde çalışabilen önde gelen veri depolama çözümleriyle içten bir entegrasyon kurmuştur. Buna ek olarak, Spark topluluğu oldukça büyüktür, aktiftir ve uluslararası yapıda bir çok geliştiriciyi barındırır. Kurumsal destekçilerinin sayısı her geçen gün artmaktadır. Bunlar Databricks, IBM, ve Spark bazlı destek veren önde gelen Hadoop dağıtımlarıdır.

Apache Storm

Storm, sınırsız veri akışını işleme de tercih edilen dağıtık ve hata toleransı bulunan gerçek zamanlı veri işleme sistemidir. Storm; yığınları işlemek için yapılmış olup Hadoop'un aksine, asıl amacı gerçek zamanlı işlemler içindir. Bunların dışında entegrasyonu ve yönetmesi çok pratiktir; verinin tamamının işleneceğinin garantini verir. Storm düğümlerle çalışır, bu düğümlerinde saniyede milyonlarca veri işleyebilir. Bu sebeple anlık etkileşimli işletim sistemi, mantıksal analiz, devamlı hesaplama gibi benzer birden fazla sistemi içerir.

Storm temel yapıda Hadoop ile benzer özellikler göstermektedir. Ancak Storm mevcut görevlerinde alışılagelmiş olan Map/Reduce yerine topolojiler çalıştırır. Hadoop ise, benzer işlemler için geliştirilen sistemlerde farklı çözüm olan Map/Reduce görevlerini yerine getirir. Map/Reduce işlemi ile topolojiler birbirlerinden çok farklıdır.

Storm topolojileri döngüsel olmayan Şekil 2.2'deki gibi yönlü birden fazla bileşenden oluşur. Şekil 2.2, basit düzeydeki bir kelime sayılarının topolojilerde bulunan bileşenler arasında nasıl aktığını göstermektedir.

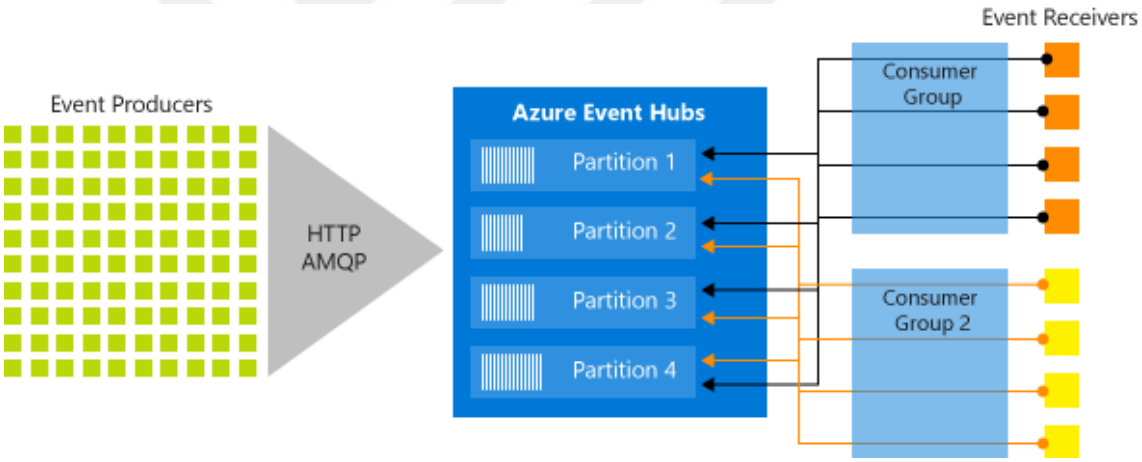


Şekil 2.2. Storm topolojileri [20]

- **Spout:** Bu bileşen veriyi bir topolojiye gönderir. Bu bileşenler topolojiye akış yayar.
- **Bolt:** Bu bileşen Spout veya boltlardan akan verileri kullanır [21]. Boltlar topolojiye tercihe bağlı akışlar yayabilir. Ayrıca HDFS veya Kafka [22] gibi veri depolama sistemlerine veri girdisinden de sorumludur.

Microsoft Azure Event Hubs

Event Hub Microsoft tarafından sağlanan saniye milyonlarca veriyi alıp içerisinde yayın yapma ve abone olma hizmetleri bulunduran ölçeklendilebilir sistemdir. Şekil 2.3'de görüldüğü gibi yayın yapılan veriler farklı sistem ve uygulamalar tarafından gerçek zamanlı kullanılmasını sağlar. HTTP (Hyper Text Transfer Protocol) protokünü kullanarak REST API ile bağlanıp kullanılabilir.



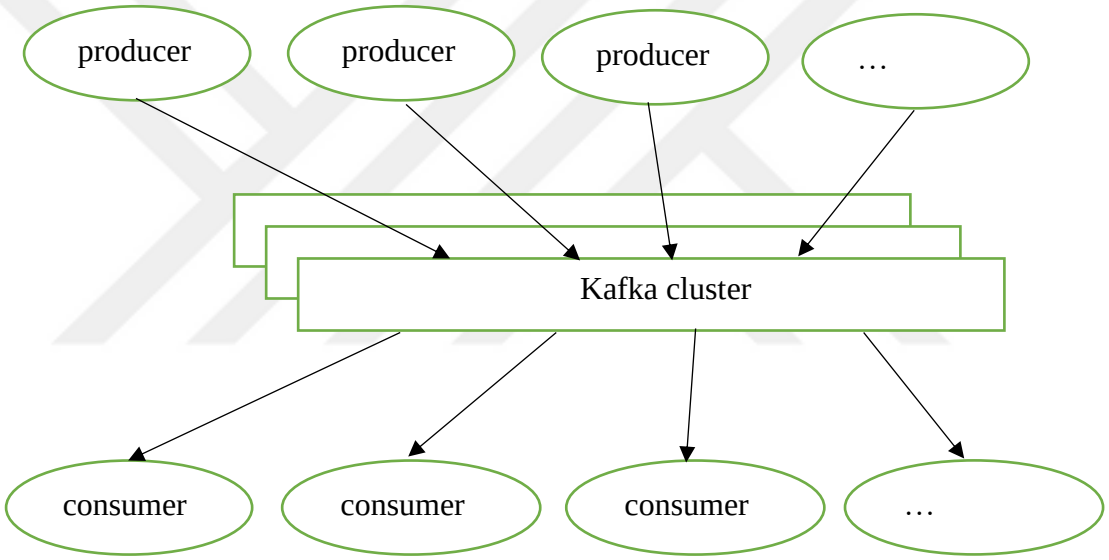
Şekil 2.3. Event HUB [23]

Apache Kafka

Apache Kafka gerçek zamanlı veri akışlarını sağlamak için yayınla ve yakala mekanizmasını kullanır. Uygulamalar arası veri iletimini sağlayan bir mesajlaşma sistemidir. Log kayıtlarına benzer veri tutan bir haberleşme servisidir [22]. Şekil 2.4'de görüldüğü gibi yayını yapılacak mesajlar konu (topic) üzerinden Kafka kümeleri ile yayınlanır. Verinin tüketimi konu üzerinden yapılır. Kafka'nın temel bileşenleri:

- **Kafka kümesi:** Birden fazla sunucu sistemi veya tek sunucunun bulunabildiği sistemlerdir. Verinin dağıtım ve iletim işlemleri yapılır kurulacak sisteme göre tasarlanır.
- **Broker:** Kafka kümelerinde bulunan her bir sunucu için verilen isimdir.
- **Producer:** Veri üreten, oluşturan kaynaklara verilen isimdir.
- **Consumer:** Verinin teslim edildiği alıcıdır.

Gönderilen veriler üretici ve alıcı arasında aynı konu içerisinde olursa ancak haberleşme sağlanabilir. Veri gönderimi ve alımı TCP üzerinde yapılır. Bu da gönderici ve alıcı alt yapılarının birbirinden bağımsız çalışmasına olanak tanır.

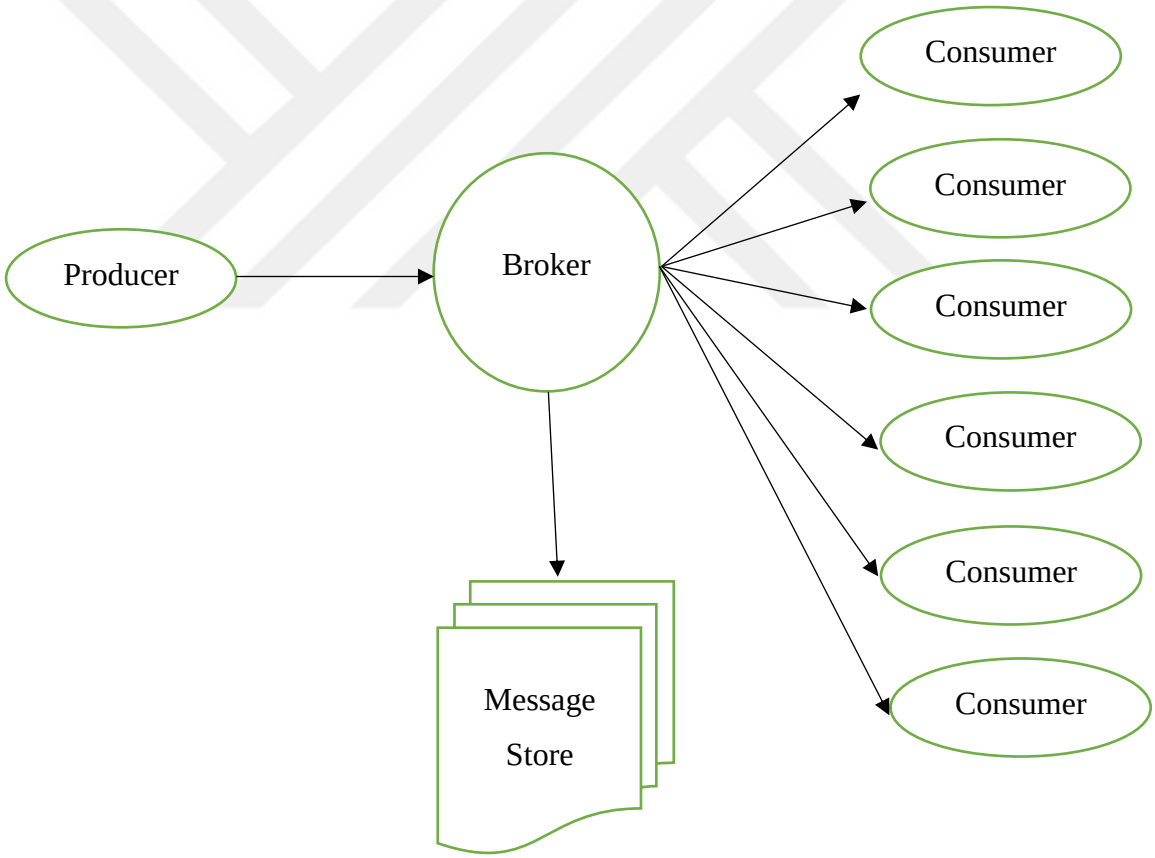


Şekil 2.4. Kafka mesajlaşma yapısı

Apache ActiveMQ

ActiveMQ popüler güçlü ve açık kaynak kodlu bir mesajlaşma sistemidir. ActiveMQ yayınla yakala mekanizmasıyla çalışır. Kafka ile benzerlik göstermektedir. Java ile JMS (Java Message Service), REST ve WebSocket arayüzü kullanılarak yazılmış bir mesaj broker sistemidir. AMQP [24], MQTT [25], OpenWire ve STOMP gibi birçok protokolü destekler. Bu protokoller ile ActiveMQ birçok uygulamada kullanılabilmesi için .NET, C, C++, Java, NodeJS, Python gibi birçok dilde kütühane ve örnekler bulunmaktadır.

Yapısında üretici (producer), tüketici (consumer) ve broker bulundurur. Üreticiler veri kaynağını sağlar ve broker'lara iletir. Brokerlar, veriyi sistem durumuna göre tüketicilere gönderir. Mesajlar kuyruk yapısında (queue) veya konular (topic) ile gönderilebilir. Akış kontrolü brokerda yapılır. İstenirse duruma bağlı olarak akış kontrolü kapatılabilir. Akış kontrolü açık iken üretici tüketiciden onay mesajını alarak veri gönderimi sağlar, böylece alıcı tarafında ön belleğin dolması işlemci performansı düşmesi gibi geçici durumlarda verinin brokerde bekletilmesi sağlanır. Kapalı olması durumunda brokerlar herhangi bir yavaş tüketiciye bağlı kalıp sistemin durmasının önüne geçer. Eğer hızlı bir alıcı varsa yavaşlamış veya tamamen bloke olmuş bir alıcıya mesaj iletimi yön değiştirerek hızlı tüketiciye veri gönderimiyle sistem çalışmaya devam eder. Şekil 2.5'de ActiveMQ'nun mesaj gönderim yapısı gösterilmiştir [26].



Şekil 2.5. ActiveMQ mesajlaşma yapısı

2.1.2 NoSQL

1998 yılında Carlo Stronzzi tarafından ortaya atılan bir kavramdır. Bu veri depolama sistemini tararlayan Carlo Stronzzi ilişkisel olmayan anlamında yani "No Relation" tarzında bir isim olması gerektiğini söyler. Sonrasında 2009 yılında Rackspace şirketinin çalışanlarından Eric Evens, açık kaynak kodlu dağıtık veritabanlarının konuşulduğu dönemde bu sistemleri isimlendirme amacındaydı.

NoSQL isimlendirmesi aslında yeni olan bu teknolojinin etkileyici bir sunumu olarak düşünülebilir. Diğer adlandırmalar çok tercih edilmemiştir, bunlardan bazıları NonRel, NoRDBMS'dir. NoSQL için "Not only SQL" yani "Sadece SQL değil" tabiri de kullanılmaktadır. Bunun sebebi SQL sorgularına alışmış geliştiricilerin aynı mantığı NoSQL veritabanlarına adapte edebilmesidir.

İlişkisel veritabanlarına alternatif bir çözüm olarak çıkan, yatay olarak ölçeklenebilen, dağıtık olarak verilerin depolanmasına imkan sağlayan ve popüler olarak kullanılan çözümler NoSQL yapısında olan veri yönetim sistemleridir. Bu veritabanı yönetim sistemlerinin en büyük avantajı veritabanı tasarlandıktan sonra bile yapılarında kolaylıkla ekleme çıkarma düzenleme yapılabilmesidir. Ayrıca veriler homojen olarak depolanmadığında bile esnek yapıları dolayısıyla kolaylık sağlar. NoSQL veritabanları genel olarak dört grupta toplanabilir.

2.1.2.1 Key-value kaydı yapan veri tabanları

Veri grubları bir anahtar ile bağlanarak kayıt altına alınır. Verinin içeriğinin ne olduğu veritabanı tarafından bilinmez. Kayıtlar key anahtarına bağlı olarak getirilir veya sisteme dahil edilir. Aşağıda örnek key-value veri depolama sistemlerinden bazıları listelenmiştir.

- Dynamo
- ArangoDB
- Berkeley DB
- MemcacheDB
- Aerospike
- Apache Ignite
- Coherence
- Hazelcast
- Redis
- Hibari
- CDB

Key-value kaydı yapan veritabanlarının avantaj ve dezavantajları:

Bu veritabanları tutulacak verinin sadece anahtarı üzerindeki işlemlerle ilgilenir, verinin içeriğiyle ilgilenmez. Tutulan veri içeriğinin şeması veriyi sisteme dahil eden kullanıcının sorumluluğundadır. Dolayısıyla basit bir arayüze sahiptir. Veritabanı veriden anlam çıkarma sorumluluğunu kullanıcı tarafına yükler. İçerik ile ilgili sorgulama veritabanının özellikleriyle yapılamaz sadece ekle, sil, getir metotları "Key" üzerinden yapılabilir. İçeriğin anlamlandırılması kullanıcı tarafında yapılır.

2.1.2.2 Döküman yapısı temel alınarak kayıt tutan veri tabanları

Doküman yapısı üzerine tasarlanan veritabanlarında veriler çoğunlukla JSON formatında kayıt altına alınır. Key-value veritabanlarına benzerlik gösterir. Bu veri tabanları; key-value veri çiftlerinin, toparlanarak versiyonlanmış ve yarı yapılandırılmış dokümanlarıdır. Aşağıda örnek döküman yapısı temel alınarak kayıt tutan veri depolama sistemlerinden bazıları listelenmiştir.

- MongoDB
- CouchDB
- OrientDB
- MarkLogic
- IBM Cloudant
- BaseX
- Counchbase Server
- Apache CouchDB
- Azure Cosmos DB

Döküman yapısı temel alınarak kayıt tutan veri tabanlarının avantaj ve dezavantajları:

Uygulamayı geliştiren geliştirici tutulacak veri dökümanlarının tipini belirleyebilir, isterse direkt erişim sağlayabilir bundan dolayı esnek bir veri depolama kayıt sistemi oluşur. Veriler kolay bir şekilde yatay ölçeklenebilir, kopyaları oluşturulabilir, doküman tipleri değiştirilebilir. Bu sistemlerle tasarlanan veritabanları içerik ile ilgili bilgi sahibidir. Veritabanı içerikle ilgili işlem yapabilir indeksleme yapabilir bu da kayıdın tutulmasında maliyet yaratmaktadır. Doküman verilerinin büyük olması durumunda istenirse veriler binary tipinde veri kaydı tutabilir [27].

2.1.2.3 Kolon yapısı temel alınarak kayıt tutan veri tabanları

Çoğunlukla dağıtık sistemler için tasarlanmış veri depolama sistemleridir. Temel olarak key-value veri depolama sistemleri ile ilişkisel veri tabanlarının mantıksal olarak birleşimi sağlanarak tasarlanmıştır. Key-value veri tabanları ile ilişkisel veri tabanları sistemlerinin mümkün olduğunca avantajları alınması hedeflenmiştir. Key-value birbiriyle ilişkilidir, key, value'nun kümede bulunan yerini göstermektedir. Aşağıda örnek kolon yapısı temel alınarak kayıt tutan veri depolama sistemlerinden bazıları listelenmiştir.

- Apache Cassandra
- CouchDB
- Druid
- HBase
- Qbase
- Amazon SimpleDB
- Stratosphere
- HPCC

Kolon yapısı temel alınarak kayıt tutan veri tabanlarının avantaj ve dezavantajları:

Kayıtlar tek bölmede tutulur, her kayıt kendi içinde kolon-değer çiftlerini oluşturur. Bu kolonlar anahtar değerleriyle ilişkilidir. Minumum ölçektede olsa ilişki kurulmasına olanak tanır. İşlemler bölme bazlı yapıldığından veri bütünlüğü ve tutarlılık problemi yoktur. Verilerin kopyalanması ve çoğaltılmasına uygundur.

2.1.2.3 Grafik yapısı temel alınarak kayıt tutan veri tabanları

Kayıtlar birbirleriyle düğümler kurarak bağlantılı olarak kayıt altına alınırlar. Arada kurulan bağlantıların isimlendirilmesi, yön bilgisi ve özellikleri olmaktadır. Aşağıda örnek grafik yapısı baz alınarak kayıt tutan veri depolama sistemlerinden bazıları listelenmiştir.

- Neo4j
- SAP HANA
- OrientDB
- JanusGraph
- AnzoGraph
- AllegroGraph

Grafik yapısı temel alınarak kayıt tutan veri tabanlarının avantaj ve dezavantajları:

Kayıtların eklenmesi ve değiştirilmesinde diğer veri depolama sistemlerine göre hız konusunda geride kalmaktadır. Veriler arası ilişkiler üzerinde tasarlanmış olmasının doğal sonucu olarak ekleme düzenleme işlemleri daha yavaş olmaktadır. Sistem yapısı değiştirilebilir olması, geliştiriciler için esneklik sağlamaktadır. Yapılam ekleme ve düzenlemeler ilişkili kaydedildiğinden sorgulama tarafında hızlıdırlar.

2.2 Apache Cassandra

Apache Cassandra [28] kolon yapısı temel alınarak tasarlanmış ölçeklenebilir, açık kaynak kodu, dağıtık sistemler üzerine uygulanabilir, NoSQL veritabanı yönetim sistemidir. Apache Cassandra, Apple (75.000 düğümlü 10PB lık veri depolaması), Netflix (2.500 düğüm, 420TB'lık veri ve günlük 1 trilyon üzeri istek), eBay (100 düğüm üzeri ve 100TB veri) gibi bazı büyük firmalar kullanmaktadır.

Apache Cassandra çoklu düğümler üzerinden büyük veriyi işlemek üzerine tasarlanmıştır. Verilerin kümeler arasında kurulmuş çoklu düğüm bağlantıları sayesinde hataya karşı dayanıklıdır. Her düğüm eşler arasında Gossip prokolünü kullanarak haberleşir ve sürekli durumu hakkında bilgileri güncellenir.

Apache Cassandra yetkilendirilmiş bir kullanıcının herhangi bir veri merkezindeki düğüme erişimine, sorgulamasına ve veri yazmasına CQL (Cassandra Query Language) diliyle olanak tanır.

2.2.1 Apache Cassandra Anahtar Yapıları

Node (Düğüm): Verilerin nerede saklandığı bilgisini tutan Cassandra'nın temel yapı taşıdır.

Datacenter (veri merkezi): İlişkili düğümlerin oluşturduğu gruptur. Veri merkezleri sanal veya gerçek olabilir. Farklı iş yükleri ayrı veri merkezlerini kullanmalıdır bu durum hem fiziksel hemde sanal veri merkezleri içinde geçerlidir.

Cluster (küme): Bir veya birden fazla veri merkezini kapsar.

Commit log (teslim logu): Tüm veriler ilk olarak dayanıklık için commit log içerisine yazılır. Sonrasında SSTables'a aktrıldıktan sonra kaydedilebilir, silinebilir veya yeniden kullanım için bekletilebilir.

CQL Table (CQL Tablo): Kolonlara birincil anahtara sahip olarak girilmiş satırların oluşturduğu tablolardır.

2.2.2 Apache Cassandra'yı yapılandırmak için gerekli bileşenler

Gossip : Düğümlerin durumları ve konumları hakkında bilgilerinin birbirleri arasında periyodik olarak keşif amacıyla paylaşıldığı peer-to-peer prokoldür.

Partitioner (bölümleyici): Düğümlerin veri bölümlerinin hangi kümelerde ve düğümlerde kopyalarının tutulacağını belirler. Bir bölümleyici ilgili satırla birincil anahtarını hash işlemine sokan fonksiyondur.

Replication factor (çoğaltma faktörü): Kümelerin toplam çoğaltma sayısıdır. Bir çoğaltma faktörü düğümdeki her satırın sadece birkez çoğlatılma halini kasteder. Bu değer kadar çoğaltma miktarı artar. Tüm kopyalar eşit değerde önemlidir, birincil veya yedek kopyalar yoktur. Genelde kopya sayısı 1'den büyük küme sayısından küçük olur.

Replica placement strategy (çoğalma yeri stratejisi): Kayıt kopyaları birden fazla düğümlerde tutulur. Çoğalma yeri stratejisi kopyaların tutulacağı düğümlerin yerini belirler.

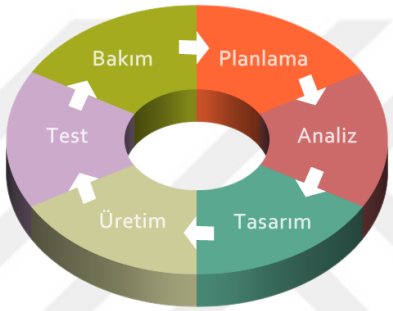
Snitch: Veri merkezindeki makine gruplarını ve çoğalma stratejisinin kullandığı topolojileri tanımlar.

cassandra.yaml konfigürasyon dosyası: Ana konfigürasyon dosyasıdır. Kümeler için özellik tanımlamaları, tabloları için önbellek parametreleri, performans için kaynak kullanım parametreleri, zaman aşımı, istemci bağlantı sınırlamaları, yedekleme ve güvenlik gibi birçok ayarlamaların yapıldığı dosyadır.

Sistem çalışma tabloları: CQL dili kullanarak depolama özelliklerinin değiştirilebilmesi için ana tablolara üzerinde değişiklik yapılmasına olanak sağlar.

BÖLÜM 3. SİSTEM TASARIMI VE MODÜLLERİ

Yapı tasarlanırken asıl hedeflenen sistem bütün olarak ele alındığında benzer sistemlere uyarlanabilen esnek ve genel bir yapı oluşturabilmektir. Yazılım yaşam döngüsünde maliyetsiz, her modül kendi içinde bir yapı olduğu için sürdürülebilirliğe uygun tasarlanmıştır. En düşük ölçekteki firmalardan en büyük ölçekteki firmalara kadar esnek ve ölçeklenebilir bir yapı kurgulanması hedeflenmiştir. Şekil 3.1'de bulunan yazılım yaşam döngüsüne göre de belirtilen esnek bir yapının kurulması amaçlardan birtanesidir.

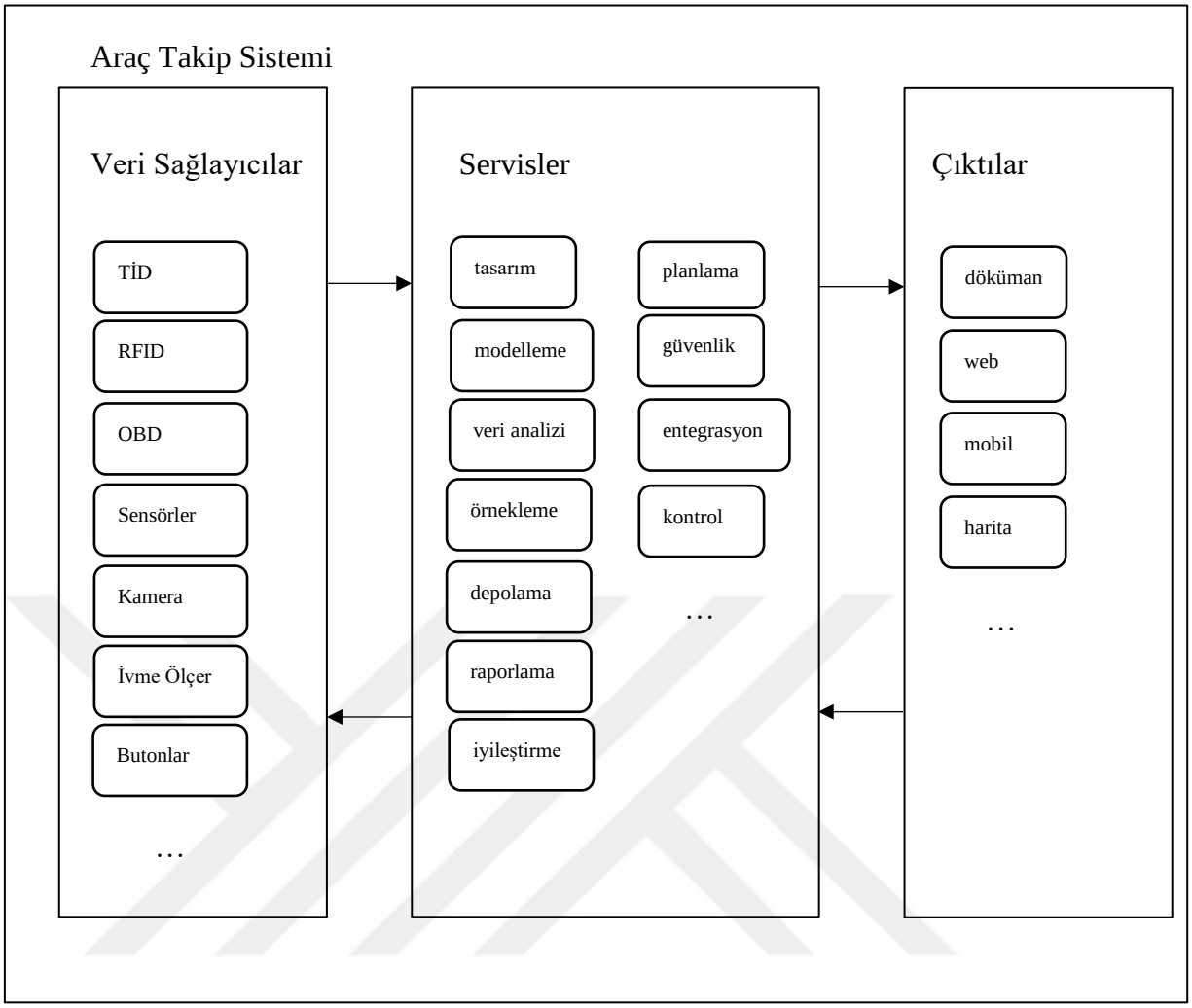


Şekil 3.1 Yazılım yaşam döngüsü

Yapılan tasarımda amaçlanan nitelikleri özetlesek;

- istemciler ile cihazlar arası etkileşim süresini en aza indirmek,
- sistem kaynaklarını doğru kullanarak daha fazla görevi daha az yük ile yapmak,
- veri güvenilirliğini arttırmak,
- sistem kaynaklarını doğru kullanarak daha fazla görevi daha az iş ile yapmak,
- yatay ölçeklenebilir bir sistem oluşturarak veri büyümesine paralel sistem büyümesini kolaylaştırmak

Araç takip sistemi değerlendirmeye alındığında genel olarak Şekil 3.2'deki tasarım yapısı ortaya çıkmaktadır.



Şekil 3.2. Araç takip sistemi mimari yapı tasarımı

Belirtilen yapının oluşturulabilmesi ancak doğru alt yapının sağlanması ve doğru teknojik seçimlerin yapılmasıyla mümkün olur. Araç takip sistemleri büyük veri yığınları üzerinde çift yönlü ve gerçek zamanlı işlem yapılmasını gerektirir. Bundan dolayı gerçek zamanlı verileri işlemek için Apache Spark tercih edilmiştir. GSM üzerinden TCP ile verilerini gönderen taşıt izleme donanımlarının verileri paralel programlamanın yetkinlikleri kullanılarak portlar üzerinden çoklu soketlerle alınmıştır. Alınan veriler sunucular üzerinden dağıtılabilmesi için açık kaynak kodlu, güçlü, ücretsiz ve hataya dayanıklı olmasından dolayı Apache ActiveMQ mesajlaşma hizmeti tercih edilmiştir.

ActiveMQ ile yakalanan ham verilerin analizi için hem gerçek zamanlı ayrıştırılmaya en uygun yapı olması, hem de verileri depolamada kullanacağımız NoSQL veritabanı olan Cassandra ile kolayca entegre edilebilir olmasından dolayı en uygun

teknoloji olan Apache Spark tercih edilmiştir. Cassandra'nın tercih edilmesinin sebebi ise ekleme, silme ve raporlama işlemlerinde hızlı olması ve kolonlar üzerinde işlemlerin yapılmasına olanak sağlamasıdır.

Veri sağlayıcıları ile servisler arasında çift yönlü veri akışı olmalıdır. Taşıt izleme donanımları (TİD), sensörler üzerinden topladığı verileri direkt sunucuya göndermektedir. Ayrıca sunuculardan TİD'lere veri gönderimi gerçek zamanlı olmalıdır. Doğru veriyi en hızlı şekilde kullanıcıya ulaştırabilmek için, veri iletimi kullanıcı tarafına soketler üzerinden sağlanmıştır. Bunun için NodeJS ve WebSocket teknolojileri kullanılmıştır.

3.1. Yapının Tasarımı

İşleyişin en başından sonuna kadar bütün olarak düşünülen sistem olabildiğince küçük modüllere ayrılarak tasarlanmalıdır. Her parça kendi içinde sorumlulukları üstlenip aradaki bağlantılar için esneklik sağlanmıştır.

Yapılan çalışma sistem kaynaklarıyla yazılım gereksinimlerine paralel olarak düşük maliyet ile artabilen, sürdürülebilir, hataya karşı dayanıklı, dağıtık bir sistem kurgusu hedeflenmiştir. Tamamlanan yapının kademeleri aşağıdaki gibidir.

Kademe 1: ActiveMQ ile birden çok veri sağlayıcısından alınan ham girdiler alıcısından bağımsız olarak kuyruk (queue) ile özdeşleştirilir.

Kademe 2: Kuyruk (queue) üzerinden okunan veriler, Apache Spark ile birleştirilerek Map/Reduce işlemine sokulur.

Kademe 3: Spark ile analiz edilip anlamlandırılan veriler alıcısı varsa bir konu ile özdeşleştirilerek yayına sunulur ve tüm veriler Cassandra veri tabanına kaydedilir.

Kademe 4: Ayrıştırılmış veriler NodeJS ile consumer olarak okunur ve ilgili kullanıcısına Socket.IO kütüphanesiyle WebSocket üzerinden gönderimi yapılır.

NodeJS, JavaScript programlama dili ile sunucu tarafında uygulama geliştirmeye olanak sağlar ve asenkron görevler için uygundur. Güçlü yapısıyla anlık işlemleri asenkron olarak düşük kaynak tüketimi ve yüksek performans ile işler.

Kademe 5: Kullanıcı arayüzü olan Web sayfası içerisinde Openlayers kullanılarak çoklu harita entegrasyonu yapılır. Openlayers açık kaynak kodlu, ücretsiz ve birden fazla harita ile entegrasyonu olan Javascript ile geliştirilmiş harita katmanlama kütüphanesidir. NodeJS ile yayın yapılan veriler istemci (client) tarafında Javascript ile yakalanır, Openlayers'ın gerekli API'leri kullanılarak haritalar üzerine anlık olarak yansıtılır.

Yukardaki kademelerde verinin veri sağlayıcılar tarafından analiz edilip kullanıcıya iletimi açıklanmıştır. Aşağıdaki kademelerde ise kullanıcı tarafından veri sağlayıcılar tarafına veri iletiminin kademeleri açıklanmıştır.

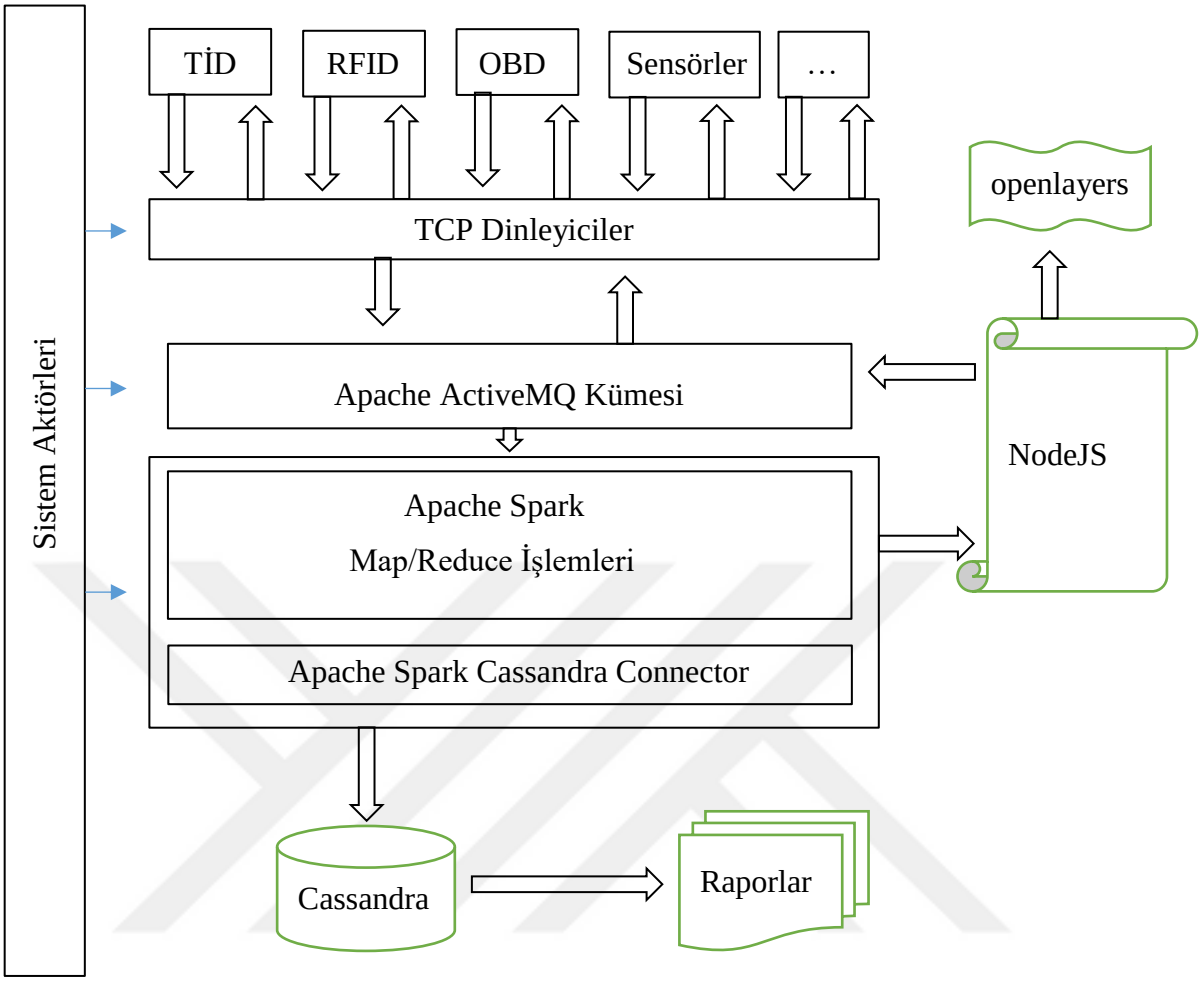
Kademe 1: Kullanıcı takip cihazını programlamak için belirlenen parametreler Javascript ve WebSoket üzerinden gönderilen komut parametreleri NodeJS ile sunucu tarafında yakalanır.

Kademe 2: NodeJS ile yakalanan veriler güvenlik kontrollerinden geçirilir. Cihaz yapısına uygun formata dönüştürülür.

Kademe 3: NodeJS verileri bir kuyruk (queue) ile özdeşleştirip ActiveMQ üzerinden üretici (producer) olarak yayın yapar.

Kademe 4: İstemcilerin (client) bağlı olduğu dinleyiciler kuyrukta (queue) bulunan veriler için tüketici (consumer) konumundadırlar. Uygun komutu alarak uygun istemciye iletme görevini de üstlenirler.

Tüm sistemin iş akış diyagramı Şekil 3.3 üzerinde görünmektedir.



Şekil 3.3. Sistem iş akış mimarisi

3.2. Verinin Anlık Olarak İletimi

Anlık işlem yapılması gereken verilerde, veri sağlayıcıların veri gönderim sıklıkları değişebileceği için, yoğunluğun ne zaman artacağının kestirilmesi zordur. Eğer sistem doğru tasarlanmaz ise gecikmelere sebep olabilir, gönderici ile alıcı arasında yığılma veya darboğaz oluşabilir. Bu durumun oluşması Araç Takip Sistemleri'nde istenmez. Bunun önlenmesi için alınan her mesaj dinleyici tarafında asgari düzeyde işleme alınarak kuyruk yapısına aktarılır. Port üzerindeki her bağlantı bağımsız iş parçacığına (thread) atanır, paralel programlamanın (parallel programming) yetkinlikleri kullanılarak üretici (producer) veri akışını başlatır.

Veri sağlayıcılarından veriler, TCP ile ServerSocket ve Socket sınıfları kullanılarak alınmıştır. Bağlantı kuran her cihaz kendi Socket sınıfından türetilmiş nesneyle bağımsız bir iş parçacığı olarak işleme devam etmektedir. Şekil 3.4. de sunucu tarafında dinleyici için kod örneği paylaşılmıştır.

```
try {
    server = new ServerSocket(Globals.PORT);
    while (true) {
        socket = server.accept();
        Device device = new Device(socket);
        DeviceListAdd(device);
        device.start();
    }
} catch (Exception e) {
    Globals.alert("MultiDeviceAccept.run()"+e.getMessage());
}
```

Şekil 3.4. TCP istemci/sunucu kod örneği

Port üzerinden okunan veriler, oluşturulan iş parçacıkları ile Şekil 3.5 de yer alan kod örneğindeki gibi üreticiler ile ActiveMQ kuyruğuna aktarılır.

```
ConnectionFactory factory = new ActiveMQConnectionFactory(Globals.brokerUrl);
Connection connection = factory.createConnection(Globals.username, Globals.password);
connection.start();
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

Destination destination = session.createQueue(Globals.DESTINATION_PRODUCER);
MessageProducer producer = session.createProducer(destination);
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

// Create a messages
TextMessage message = session.createTextMessage(Msg);

producer.send(message);

// Clean up
session.close();
connection.close();
```

Şekil 3.5. Mesajın ActiveMQ kuyruğa eklenmesi kod örneği

ActiveMQ ile veri transferi işlemi sırasında yük devretme (failover) yapılarak, sunuculardan birine anlık erişilememesi durumunda veya zaman aşımında diğer sunuculara veri iletimi sağlanır. Yük devretme ile gönderim metodu olarak kullanılan bağlantı parametrelerine eklenen "randomize" parametresi "true" yapılarak sunuculara rastgele erişim sağlananır. Erişim herhangi bir nedenden başarısız olursa kalan sunucular arasında rastgele seçim yapılır. Bu işlem başarılı olana kadar süreç devam eder. Şekil 3.6 da yük devretme örneği paylaşılmıştır.

```
failover:(tcp://localhost:61616,tcp://remotehost:61616)?initialReconnectDelay=100
```

Şekil 3.6 ActiveMQ failover çoklu sunucu örneği

3.3. İşleme Alma, Anlamlandırma, Temizleme

ATS'lerde verilerin anlık olarak okunup işleme alınması herzaman ihtiyaçların tamamını karşılayamaz. Okunan verilerde anlam çıkarımı, gruplama, özelleştirme, müdahale etme gibi çeşitli özelliklerin kazandırılması gerekir. Bu özelliklerin kazanımı için Apache Spark kullanılarak büyük ölçekli verilerde dağıtık işlem yapılması amaçlanmıştır. Apache Spark anlık veri işleme yeteğine sahiptir. Kendisine gelen istekleri aksiyon fonksiyonu (action function) çağrılana kadar bekletir (lazy evaluation), elde ettiği bulguları geçici bellekte tutar ve işlemlerde bekleme olmadan çalışmaya devam eder.

Apache Spark Streaming, Apache Spark'ın veri akış kütüphanesidir. Yerleşik olarak Kafka, Flume, Kinesis, Sockets gibi birçok mesajlaşma sistemini desteklediği gibi harici kaynaklardan da veri okunmasını desteklemektedir. Apache ActiveMQ ile Apache Spark entegrasyonu Şekil 3.7'de görüldüğü üzere akan veriler için Spark Stream tarafından DStream (Discretized Stream) oluşturulur. DStream RDD (Resilient Distributed Dataset)'lerin sürekli olarak oluşturulduğu ve aynı tipte olarak üretilen dizilerdir.

```
JavaDStream<String> queueReceiverStream = ssc.receiverStream(new ActiveMQReceiver());
JavaDStream<String> parsed = queueReceiverStream.flatMap(s -> ...);
```

Şekil 3.7. Activemq-Spark entegrasyonu

ActiveMQ üzerindeki kuyruk ilişkili yığından okunan veriler, gerekli işlemlerden geçirildikten sonra kuyruktan silinir. Böylece sürekli olarak kuyrukta temizleme işlemi yapılmaktadır.

3.4. Veri depolama ve yayınlama

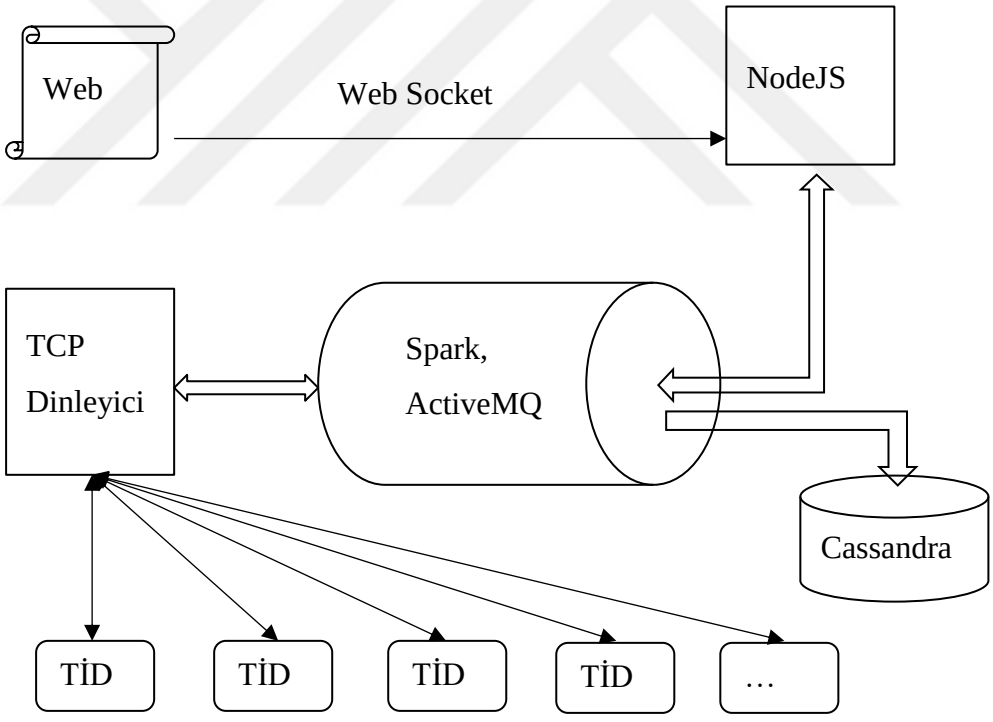
Apache Spark ile Cassandra entegrasyonu Spark Cassandra Connector kütüphanesi ile sağlanmıştır. Bu kütüphane Spark ile entegre çalıştığı için Map/Reduce ile kayıt işlemlerini tamamlar.

Akan verilerin son kullanıcılara gerçek zamanlı gösterimi için sistem veritabanına kaydetmeden önce konu üzerinden okunur. Spark ile anlamlandıran veriler, ActiveMQ üzerinden bir konu ile JSON formatında yayınlanır. Konu ile yayın yapıldığında sadece o yayının abonelerine gönderim sağlanır. NodeJS yayın yapılan verileri yakalamak için ActiveMQ istemcisi olarak davranır. Her verinin yayınlanarak sistem kaynaklarının boş yere tüketilmesinin önüne geçmek için, sistem üzerinden aktif olan kullanıcı bilgisi NodeJS ile ActiveMQ üzerinden yayın yapan yazılıma bildirilir.

NodeJS veriyi ilgili kullanıcı ile eşleştirir. Son kullanıcı ile Web sunucusu arasındaki iletişimin sağlanması için Socket.IO kullanılmıştır. Socket.IO ile son kullanıcı

tarafında istek olmadan sunucu tarafına veri gelmesi durumunda cevap dönülebilir. Socket.IO ikiyönlü (full duplex) ve eş zamanlı (simultane) iletişime izin verir. İstemci tarafında yakalanan konum verileri Openlayers kullanılarak haritalar üzerinde konumlandırılır.

Web istemcisi tarafında TİD'ler için hız limit ayarlaması, bölge tanımlama ve TİD'lere bağlı sensörlere veri gönderimi gibi isteklere gereksinim duyarlar. Sistemin aslında Web istemcisi tarafından sunucuya istek göndermesi gerekir. Bu noktada TİD'den veri alma işleminin tam tersi yolun izlenmesi gerekmektedir. Şekil 3.8'de görüldüğü gibi istemci WebSocket ile NodeJS'e mesajını gönderir. NodeJS ActiveMQ üzerinden bir kuyruk ile ilişkilendirip yayın yapar. Spark üzerinden ileti okunup RDD'ler ile Cassandra'ya kayıt yapılır. Aynı anda TCP dinleyici üzerinden ilgili cihazlara port üzerinden gönderim sağlanır.



Şekil 3.8. İstemciden TİD'e mesaj gönderim şeması

3.5. Yapılandırma

Sistem üzerinde kullanılan araçlar ve bunların versiyonları Çizelge 3.1'de belirtilmiştir.

Çizelge 3.1. Yapılandırma ayarları

Bileşen	Versiyon
Java	1.8.0_111
NodeJS	6.9.5
Apache Spark	1.6.3
Spark Cassandra Connector	1.1.0
Apache Cassandra	2.2.12
ActiveMQ	5.13.12
Openlayers	3.20.1

Veri kaynaklarından okuma ve anlamlandırma işlemi yapıldıktan sonra veriler JSON formatında iletilimi sağlanır. TİD' lerden ve diğer sensörlerden alınan veriler, TİD donanımının IMEI (Uluslararası Mobil Cihaz Kodu) numarası ve olayın gerçekleşme zamanı birincil anahtar alınarak Şekil 3.9 görülen konumlar tablosuna kaydedilir.

```
CREATE TABLE locates (
    imei bigint,
    eventdatetime timestamp,
    ......
    PRIMARY KEY (imei,eventdatetime)
)
```

Şekil 3.9. Konumlar tablosu

NodeJS tarafında Web istemcilerine Socket.IO ile gönderilen veriler Openlayers'ın Javascript API'leri kullanılarak haritalara yansıtılır. Şekil 3.10'de görülen harita üzerinde araçların konum, hız, durum, yön bilgileri ikonlar ile gösterilir.

Şekil 3.10. Openlayers harita ekran görüntüsü

BÖLÜM 4. BULGULAR VE YORUMLAR

Openlayers açık kaynak kodlu ve güçlü bir haritalama yapısına sahiptir. Direkt olarak Google Map, BingMap, OpenMap gibi haritalara entegre olduğu gibi dışardan başka bir harita servisine de entegre edilebilir. Katmanlar üzerine tasarlanan bir yapıya sahiptir.

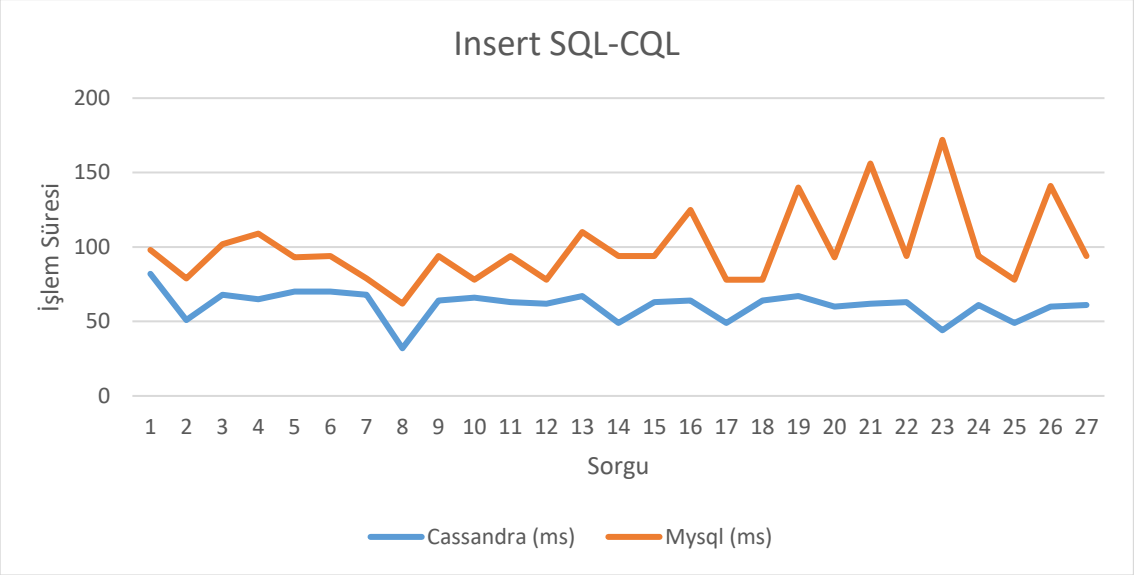
Apache Spark güçlü ve hızlı bir yapıya sahiptir. ATS gibi sistemlerde yani veri sağlayıcılardan elde edilen veri miktarının tahmin edilmesi zor olan sistemler için uygun ve dinamik bir yapısı vardır. Bu tür sistemlerde darboğazın önüne geçilmeli ve ölçeklenebilir bir yapının kurulması gerekmektedir. Büyük veri işleme yöntemleri klasik yöntemlere göre daha yeni oldukları için sürekli geliştirilmektedir. Dolayısıyla sistem kurgulanırken kullanılan araçların versiyonlarına dikkat edilmelidir. Birbirleriyle uyumlu versiyona sahip araçlar seçilmelidir.

Literatürde yapılan araştırmalar incelendiğinde ATS'lerin çoğunlukla klasik yöntemlerle tasarlandığına ve büyük veri işleme metotları için yapılan önermelere karşılaşılmaktadır. Ancak veri sağlayıcılarından son kullanıcıya kadar olan aşamaların tamamı için büyük veri çözümü bulunmamaktadır. Bu çalışmanın en büyük farkı verinin ilk aşamasından son aşamasına kadar iletişimin her aşamasına çözüm sunabiliyor olmasıdır.

4.1. İlişkisel Veri Tabanları ile NoSQL Veri tabanları kıyaslaması

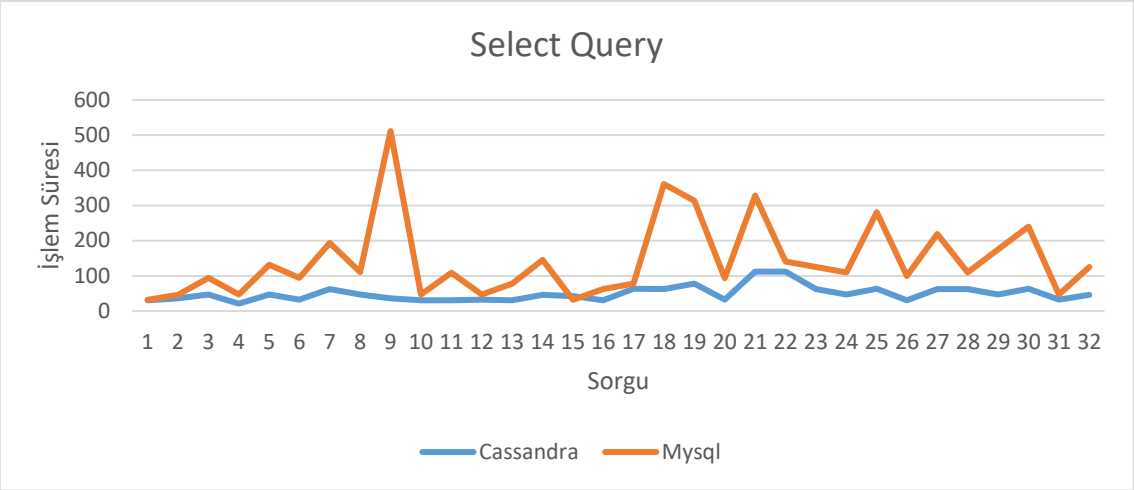
Yapılan araştırmada verilerin kayıt altına alınması için NoSQL çözümü olan kolon tabanlı Apache Cassandra veri tabanı seçilmiştir. Apache Cassandra'nın hataya dayanıklı dağıtık ve ölçeklendirilebilir olması, birçok yazılımı ile uyumlu olması tercih sebebidir. Bu çalışmada Apache Cassandra ile ilişkisel veri tabanı olan MySQL veri tabanı karşılaştırılması yapılmıştır. İki veri tabanı çözümü üzerinde aynı tablolar oluşturulmuştur. MySQL ve Cassandra aynı özelliklerde iki farklı sanal makineye kurulmuştur. Sanal makineler üzerinde Ubuntu 14.04 LTS işletim sistemi çalışmaktadır.

Veri tabanları üzerinde çalıştırılan INSERT SQL-CQL sorgularının sonuçları Şekil 4.1'de görülmektedir. ATS gibi IOT yapıları için INSERT işlem yoğunluğu önceden kestirilmesi imkansızdır. Sistem kaynaklarının doğru kullanımı ve veri kaybının önüne geçilmesi için en düşük sürede işin gerçekleşmesi kritiktir. Yapılan INSERT işlemlerinde aynı sistemler üzerine kurulu iki veri tabanı sistemi içinde her sorguda Cassandra'nın hız konusunda daha iyi sonuçlar elde edildiği görünmüştür.



Şekil 4.1. MYSQL-Cassandra kayıt karşılaştırması

Her iki sistem içinde 6 ay süre boyunca 16 farklı cihazın veri kayıtları tutulmuştur. Oluşturulan birebir aynı veri kayıtlarının üzerinde iki sistemde de sorgular çalıştırılarak süreleri ölçülmüştür. Elde edilen sonuçlar Şekil 4.2'de gösterilmektedir.



Şekil 4.2. MYSQL-Cassandra kayıt getirme süresi karşılaştırma

Sorgu işlemleri birincil anahtarlar üzerinden yapılmıştır. TİD konum kayıtları locates tablosu üzerinde tutulmuştur, altı aylık süre içerisinde 28,055,288 adet kayıt oluşmuştur.

Konumlar tablosunda tutulan sütun bilgileri:

- Cihazın IMEI bilgisi,
- Cihazın uydudan aldığı tarih ve saat,
- Paket geldiğindeki sunucu tarih ve saati,
- Alarm listesi,
- Can-Bus üzerinden alınan yakıt, devir, mesafe, hız, servis vs. bilgiler,
- DOP (Dilution of precision) bilgisi,
- Yön bilgisi,
- Enlem ve Boylam bilgisi,
- Kontak bilgisi,
- Uydular üzerinden alınan hız bilgisi,
- Mesafe bilgisi,
- İvme ölçer ile alınan veriler,
- Sürücü tanıma birimleri,
- Analog ve dijital sensörler,
- Harici sensörlerden okunan veriler (Navigasyon, Ağırlık sensörü vs.).

Çizelge 4.1. Kıyaslama için kullanılan araçlar

Tip	Araç	Versiyon
İşletim Sistemi	Ubuntu	14.04 LTS
Sanallaştırma Platformu	VMware	14
Veritabanı Arayüzü	DBeaver	Enterprise 5.2.2
Veritabanı	MySQL	5.7.24
MySQL Driver	com.mysql.jdbc.Driver	4.x-5.7
Veritabanı	Cassandra	2.2.12
Cassandra Driver	org.jkiss.jdbc.cassandra.CassandraDriver	2.x-3.x

BÖLÜM 5. SONUÇ

Araç takip sistemleri nesnelerin interneti (IOT) genellemesinin bir özel alanı olarak düşünülebilir. İnsanların hayatlarını kolaylaştırma, kontrolü ve tasarruf sağlama, iş gücünü azaltmak için her geçen gün nesnelerin internetine taleplerin artması kaçınılmaz bir gerçektir. Bu bağlamda bu ihtiyacın karşılanması için yenilikçi çözümlerin sunulmasını zorunlu kılmıştır.

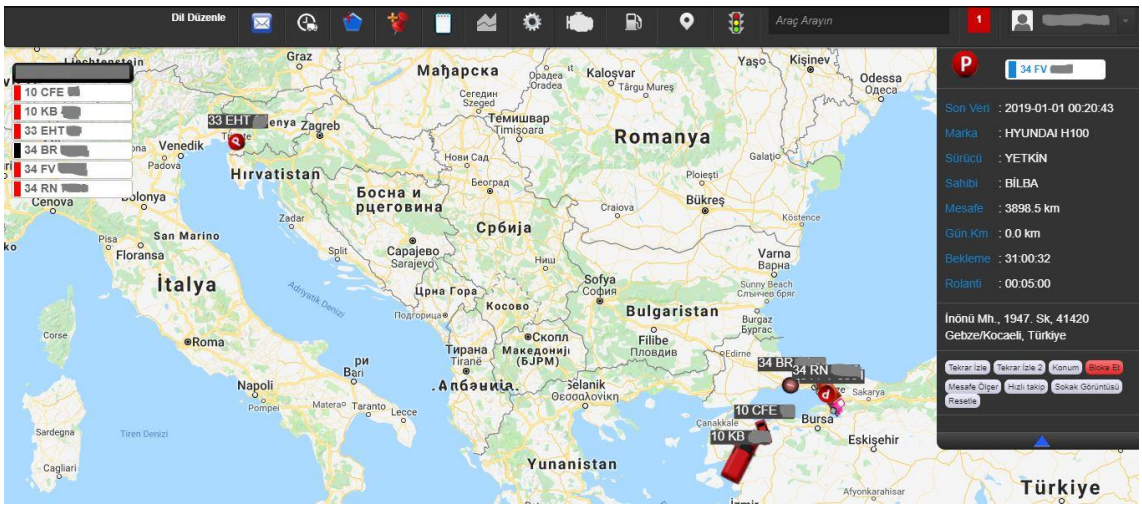
Kendi içerisinde bir ekosistem sunan klasik sistemlerin internetin yaygınlaşıp teknolojinin gelişmesiyle IOT sayesinde bir merkeze bağlanarak, akıllı sistemlere dönüştürülmesine olanak sağlar. IOT gibi büyük veri trafiği problemi olan sistemler için klasik çözümler yerine büyük veri çözümlerinin uygun olduğu önerilir. Bu bağlamda Klasik yöntemlerle geliştirilen Araç Takip Sistemleri'nin güncel metot ve yöntemlere uyarlanarak büyük veri çözümleriyle daha esnek, kabiliyetli bir sistem kurulması sağlanamıştır. Tasarlanan yapı akademik, endüstriyel, askeri alanda veri işleme saklama ve sunma yeteneklerinin geliştilmesini sağlamak için yeni çözümler sağlayabilir.

Bu tezde, araç takip sistemlerinin gelişmesine yardımcı olmak amacıyla dayanıklı, ölçeklenebilir, tutarlı ve dağıtık bir yapı oluşturulup uygulama gerçeklenmiştir. Tasarlanan sistemin son kullanıcıya sunma biçimleri gibi birkaç küçük değişiklik yapılarak istenirse farklı internet nesneleri ile uyumlu yapının korunmasına dikkat edilmiştir. Uygulamanın analiz aşamasında, sistemin tüm işlevlerinin detayları belirlendi. Bu belirtimlere bağlı sistem gereksinimleri genel yapıyla ters düşmeyecek unsurlar ile netleştirildi ve çözümler belirlendi. Analiz, araştırma, geliştirme ve sürekliliği sağlama üzerine kurulan yapı, birbirleriyle iletişim halinde olan ancak bağımsız çalışan birden çok teknolojik bileşenler ile tasarlanmıştır. Ham verinin (raw data) TİD'ler üzerinden alınmasından itibaren son kullanıcıya sunulması arasındaki tüm detaylar kendi içerisinde çözümlenmiş yani büyük problem küçük parçalara bölünerek çözümlenmiştir. Uygun yollarla bu çözümler birleştirilerek nihai genel çözümün elde edilmesi sağlanmıştır.

Tez kapsamında gerçek zamanlı akış işleme için akan veriler paralel olarak Map/Reduce yapısında dağıtık olarak işlenmiş, son kullanıcıya asenkron olarak sunulmuştur. Geçmişe dönük verilerin raporlanabilmesi analiz edilerek işlenebilmesine olanak sağlayabilen bir sistem elde edilmiştir. Geliştirilen sistemin ihtiyaç olan her modülü arasında tam çift yönlü istek ve cevap gönderebilir yapıda olması ile zaman, sistem kaynakları ve iletişim optimize edilmiştir.

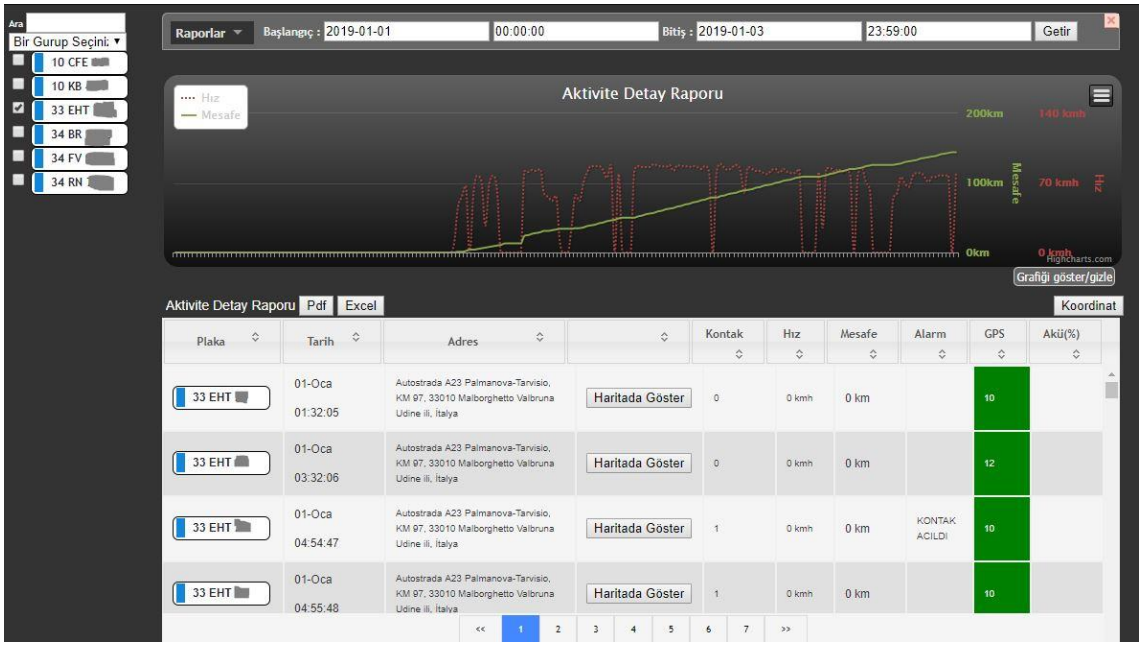
Geliştirilen yapının klasik veri tabanları ile kıyaslanabilmesi için örnek seneryolar geliştirilip sonuçlar elde edilmiş ve raporda sunulmuştur. TİD'larına bağlı sensörlerden alınan veriler sistem üzerinden gerçek zamanlı ve dağıtık işlenerek son kullanıcıya Web uygulaması üzerinden sunulmuştur. Geliştirilen her modül düşünüldüğünde kendi başına bir genel çözümü ürettiğini ve bu çözümü ATS üzerine uyguladığını farketmek mümkündür.

Geliştirilen ATS uygulamasında Şekil 5.1'de görüldüğü üzere harita üzerinde gerçek zamanlı olarak araçların gösterimini amaçlamıştır.



Şekil 5.1. Araçların harita üzerinde gerçek zamanlı uygulama ekran görüntüsü

Ayrıca Şekil 5.2'de görüldüğü gibi sistem geçmişe dönük verilere ulaşılması için araç verilerini veri tabanında kayıt altına almaktadır.



Şekil 5.2. Araçların geçmişe dönük detay verileri

KAYNAKÇA

[1] CENSUR, Çağrı. Akilli Kent İçin IOT Tabanlı Akıllı Sokak Aydınlatma Sisteminin Tasarlanması. Mersin: Mersin Üniversitesi, 2018.

[2] EĞER, Önder. http://www.endustri40.com/big-datanin-buyuk-veri-endustriyel-kullanimi/. [t.y.]. (9 Ocak 2019).

[3] KOCA, Başar & Ayhan CEYLAN. "Uydu Konum Belirleme Sistemlerindeki (GNSS) Güncel Durum," **Geomatik Dergisi**. 3, 1: 63-73, 2018.

[4] Raneland, Daniel. Performance- and Cost-efficient Cloud Architectures. 2014.

[5] ÖZTÜRK, Serdar & Hatice Ediz ATMACA. "İlişkisel ve İlişkisel Olmayan (NoSQL) Veri Tabanı Sistemleri Mimari Performansının Yönetim Bilişim Sistemleri Kapsamında İncelenmesi," **BİLİŞİM TEKNOLOJİLERİ DERGİSİ**. 10, 2: 199-209, 2017.

[6] https://www.geeksforgeeks.org/acid-properties-in-dbms/. [t.y.]. (9 Ocak 2019).

[7] "Introducing JSON," **JSON**. www.json.org. [t.y.]. (28 Haziran 2018).

[8] https://www.computerworld.com/article/2493701/data-center/by-2020--there-will-be-5-200-gb-of-data-for-every-person-on-earth.html. [t.y.]. (9 Ocak 2019).

[9] YAVUZ, Gürcan, Sevcan AYTEKİN & Muammer AKÇAY. "APACHE HADOOP VE DAĞITIK SİSTEMLER ÜZERİNDEKİ ROLÜ," **Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Dergisi**. 27: 43-54, 2012.

[10] Shvachko, Konstantin, Hairong Kuang, Sanjay Radia & Robert Chansler. "The Hadoop Distributed File System". California: Yahoo!, 2010.

[11] http://www.ibm.com/developerworks/aix/library/au-cloud_apache/(2010). [t.y.]. (10 Ocak 2019).

[12] MIAO, Li. Comparative Analysis of Two Clustering Algorithms: K-means and FSDP (Fast Search and Find of Density Peaks). "Master's Theses and Graduate Research," San Jose State University, 2015.

[13] Ranganathan, Sindhuja. Improvements to K-means Clustering. Tampere University of Technology, 2013.

[14] https://www.ibm.com/developerworks/library/j-mahout/index.html. [t.y.]. (11 Ocak 2019).

[15] Ruiz, Roger Carrillo. Secondary indexing for the HBase distributed database. Montréal: McGill University, 2013.

[16] Bhamra, Kavita. A Comparative Analysis of MongoDB and Cassandra. University of Bergen, 2017.

[17] https://spark.apache.org/news/spark-wins-daytona-gray-sort-100tb-benchmark.html. [t.y.]. (11 Ocak 2019).

[18] Tchutchulashvili, Mariami. Micro-activities for Java Programming Learning. 2016.

[19] Kohn, Tobias. Teaching Python Programming to Novices: Addressing Misconceptions and Creating a Development Environment. 2017.

[20] "Microsoft," https://docs.microsoft.com/tr-tr/azure/hdinsight/storm/apache-storm-overview. [t.y.]. (11 Ocak 2019).

[21] http://storm.apache.org/releases/1.0.6/Concepts.html. [t.y.]. (10 Ocak 2019).

[22] https://kafka.apache.org/intro.html. [t.y.]. (10 Ocak 2019).

[23] "Microsoft," https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about. [t.y.]. (10 Ocak 2019).

[24] McAteer, Ian Noel, Muhammad Imran Malik, Zubair Baig & Peter Hannay. "Security vulnerabilities and cyber threat analysis of," Australian Information Security Management: 2017.

[25] Manandhar, Srijan. Mqtt Based Communication In Iot. "Master of Science thesis," 2017.

[26] https://activemq.apache.org. [t.y.]. (10 Ocak 2019).

[27] http://bsonspec.org/. [t.y.]. (10 Ocak 2019).

[28] http://cassandra.apache.org/. [t.y.]. (10 Ocak 2019).