
GAZİ ÜNİVERSİTESİ
Gazi
1926

SINIR GÜVENLİĞİ İÇİN BÜYÜK VERİ TEKNİK VE TEKNOLOJİLERİ İLE BORU HATTI TASARIMI

Fatih AYDEMİR

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2016

Fatih AYDEMİR tarafından hazırlanan "SINIR GÜVENLİĞİ İÇİN BÜYÜK VERİ TEKNİK VE TEKNOLOJİLERİ İLE BORU HATTI TASARIMI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Aydın ÇETİN

Teknoloji Fakültesi Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……

Başkan : Prof. Dr. O. Ayhan ERDEM

Teknoloji Fakültesi Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……..

Üye : Doç. Dr. İlyas ÇANKAYA

Mühendislik ve Doğa Bil. Fakültesi, Yıldırım Beyazıt Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……..

Tez Savunma Tarihi: 03/06/2016

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

…….

Prof. Dr. Metin GÜRÜ

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Fatih AYDEMİR
03/06/2016

SINIR GÜVENLİĞİ İÇİN BÜYÜK VERİ TEKNİK VE TEKNOLOJİLERİ İLE BORU HATTI TASARIMI

(Yüksek Lisans Tezi)

Fatih AYDEMİR

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2016

ÖZET

Gelişen teknoloji, iletişim ve ulaşımdaki kolaylıklar sınırların yavaş yavaş ortadan kalkmasına yol açmıştır. Bu sebeple farklı coğrafyalarda gerçekleşen olaylar, sınırların ötesine geçerek etkisini büyük kitlelere hissettirmektedir. Sınırları denetlemek, korunmak, kaçak geçişlere karşı hızlı ve kapsamlı çözümler bulmak birçok ülkenin ana problemi haline gelmiştir. Sınır güvenliğinin kapsamı düşünüldüğünde farklı zamanlarda, farklı kaynaklardan ve farklı tiplerde gelen verinin, gerçek zamanlı olarak alınması ve çözümlenmesi gerekmektedir. Bu tezde, sınır güvenliği için gerçek zamanlı akan veri ve yığın veri üzerinde analiz yapmayı sağlayan tutarlı, dayanıklı, dağıtık ve ölçeklendirilebilir bir sistem ilk örneği geliştirilmiştir. Sistem akan veri ve yığın veri üzerinde işlem yapmayı sağlayan Lambda Mimarisi temel alınarak hem gerçek zamanlı hem de geçmişe dönük veriler üzerinde analiz yapılmasına olanak sağlayan bir yapıda oluşturulmuştur. Geliştirilen sistemin test edilebilmesi için örnek senaryolar oluşturulmuş elde edilen sonuçlar gözlemlenmiştir. Mevcut sistemlerin sahip oldukları veri işleme yetenekleri ile kıyaslandığında önerilen sistemin yeni yaklaşımlar ve çözümler geliştirilmesine olanak sağlayan erken uyarı sistemi olarak kullanılabilecek bir tasarıma sahip olduğu görülmüştür.

Bilim Kodu	:	92414
Anahtar Kelimeler	:	Sınır Güvenliği, Büyük Veri, Boru Hattı, Lambda Mimarisi, Gerçek Zamanlı Veri İşleme, Yığın İşleme, Apache Kafka, Reaktif Akışlar, Apache Spark, Apache Cassanda
Sayfa Adedi	:	54
Danışman	:	Doç. Dr. Aydın ÇETİN

DESIGNING A PIPELINE WITH BIG DATA TECHNIQUES AND TECHNOLOGIES FOR BORDER SECURITY

(M. Sc. Thesis)

Fatih AYDEMİR

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2016

ABSTRACT

Developing technology, communication and transportation facilities have led the gradual disappearance of borders. The incidents taking place in different countries go beyond the limits and are being felt by large audiences. Countermeasures against rapidly increasing terrorist attacks and internal turmoil have caused more complicated security problems in recent years. Protecting and controlling the borders, finding rapid and comprehensive solutions to illegal border passing problems have become a major problem in many countries. When considering the scope of the border security, it would be required to receive and analyze the data coming from different sources at different times and types in real time. In this thesis, a cost-effective, robust, scalable and flexible system prototype was developed. The developed system is based on Lambda Architecture which provides real time data processing and batch processing capabilities for the border security applications. The behavior of the system was observed with test scenarios. Results show that when compared with the data processing capabilities of existing systems, proposed system has a structure that provides a base for new approaches and solutions that could be used as an early warning system.

Science Code	:	92414
Key Words	:	Border Security, Big Data, Pipeline, Lambda Architecture, Real-time Data Processing, Batch Processing, Apache Kafka, Reactive Streams, Apache Spark, Apache Cassanda
Page Number	:	54
Supervisor	:	Assoc. Prof. Dr. Aydın ÇETİN

TEŞEKKÜR

Tez konusunun tayininden tamamlanmasına kadarki her aşamada yakın ilgi ve desteğini gördüğüm, çalışmalarımın yönlendirilmesi ve sonuçlandırılmasında büyük emeği geçen tez danışmanım Sayın Doç. Dr. Aydın ÇETİN'e, gösterdikleri sabır ve verdikleri her türlü destek için ailem ve arkadaşlarıma teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
fps	Frame per second (Saniyede kare)
sn	Saniye

Kısaltmalar	**Açıklamalar**
Dstream	Direct stream
HDFS	Hadoop distributed file system
IDC	International Data Corporation
JSON	JavaScript object notation
NoSQL	Not only SQL
OTM	Olay tabanlı mimari
PSVM	Parallel support vector machine
PTDS	Persistent Threat Detection System
RAM	Random access memory
RDD	Resilient distributed dataset
SVM	Support vector machine
SQL	Structured query language
TCP	Transmission control protocol

1. GİRİŞ

Teknolojideki hızlı gelişimin sonucu olarak iletişim, ulaşım ve silahlanma imkânları kolaylaşmış; böylece yeni tehditlerin ortaya çıkmasına ve var olan tehditlerin yöntem değiştirmesine yol açmıştır. Askeri alandaki teknik ve teknolojik değişim, yönetim katmanındaki fikir ayrılıkları, ekonomik etkenlerin birey, toplum ve devletler üzerindeki olumsuz etkisi gibi problemler tehditlerin suç işleme girişimlerini artırmış; bireylerin can ve mal güvenliğinin sağlanması zorlaşmıştır. Bu sebeple, insan güvenliğini sağlamakla sorumlu olan birimler yeni yaklaşımlara ve tedbirlere ihtiyaç duymaktadırlar [1]. Güvenlik çok katmanlı bir mimari gibi düşünüldüğünde, en dıştaki katman yaşanılan bölgenin dış etkenlere karşı korunması olacaktır. Dolayısı ile güvenlik, sınırlardan başlar denilebilir.

Son yıllarda yaşanan terör eylemleri sonucunda sınır güvenliği, tekrar birçok ülkenin ana gündem maddesi haline gelmiştir. Sosyal medya, gazete, haber programları ve meclis gündemlerinde ilgili birimlerin yetersizliklerinden bahsedilmiştir. Tüm bunların sonucu olarak, yeni güvenlik teşkilatları kurulmakta ve çözüme yönelik yeni tedbirler alınmaya başlanmıştır.

Teknoloji şirketleri tarafından geliştirilen, sınır güvenliği alanında, birçok sistem mevcuttur. Bunlardan en büyüğü Lockheed Martin [2] şirketi tarafından geliştirilerek Amerika-Meksika sınırına yerleştirilen, Afganistan ve Irak'ta aktif olarak kullanılan Persistent Threat Detection System (PTDS) [3] adlı sistemdir. Balon tabanlı bu sistem, sahip olduğu çoklu görev sensörleri, uzak noktalar arası iletişim sağlayabilme, istihbarat ve gözetleme kabileyetleri sayesinde kritik roller üstlenebilir. ASELSAN [4], HAVELSAN [5], Poseidon [6] gibi şirketler de benzer yetenekleri ülkemize kazandırabilmek adına çalışmalar yapmaktadır. Geliştirilmeye çalışılan bu sistemlerin altyapı ve yaklaşımları farklılıklar göstermesine rağmen hepsinde ortak bir amaç vardır. Bu amaç farklı çalışma prensiplerine sahip programların birbirleriyle uyumlu halde çalışmasını sağlamak olarak ifade edilebilir.

Literatürde sınır güvenliğinin sağlanabilmesi için birçok yaklaşım önerilmektedir. T. Lilien ve arkadaşları tarafından yapılan çalışmada insansız hava araçları için tasarlanmış Gezgin Tasarsız Ağ yöntemlerinden birisi olan Fırsatçı Kaynak Kullanımı Ağları (Opportunistic

Resource Utilization Networks) kullanılarak benzetim hazırlanmış ve elde edilen sonuçlar gözlemlenmiştir [7]. D. Choi, B. Ko ve P. Kim tarafından yapılan çalışmada ise web uygulamalarından alınan text verilerinin analiz edilerek tehditlerin tespit edilebileceği öne sürülmektedir [8]. L. Ogiela ve M. Ogilea tarafından yapılan çalışmada kişilerin biyometrik bilgilerinin merkezi bir sistem tarafından tutularak terör durumlarında ilgili şahısların tespitinin mümkün olabileceği ve ek olarak gerçekleşebilecek olaylar için öneriler yapabilen bir sistemin gerekliliği tartışılmıştır [9]. S. Seo ve arkadaşları tarafından yapılan çalışmada kişilerin mobil cihazlarına bir uygulama yüklenerek şüpheli hareketlerin tespit edilmesi ve sonrasında ilgili kurumlara bilgi verilmesi gerektiği tezi savunulmuştur [10]. K. Irgan ve arkadaşları tarafından yapılan çalışmada ise sınır güvenliği için kullanılan cihazlardan elde edilen verilerin düşük gecikme ve düşük enerji tüketimi ile merkeze iletilmesini sağlayan bir kablosuz ağ tasarımı önerilmiştir [11].

Sınır güvenliği ile alakalı çalışmalarda güvenliğin; yeraltı ve yerüstü algılayıcılar, düşük ve yüksek çözünürlüklü kameralar, insansız hava aracı (İHA), uydu, radar gibi birçok farklı disiplinin birlikte çalışması sonucu sağlanabileceği kanısı hâkim olmuştur [12]. Yani bir bölgenin güvenliğinin sağlanması, farklı alanlarda faaliyet gösteren birimlerin ve sistemlerin koordineli çalışmasıyla mümkün olabilir. Buradaki en büyük problem, elde edilen verilerin nasıl, hangi amaçla ve kimler tarafından kullanılacağıdır. PTDS ve diğer sistemler incelendiğinde genel problem, alt programlardan elde edilen verilerin ayrı ayrı ele alınmasıdır. Örnek vermek gerekirse, aktif olarak kullanılan sistemlerde; drone ve İHA'lar pilotlar tarafından uçurulmakta, alınan görüntüler farklı operatörler tarafından anlık olarak incelenmektedir. Ayrıca her İHA ve drone kullanımı için ayrı ayrı pilotlar ve operatörler gerekmektedir. Bunlara hareket algılayıcılar, silah sistemleri gibi birimler de eklendiğinde; harcanan iş gücü, yetkinliğe sahip insan bulma zorluğu, ilişkisiz birimler tarafından karar verme zorunluluğu gibi durumlar ortaya çıkmaktadır. İhtiyaçlar göz önüne alındığında ortak bir akılla hareket eden, öğrenebilen, hatta kendi kendine karar verme yetisine sahip teknolojilerin geliştirilmesi kaçınılmazdır. Bu kazanım için ilk aşama farklı disiplinlerden elde edilen verilerin, sağlam ve bütünleşik bir analiz sürecinden geçirilmesidir. Sınırların denetimi gibi büyük problemlerin çözümüne yönelik veri işlemlerinde;

- Akan veri üzerinde, düşük gecikmeli anlamlandırma/yapılandırma ve temizleme işlemlerini yapma,
- Kayıt ve sorgu işlemlerini yüksek hızda gerçekleştirme,

- Kullanıcılara anlık bilgi/uyarı verebilme ve kayıtların tekrar incelenmesine olanak sağlama olmazsa olmaz özelliklerdir.

Farklı disiplinlerin bir arada çalışması ile veriler, farklı formatlarda, farklı zaman aralıklarında ve farklı büyüklüklerde bir arada bulunabilir. Ancak yüksek hacimli verileri saklamak, yüksek hıza ve değişkenliğe sahip yapılandırılmamış verileri analiz etmek, günümüzde yaygın olarak kullanılan veri tabanı yönetim sistemleri ve mimarileri ile oldukça zordur. Bu karmaşık problemlerin çözümünde Büyük Veri (Big Data) teknolojilerinden yararlanmak büyük avantajlar sağlayabilir. Büyük Veri kavramı verinin, ekonomik, anlamlı, ölçeklendirilebilir bir şekilde kaydedilmesi ve sorgulanması olarak açıklanabilir [13].

Bu tez çalışmasında, Büyük Veri teknoloji ve tekniklerini kullanarak bir boru hattı geliştirilmesi amaçlanmıştır. Uygun teknik ve teknoloji seçimi için ön araştırma yapılmış ve edinilen bilgiler sunulmuştur. Seçim aşamalarının sonrasında geliştirme tamamlanmış ve çalışabilirliği test edilmiştir. Önerilen tasarım Lamda Mimarisi üzerine kurulmuştur;

- Veri iletimi için Apache Kafka,
- Gerçek zamanlı verileri ve yığın verileri işleme için Apache Spark Streaming,
- Verilerin saklanması için Apache Cassandra,
- İşlenen verilerin son kullanıcılara sunulması için NodeJS,
- Nesnelerin haritada 3 boyutlu gösterimi için CesiumJS

kullanılmıştır. Yapılan araştırma ve geliştirme faaliyetleri ile sınır güvenliği problemi ve benzer problemlerin çözümüne yardımcı olacak güvenilir, dayanıklı, ölçeklendirilebilir, uygun maliyetli ve esnek bir ilk örnek elde edilmiştir.

Güvenliğin sağlanabilmesi için, farklı coğrafi alanda farklı tedbirler alınması gerekebilir. En çok nerede sızıntı olduğu, hangi zaman aralıklarında hareketlilik gözlendiği, yeryüzü şekilleri gibi bilgiler, tedbirlerin belirlenmesi ve önceliklendirilmesi açısından önemlidir. Bu çalışma, bu bilgilerin elde edilmesine yardımcı olabilir ve hatta erken uyarı sistemi olarak da kullanılabilir.

Bu tez çalışması beş bölümden oluşmaktadır. Birinci bölüm giriş olarak düzenlenmiş ve bu bölümde çalışma hakkında genel bilgilere yer verilmiştir. İkinci bölümde, Büyük Veri

teknik, araç ve yöntemlerinin ne olduğu anlatılmıştır. Ek olarak çalışma yapılarının nasıl olduğu ve diğer alanlarla olan ilişkilerine değinilmiştir.

Tezin üçüncü bölümünde tasarlanan mimari ve kodlanan uygulama anlatılmıştır. Mimarinin geliştirilmesinde temel alınan yaklaşımlar, öneriler ve çalışmalar detaylı olarak incelenmiştir. Verinin toplanmasından sunulmasına kadar olan her aşama hakkında bilgi verilmiştir.

Dördüncü bölümde yapılan çalışma süresince elde edilen bilgi ve deneyimlere yer verilmiştir. Daha önce yapılan benzer çalışmalarla kıyaslanarak artı ve eksi yönleri tartışılmıştır.

Çalışmanın son bölümünde elde edilen kazanımlara, değerlendirmelere ve önerilere yer verilmiştir. Yapılan çalışmanın sınır güvenliği sağlamasında yardımcı olabileceği ve gelecekte yapılacak çalışmalara yol gösterici olabileceği konusunda bilgi verilmiştir.

2. ARAÇLAR VE YÖNTEM

Büyük Veri son yılların gözde kavramlarından bir tanesidir. Sosyal medya, güvenlik, müşteri odaklı çalışma gibi birçok alanda Büyük Veri teknik ve teknolojilerinden faydalanılmaktadır. Bu sebeple tezin konusu olan; sınır güvenliğinin sağlanmasına yardımcı olacak boru hattı tasarımında, uygun teknik ve teknolojinin seçimi için ön araştırma yapılmıştır. Büyük Veri kavramı detaylı şekilde incelenmiş, örnek çalışmalara da yer verilmiştir.

2.1. Büyük Veri ve Teknikleri

Teknolojinin ilerlemesi bize yeni fırsatlar oluştururken, beraberinde zorlukları da getirmektedir. Her gelişme yeni bir bilgi; beraberinde saklama, paylaşma, analiz etme ve görüntülenmesi problemi ortaya çıkarmaktadır.

Geçmişte Büyük Veri işlemleri ile ilgili teknolojik bir imkân ve ihtiyaç bulunmamasından dolayı günümüzde bu problem yavaş yavaş aşılmaya başlanmıştır. Büyük Veri ve platformlarının oluşumu 5V olarak adlandırılan 5 temel bileşen ile açıklanmaktadır.

- *Volume (Hacim):* International Data Corporation'nin (IDC) 2013 yılında yaptığı çalışmaya göre 2005-2020 yılları arasında küresel veri hacmi 130 exabyte' tan 40.000 exabyte'a çıkacaktır. Yani 2020 yılına kadar her 2 yılda bir, veri hacmi var olan veri hacminin 2 katına çıkacaktır [14]. Bu büyüklükte bir veri yığınının saklanacağı ve yönetilebileceği bir platform gerekmektedir.
- *Variety (Çeşitlilik):* Teknolojinin ilerlemesi farklı formatlara sahip verilerin ortaya çıkmasına olanak sağlamaktadır. Farklı formatlara sahip bir veri yığınının saklanabileceği, yönetilebileceği ve anlamsal olarak bir bütünlük sağlayabileceği bir platform gerekmektedir.
- *Velocity (Hız):* Hacmi artan ve formatları farklı verilerin sisteme hızla alınabilmesi ve sorgulanabilmesi mevcut teknolojiler ile zor ve maliyetlidir. Hacimsel olarak büyük, formatsal olarak farklı tipte verilerin sisteme hızla dâhil edilebileceği ve sorgulanabileceği bir platform gerekmektedir.
- *Veracity (Doğruluk):* Verinin sistem içerisinde kullanımı sırasında gerekli güvenlik

seviyesinde izlenmesi, yetkili kişiler tarafından görülebilir olması ya da gizli kalması, hatalı bir şekilde ya da zararlı yazılımlarla değiştirilmesini engelleyebilecek bir platform gerekmektedir.

- *Value (Değer):* En önemli bileşenidir; tüm bileşenlerin asıl amacı 'Değer' bileşenine hizmet etmektir. Sahip olunan verilerin karar verme sürecinde etkili olmasını ve güncelliğinin devamlı olarak sağlanabileceği bir platform gerekmektedir.

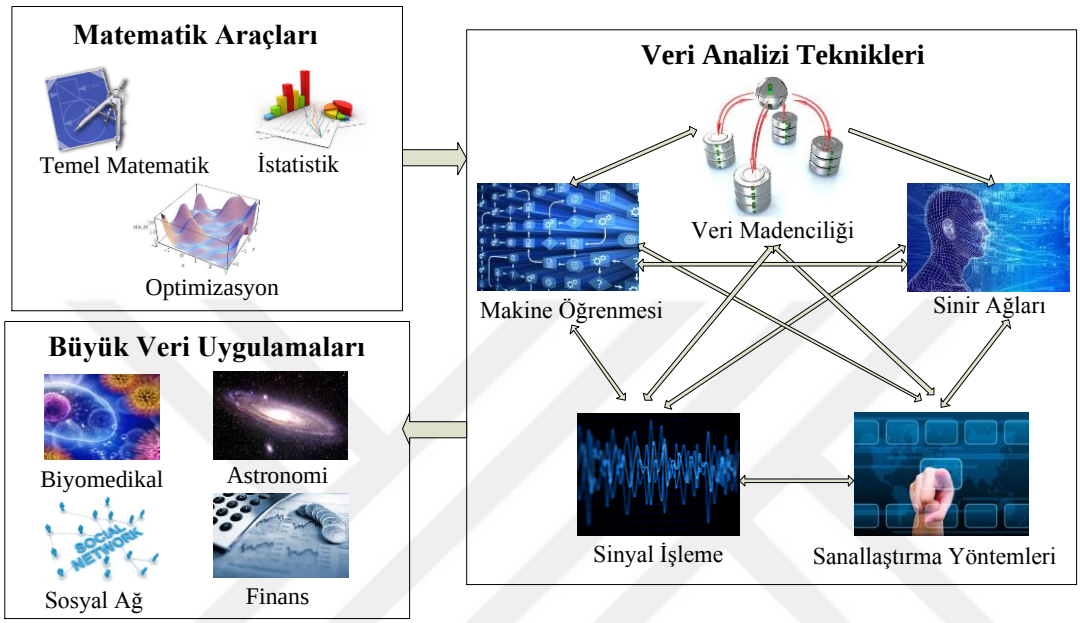
Veri boyutu üstel olarak büyümeye devam etmesine rağmen, günümüzdeki çalışılabilir kapasite, petabytes, exabytes and zettabytes'dan nispeten daha küçük seviyelerinde yapılabilmekte, geniş ve dağınık veri yönetimi ve veritabanı tasarımına ihtiyaç duymaktadır.

Büyük Veri içerisinden değerleri yakalayabilmek ve analiz edebilmek için, yeni teknikler ve teknolojiler geliştirilmelidir. Geliştiricilerin şimdiye kadar çok çeşitli teknikler ve teknolojiler geliştirmelerinin sebebi; Büyük Veri' yi saklamak, analiz edebilmek ve görüntüleyebilmektir. Yine de bunlar, ihtiyaçları karşılamanın çok uzağında kalmıştır. Bu teknik ve teknolojiler, bilgisayar bilimleri, ekonomi, matematik ve istatistik gibi birçok bilim dalıyla kesişmiştir.

Büyük Veri'den anlamlı veriler elde edebilmek için, çeşitli araçlara ihtiyaç vardır. Şu anki araçlar, yığın işleme aracı (batch processing tools), akış işleme aracı (stream processing tools), etkileşimli analiz araçları (interactive analysis tools) olarak 3 ana başlık altında gruplandırılır.

Büyük hacimli verileri belirli bir çalışma süresi içerisinde, verimli bir şekilde işleyebilmek için özel tekniklere ihtiyaç vardır. Büyük Veri teknikleri, doğal olarak belirli uygulamalar tarafından yönlendirir. Örneğin Wall-Mart, büyük hacimli işlenmiş verilerin içerisinden örnek çıkarmak için, makine dili ve istatistiksel teknikler kullanmaktadır. Bu örnekler, fiyat stratejilerinde ve reklam kampanyalarında yüksek rekabet oluşturur. Taobao (Çin'in eBay'i), büyük akımlı veri madenciliği kullanan bir sistem edinmiştir. Bu sistem, kullanıcı alışkanlıklarını ve kullanıcıların site üzerindeki hareketlerini kaydederek, kullanıcının karar verme sürecini destekleyen işe yarar veriler elde etmektedir.

Büyük Veri birçok bilim dalı ile birlikte gelişme göstermektedir. Bunlar istatistik, veri madenciliği, özdevimli öğrenme, sinir ağları, sosyal ağ analizleri, sinyal işleme, örüntü tanıma (pattern recognation), optimizasyon yöntemleri, görüntüleme yaklaşımları gibi dallardır [15].



Şekil 2.1. Büyük Veri teknikleri

Şekil 2.1. Büyük Veri teknikleri'nde görüldüğü üzere optimizasyon metotları, fizik, biyoloji, mühendislik, ekonomi gibi birçok alandaki hesaplanabilir problemlerin çözümü için uygulanır. Genetik programlama, evrimsel programlama ve parçacık sürüsü optimizasyonu, stokastik optimizasyon yöntemleri arasında yer alan özel ve kullanışlı optimizasyon teknikleridir. Bununla birlikte, genellikle hafıza açısından fazla karmaşık ve zaman alan yöntemlerdir. Birçok araştırma, büyük ölçekli optimizasyon işlemlerinin eş-evrimsel (co-evolutionary) algoritmalar iş birliği ile yükseldiğini göstermiştir. Veri azaltma ve paralelizasyon da optimizasyon problemlerinde kullanılabilecek alternatif yöntemlerdir [16].

İstatistik, verileri toplayan, organize eden ve yorumlayan bilim dalıdır. Sayısal açıklamalar, istatistik yöntemleri ile sağlanır. Ancak birçok standart istatistik tekniği genelde Büyük Veri'nin çalışma matığına yeterince uymamaktadır. Bu yüzden birçok araştırmacı, klasik yöntemlerin geliştirilmesiyle elde edilen yöntemleri ya da tamamen yeni metotları önermektedir. Veri odaklı istatistiksel analizde bir diğer eğilim de istatistiksel

algoritmaların paralel uygulamalarına odaklanmaktır. “İstatiksel hesaplama” ve “istatiksel öğrenme” bu konunun en önemli alt dallarıdır [17].

Veri madenciliği verilerden, kümeleme analizi, sınıflandırma, regresyon gibi yöntemlerle işe yarar bilgiler elde etmeyi sağlar. İstatistik ve makine öğrenmesinden türetilen bir yöntemdir. Büyük Veri madenciliği standart veri madenciliği algoritmalarına kıyasla daha zordur. Büyük Veri’yi kümelemenin doğal yolu, mevcut yöntemlerden türetilen yöntemleri kullanmaktır ve bunun sonucunda çok büyük iş yükü ile başa çıkmak zorunda kalınır. Genetik algoritmalar da optimizasyon kriteri olarak kümeleme yöntemini kullanır [17].

Makine öğrenmesi, bilgisayarların ampirik verilere dayalı davranışlarını geliştirmesine olanak veren algoritmalar tasarlamayı amaçlamış yapay zekânın önemli bir ürünüdür. Makine öğrenmesinin en karakteristik özelliği, bilgiyi keşfetmek ve otomatik olarak akıllı kararlar almaktır. Büyük Veri söz konusu olduğunda bununla başa çıkabilmek için, makine öğrenmesi algoritmaları kullanılmalıdır. “Map/Reduce”, “DryadLINQ” ve “IBM paralel makine öğrenmesi tool box” gibi altyapılar makine öğrenmesini yükseltme kapasitesine sahiptir. Örneğin “Support Vector Machine (SVM)”, sınıflandırma ve regresyon problemlerinde kullanılan, hafıza kullanımı ve hesaplama süresi gibi ölçülebilirlik kriterlerinden mustarip çok temel bir algoritmadır. Son zamanlarda, hafıza kullanımını ve zamanı düşüren paralel SVM (PSVM) ortaya çıkmıştır [18].

Yapay sinir ağları, geniş bir uygulama kapsamına sahip olgun bir tekniktir. Bunun başarılı uygulamaları örüntü tanıma, görüntü analizi, adaptif kontrol ve diğer alanlarda görülebilir. Günümüzde yapay zekâ için kullanılan birçok yapay sinir ağı, istatistiksel tahminlere, sınıflandırma optimizasyonuna ve kontrol teorisine dayanmaktadır. Daha çok gizli katmanın ve düğüm noktasının daha yüksek bir hatasızlık getirdiği kabul edilen bir gerçektir. Bununla birlikte, sinir ağlarındaki karmaşıklık, öğrenme süresini arttırmaktadır. Buna bağlı olarak sinir ağlarındaki öğrenme zamanı, hafıza kullanımı ve zaman açısından Büyük Veri’ ye göre daha fazladır. Büyük ölçekli verilen sinirsel işlenmesi, genellikle büyük ağlara öncülük eder. Bu durumda, iki problemle karşılaşılabilir. Bunların ilki geleneksel eğitim algoritmaların performansının çok düşmesi, ikincisi ise hafıza kısıtlamalarının artmasıdır. Bu durumlarda uygulanabilecek, yaygın iki yöntem vardır. Birincisi, sinir ağının yapısı aynı kalacak şekilde, bazı örnekleme metotlarıyla verinin

boyutunu azaltmak, ikincisi ise paralel ve dağıtık yöntemlerle, sinir ağının yapısını büyütmektir.

Görüntüleme yaklaşımları, verileri anlayabilmek için tablo, görüntü, diyagram oluşturmak gibi tekniklerdir. Büyük Veri karmaşık bir yapıya sahip olduğu için, verilerin görüntülenmesi alışılmış küçük veri kümelerinde olduğu kadar kolay değildir. Şu anda, geleneksel görüntüleme tekniklerinin uzantıları mevcuttur ancak bunlar yeterlilikten çok uzaktadır. Büyük ölçekli veri görüntüleme işleminde ise birçok araştırmacı, asıl veri işleme kısmından önce, verilerin özniteliğini ve geometrik modelini çıkararak verinin boyutunu küçültmektedirler.

Sosyal ağ analizi, modern sosyolojinin anahtarı olarak ortaya çıkmıştır ve sosyal ilişkileri ağ teorisi içerisinde inceler. Sosyal ağ analizi; sosyal sistem tasarımı, insan davranış modeli, sosyal ağ görüntülenmesi, sosyal ağ evrim analizi ve grafik sorgularını içerir. Son zamanlarda, çevrimiçi sosyal medya ve sosyal medya analizi popüler duruma geldi. Sosyal ağ analizi ile ilgili en büyük engellerden biri Büyük Veri' nin büyüklüğüdür. Milyonlarca ya da milyarlarca nesnenin bağlı olduğu bir ağın analizini yapmak genellikle çok maliyetlidir.

Yüksek seviyeli Büyük Veri teknolojileri; dağıtık dosya sistemleri, dağıtık hesaplama sistemleri, düzenli paralel-işleme sistemleri, veri madenciliği tabanlı dağıtımlı hesaplama, bulut tabanlı depolama ve hesaplama kaynaklarını içerir.

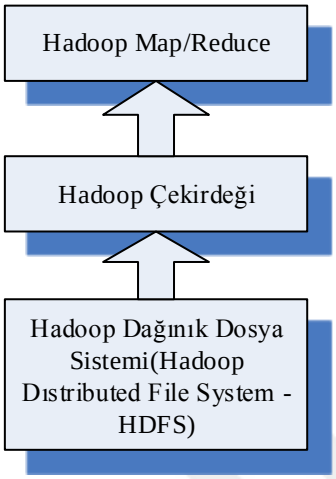
2.1.1. Yığın işlemeye dayalı Büyük Veri araçları

En ünlü ve en güçlü yığın işleme tabanlı Büyük Veri araçlarından bir tanesi Apache Hadoop'tur. Diğer Büyük Veri uygulamalarına, altyapı ve platform sağlar. Hadoop üzerinde yapılanan bazı Büyük Veri sistemlerinin farklı alanlarda özel kullanımları vardır. Örneğin veri madenciliği ve makine öğrenmesi, iş ve ticaret alanında kullanılır.

Apache Hadoop ve Map/Reduce

Apache Hadoop, dağıtık veri-yoğunluklu uygulamaları destekleyen en iyi yapılandırılmış yazılım platformlarından birisidir. Map/Reduce (Planla/Küçült) olarak adlandırılan,

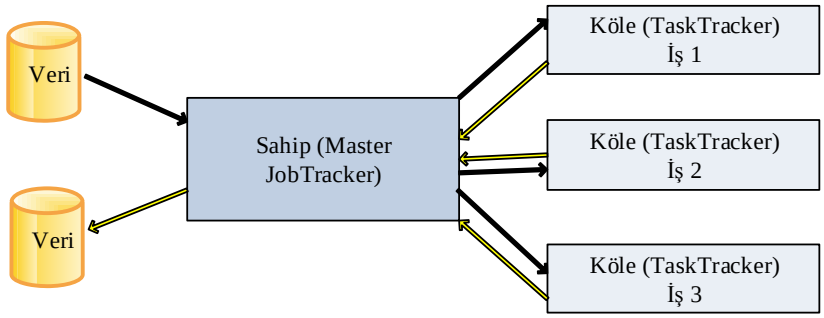
hesaplanabilir örnekleri uygular. Şekil 2.2. Hadoop sistem mimarisi'nde belirtildiği üzere Apache Hadoop platformu; Hadoop çekirdeği, Map/Reduce ve Hadoop dağıtık dosya sistemi (Hadoop Distributed File System - HDFS) içerir.



Şekil 2.2. Hadoop sistem mimarisi

Map/Reduce, Google'ın öncülük ettiği ve Yahoo'nun geliştirdiği büyük hacimli veri kümelerini üreten ve işleyen bir programlama modelidir. Map/Reduce böl ve yönet metodu kullanır. Karmaşık problemleri, ölçeklenebilir ve direkt çözülebilir hale gelene kadar sürekli parçalayarak birçok alt probleme ayırır. Sonrasında, bu alt problemler ayrı ayrı ve paralel olarak çözülür. Son olarak bu alt problemlerin çözümleri birleştirilerek ana problemin çözümü olarak sunulur. Böl ve yönet metodu iki adımda gerçekleşir: Map ve Reduce adımı. Hadoop yapısında, iki tür düğüm vardır. Bunlar sahip(master) düğüm ve işçi(worker) düğümlerdir. Master düğüm Map adımında, girdiyi alarak daha küçük alt problemlere böler bunları işçi düğümlere dağıtır. Daha sonra, Reduce adımında tüm alt programların çözümleri, efendi düğüm tarafından toplanarak birleştirilir ve çıktı oluşturur. Map/Reduce yapısında, efendi (master - JobTracker) ve köleler (slave - TaskTracker) vardır (Şekil 2.3. Map/Reduce genel bakış). Sahip düğüm, iş çizelgesini doldurur ve kölelere (TaskTracker) görev dağıtımını yapar. Köleler, sahip tarafından atanan görevleri yerine getirir. Sistem çalışmaya başladığı andan itibaren, sahip düğüm bütün düğümleri izlemeye başlar. Eğer düğümlerden birisi verilen görevi yerine getiremezse, sahip düğüm bu düğümden ya da diğer düğümlerden başarısız olan görevin yeniden yerine getirilmesini ister. Pratikte, uygulamalar girdi dosyalarını, çıktı konumlarını ve Map/Reduce fonksiyonlarını istemci ara yüzleri üzerinden belirler. İş yapılandırılması açısından, bu parametreler önemlidir. Bundan sonra Hadoop iş istemcisi, görevi girer ve JobTracker'ı

yapılandırır. JobTracker gerekli bilgileri aldıktan sonra, görevleri dağıtır, görevlerin zamanını belirler ve izler, teşhis bilgilerini sağlar [19].



Şekil 2.3. Map/Reduce genel bakış

Dryad

Dryad, çok küçük veri kümelerini işleyerek büyük veri kümeleri elde etmeye yarayan bir diğer programlama modelidir. Dryad çalışma alt yapısı, bilgisayarın düğüm kümelerine dayanır ve programcılar programlarını çalıştırmak için bunları kaynak olarak kullanır. Dryad programcılar, her biri çoklu işlemci ya da çekirdek içeren binlerce makineyi kullanabilirler. Bu programcıların, eş zamanlı programlama ile ilgili hiçbir şey bilmesine gerek yoktur. Dryad uygulamaları, hesaplama düğümlerini ve iletişim kanallarını içeren yönlendirilmiş hesaplama grafiğini çalıştırırlar. Dryad programcıları, birçok sıralı program yazar ve bunları tek yönlü kanallar aracılığı ile birbirine bağlar. Dryad ın görevi, grafik üretmektir ve bunu yönlendirilmiş çevrimsiz her grafiği birleştirebilme kapasitesine sahiptir. Bu grafikler, hesaplama sırasında beklenmedik durumlar oluşursa sonradan güncellenebilme özelliğine sahiptir. Dryad, görev grafiği üretme, mevcut makinelerdeki süreçleri planlama, kısa süreli hatalarla başa çıkma, performans ölçümlerini toplama, görevleri görüntüleme gibi birçok fonksiyona sahiptir. Her Dryad görevinde olduğu gibi merkezde bir yönetici vardır. Çalışmayı kontrol etmek için küçük veri kümelerini kullanır [20].

Apache Mahout

Apache Mahout, büyük ölçekli ve anlamlı veri analizi uygulamalarına ticari ve ölçeklenebilir makine öğrenmesi teknikleri sağlamayı amaçlar. Google, Amazon, Yahoo, Facebook ve Twitter gibi birçok büyük şirket, projelerinde ölçülebilir makine öğrenmesi

algoritmaları kullanmaktadır. Birçok projede Büyük Veri ile ilgili problemler oluşmakta ve Apache Mahout bunları hafifleten araçlar sağlamaktadır. Mahout çekirdek algoritması, Hadoop platformunun en üstünde Map/Reduce yapısı üzerinden çalışır ve kümeleme, sınıflandırma, örüntü madenciliği, boyut küçültme, evrimsel algoritmalar içerir. Bu algoritmalar; yüksek performans ve kapasite için optimize edilmiş, iyi bir şekilde tasarlanmış kütüphane içerisinde bulunur. Bu kütüphane, dağıtık olmayan algoritmalar da içerir [21].

Pentaho Business Analytics

Pentaho, Büyük Veri için kullanılabilecek başka bir platformdur. Hem biçimlendirilmiş hem de biçimlendirilmemiş verilerden rapor oluşturabilir. Penatho, Büyük Veri'nin daha çok iş yaşamına uygun olarak geliştirilmiş bir platformudur. İş adamları için kolay erişim, entegrasyon, görüntüleme ve veri keşfi gibi servisler sağlar. Bu nedenle Penatho kullanımı, iş kullanıcılarının veriye dayalı kararlar vermesini pozitif bir katkıda bulunur. Güvenlik, ölçüle bilirlik, erişilebilirlik gibi önemli özellikleri içerisinde gömülü olarak bulundurur. Kullanıcılar, Penatho tarafından sağlanan web tabanlı ara yüzü kullanarak verilerine erişebilirler. Wizard-based tarzı bir yaklaşımla, kullanıcılar verilerin detayını daha iyi anlayabilir ve karar verme süreçlerini hızlandırabilir. Penatho tarafından geliştirilen Kettle, Penato Data Integration gibi grafiksel programlama ara yüzleri, büyük verileri işlemek için güçlü araçlardır [22].

Skytree Server

Skytree Server, büyük veri kümelerini doğru ve hızlı bir şekilde işlemek için tasarlanmış ilk genel amaçlı makine öğrenmesi ve gelişmiş analiz sistemidir. Birçok gelişmiş ve çok yönlü makine öğrenmesi algoritması sunar. Kullanımı çok kolaydır, kullanıcı sadece komut satırına doğru komutu yazmalıdır. Skytree Server in kullanıldığı beş farklı kullanım alanı vardır. Bunlar isimlendirme, tavsiye sistemi, anormalliklerin/aykırılıkların tespit edilmesi, tahmine dayalı analitik, kümeleme ve market bölümlemesi, benzerliklere göre arama yapılmasıdır. Sytree daha çok gerçek zamanlı analitiklere odaklanmıştır. Ayrıca ilişkisel veri tabanlarından, HDFS'den, dosyalardan, ortak istatistiksel paketlerden ve makine öğrenmesi kütüphanelerinden aldığı biçimlendirilmiş ve biçimlendirilmemiş verilerle başa çıkabilir [23].

Tableau

Tableau'nun büyük ölçekli veri kümelerini işlemek için kullandığı Tableau Server, Tableau Desktop ve Tableau Public olarak isimlendirilen üç ana ürünü vardır. Tableau Desktop veriyi kolay bir şekilde ve farklı biçimlerde görüntüleyebilmek için kullanılan bir görüntüleme aracıdır. Tableau Desktop, kullanıcının verileri birbirleriyle karıştırabilmesi (mix) için optimize edilmiştir. Tableau Server tarayıcı tabanlı analitikler sağlayan bir sistem, Tableau Public ise etkileşimli görseller yaratmak için kullanılan bir sistemdir. Tableau aynı zamanda Hadoop alt yapısına gömülüdür. Bu sayede kullanıcı ve Büyük Veri uygulamaları arasında etkileşimli bir mekanizma sağlar [24].

Karmasphere Studio and Analyst

Karmasphere, iş verilerinin analiz edilmesi için kullanılan Hadoop tabanlı başka bir Büyük Veri platformudur. Büyük Veri'nin hızlı bir şekilde mantıksal çözümlemesinin yapılması, kendin-al (selfservice) erişimin etkili ve işbirlikçi yöntemlerle yapılabilmesi için yeni yaklaşımlar sağlar. Karmasphere, Hadoop platformu için tasarlanmıştır ve kullanıcıların Büyük Veri'yi işlemesi ve iş akışını sunması için kullanıcı dostu bir ara yüz sağlar. Muazzam büyüklükte veriler içerisindeki işi çıkararak, tekrar tekrar analizini yapabilir, görüntüleyebilir ve raporlayabilir. Karmasphere Studio, Eclipse üzerine geliştirilmiş bir eklentidir. Kullanıcılar Hadoop işlerini bu iyi tasarlanmış ve entegre olmuş geliştirme ortamı ile kolaylıkla yazabilir ve uygulayabilir. Karmasphere Analyst, Karmasphere tarafından Hadoop kümeleri üzerine tasarlanan ve analitik işlemeyi arttırmayı amaçlayan bir Büyük Veri aracıdır. Buna ek olarak, Hadoop kümeleri üzerinde bulunan biçimlendirilmiş ve biçimlendirilmemiş verileri işlemek için Hive projesi içerisine gömülmüştür. Teknik analistler, SQL programcıları ve veri tabanı yöneticileri Hadoop aracılığı ile grafiksel ortamı tecrübe edebilirler [25].

Talend Open Studio

Talend Open Studio, kullanıcılara analizlerini görsel olarak yönetebilmeleri için grafiksel ortam sağlayan açık kaynak kodlu bir Büyük Veri uygulamasıdır. Apache Hadoop tarafından geliştirilmiş ve HDFS, Pig, HCatalog, HBase, Sqoop ve Hive eklentilerini içerir. Kullanıcılar, Büyük Veri ile ilgili problemlerini karmaşık Java kodları yazmalarına gerek

kalmadan bu platformu kullanarak çözebilirler. Kullanıcılar Talend Studio ile çeşitli ikonları sürükleyip bırakarak kendi görevlerini yaratabilirler. Kullanıcı, bileşenlerin ne yaptığını ya da yapmadığını öğrenip kendini rahat hissettikten sonra blokları görsel olarak birbirine bağlamak daha kolay olur. Görsel programlama çok yüksek bir hedef olarak görünebilir ancak bunun derinlemesine anlaşılabilmesi için, ikonlar hiçbir zaman yeterli detayı vermez [26].

2.1.2. Akış işleme için kullanılan Büyük Veri araçları

Hadoop, büyük boyutlu verileri paralel bir şekilde işlemek için kullanılan iyi bir uygulamadır. İş yükünü, farklı makinelere dağıtmayı sağlar. Ancak, Hadoop yığın işleme (batch processing) için tasarlanmıştır. Hadoop genel amaçlı bir platformdur; ancak uygulama boyunca, yüksek gecikmeler yaşandığı için gerçek zamanlı ve yüksek performans gerektiren uygulamalar için uygun değildir. Log dosyalarının işlenmesinde, sensörlerin kullanıldığı endüstri kollarında, M2M ve telematik gibi büyük boyutlu veri akışlarının desteklenmesinde gerçek zamanlı yanıtlar alınmalıdır. Akış halindeki Büyük Veri, yüksek hacim, yüksek hız ve karmaşık veri tiplerine sahiptir. Gerçek zamanlı işlemenin önemli olduğu uygulamalar Hadoop'un Map/Reduce yetenekleri ile yapılmaya çalışıldığında, hız ve zaman problemi oluşturabilmektedir. Bu yüzden, SQLstream, Storm and StreamCloud uygulamaları özellikle akış halindeki verilerin gerçek zamanlı mantıksal analizi için tasarlanmıştır. Gerçek zamanlı işlemenin anlamı, veri işleme sonucunun zamanında veya çok küçük gecikmelerle verilmesidir. Büyük Veri genellikle tek bir depolama biriminde değil, dağıtık bir şekilde depolanır. Map/Reduce yapısında, Map fazı bittikten sonra Reduce fazı başlayabilir. Ancak genelde, Map fazında üretilen orta ölçekli veriler, bir sonraki faz için gönderilmeden önce diske kaydedilir. Tüm bunlar, işlemlerde sık sık gecikmelerin yaşanmasına sebep olur. Hadoop'un tüm bu karakteristik özellikleri, bunun gerçek zamanlı mantıksal analizlerde kullanılmasını imkânsız kılar [27].

Akış işleme için kullanılan en ünlü platformlardan biri Spark'tır. Aşağıdaki tabloda platformların amaç ve çalışma mantıklarına yer verilmiştir:

Çizelge 2.1. Akış işleme tabanlı Büyük Veri araçları

Adı	Kullanım Amacı	Avantajlar
Storm	Gerçek zamanlı hesaplama sistemleri	Ölçeklenebilir, hızlı, hata toleranslı, dağıtık, kolay kullanım ve yönetme
S4	Verilerin bağımsız olarak işlem akışı	Dağıtık, ölçeklenebilir, hata toleranslı, portatif platform
SQLstream	Sensorler ve telematik uygulamalar	SQL tabanlı, gerek zamanlı veri gönderme Büyük Veri platformu
Splunk	Veri toplama	Kolay ve hızlı kullanım, dinamik ortam
Apache Kafka	Dağıtık pub-sub sistemleri	Veri iletiminde yüksek performans, açık kaynak kod
SAP Hana	Gerçek zamanlı iş platformu	RAM üzerinde hızlı işlem, gerçek zamanlı analiz
Spark	Gerçek zamanlı ve yığın veri hesaplama sistemleri	Ölçeklenebilir, hızlı, hata toleranslı, dağıtık, kolay kullanım ve yönetme, RAM üzerinde hızlı işlem, gerçek zamanlı analiz
Microsoft Azure Event Hub	Dağıtık pub-sub sistemleri	Ölçeklenebilir, hızlı, hata toleranslı, dağıtık, ücretli, gerçek zamanlı analiz
Microsoft Azure Stream Analystics	Gerçek zamanlı ve yığın veri hesaplama sistemleri	Veri iletiminde yüksek performans, açık kaynak kod
Amazon Kinesis Stream	Dağıtık pub-sub sistemleri	Veri iletiminde yüksek performans, ücretli

Storm

Storm, limitsiz veri akışını işlemek için kullanılan dağıtık ve hata toleranslı gerçek zamanlı hesaplama sistemidir. Storm; yığın işlemek için tasarlanmış olan Hadoop un tersine, özellikle gerçek zamanlı işlemeler için tasarlanmıştır. Aynı zamanda kurulumu ve yönetmesi çok kolaydır; tüm verinin işleneceğini garanti eder. Storm, her düğümünde saniyede bir milyonun üzerinde veri demeti işleyebilir. Bu nedenle gerçek zamanlı mantıksal analiz, etkileşimli işletim sistemi, çevrimiçi makine öğrenmesi, sürekli hesaplama gibi birçok uygulamayı bünyesinde bulundurur. Storm kümesi, görünüm olarak Hadoop kümesine benzer. Oysaki Storm kullanıcıları farklı Storm görevlerinde farklı topolojiler çalıştırır. Hadoop platformu ise, eş uygulamalarda aynı Map/Reduce işlerini uygular. Map/Reduce görevleri ve topolojileri arasında birçok fark vardır. Bunlardan en önemlisi, kullanıcı durdurana kadar Map/Reduce işinin yapılmaya devam edilmesidir. Storm da gerçek zamanlı hesaplamalar yapabilmek için, kullanıcının farklı topolojiler

yaratması gerekmektedir. Bu topoloji bir hesaplama grafiğidir ve her hangi bir programlama dilinde yazılıp kaydedilebilir. Topolojide, ‘spout’ ve ‘bolt’ olarak adlandırılan iki çeşit düğüm vardır. Spout, akışın kaynağını oluşturan ve grafiğin başlangıç noktalarından birisi olan düğümdür. Bolt, girdi akımlarını işleyerek yeni çıktı akışları oluşturur. Topolojideki her düğüm bir işlem mantığı içerir ve düğümler arasındaki bağlantılar, verinin düğümler arasında nasıl işleneceğini gösterir. Dolayısıyla akıştaki dönüşümün nasıl olacağı grafiklerle ifade edilir ve topolojideki her düğüm paralel olarak çalışır. Storm, iki çeşit düğüm içerir. Bunlardan bir tanesi sahip düğüm, diğerleri çalışan düğümlerdir. Master düğüm ve çalışan düğümler Nimbus ve Supervisor olmak üzere iki hizmet sağlarlar. Bunlar Map/Reduce yapısında bulunan JobTracker ve TaskTracker ile benzer fonksiyonlara sahiptir. Nimbus, Storm kümesi boyunca kodu dağıtır, çalışan düğümlere atanan görevlerin zamanlamasını belirler ve tüm sistemi izler. Eğer bir hata oluşursa, Nimbus bunu tespit eder ve ilgili görevi yeniden çalıştırır. Supervisor, Nimbus’un verdiği talimatları uygulayarak işlemleri başlatır veya durdurur. Tüm hesaplama topolojisi, dağınık ve parçalı olarak işlere bölünmüştür ve her iş topolojinin bir parçasını tamamlar. Zookeeper ise sistemin koordine edilmesinde önemli bir rol oynar. Nimbus ve Supervisor tarafından gerçekleştirilen tüm durumlar, Zookeeper tarafından yerel diske kaydedilir [28].

Apache Spark

Apache Spark hızlı, kullanımı kolay ve karmaşık analizleri yapabilme kabiliyetine sahip, açık kaynak kodlu, küme hesaplama yapısına sahip büyük veri işleme yapısıdır. Yığın işleme ve gerçek zamanlı akış işleme yetenekleri oldukça gelişmiştir. Planla/Küçült (Map/Reduce) programlama modeli üzerine kurgulanmıştır. Spark kümesi, Apache Mesos, Hadoop YARN ve Amazon EC2 kullanılarak yönetilebilmesinin yanı sıra, kurulumda hali hazırda gelen Standalone Scheduler ile de yönetilebilir. Efendi-köle (master-slave) prensibi ile çalışır. Üzerinde işlem yapılacak mikro yığınlara Esnek Dağıtık Veriseti (Resilient Distributed Dataset**,** kısaca RDD) denir. Her bir RDD değiştirilemeyen, paralel olarak işlenebilen bir veri listesidir. Her dönüşüm işlemi için yeni bir RDD oluşturulur ve asıl RDD’nin korunması sağlanır. Spark’ın özellikleri aşağıdaki gibi sıralanabilir [29];

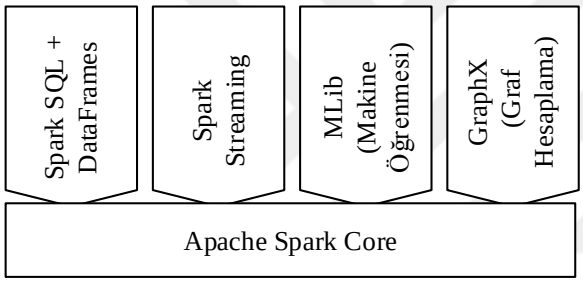
- Map/Reduce (Planla/Küçült) işlemlerine ek olarak SQL çalıştırma, makine öğrenmesi (machine learning) ve graf (graph) veriler üzerinde işlem yapma yeteneklerine sahiptir,
- Scala programlama dili ile kodlanmıştır. Java, Python, Scala ve R programlama

dillerini destekler,

- bellek üzerinde (in-memory) veri saklama ve gerçek zamanlı veri işleme yeteneklerinden dolayı diğer büyük veri teknolojilerine göre daha hızlıdır,
- tembel çalıştırma (lazy evaluation) özelliğinden dolayı iş akışındaki sorgular optimize edilebilir,
- sorgu sonuçları bellekte tutulur; bu sayede aynı veriseti üzerindeki işlemler hızlı sonuçlanır.

Spark Bileşenleri

Apache Spark, Şekil 2.4. Spark bileşenleri'nde belirtildiği gibi Spark Core ve bir grup kütüphanenin bileşiminden oluşmaktadır.



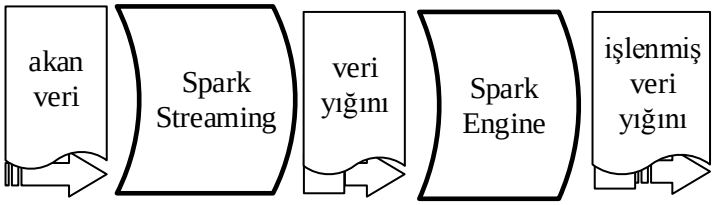
Şekil 2.4. Spark bileşenleri

Spark Core: Verinin paralel işlenmesinden ve görev dağılımından sorumlu mekanizmadır. Spark Core;

- bellek yönetimi ve hata kurtarma,
- küme hesaplamalarında paralel işleme, görev dağılımı ve takibi,
- harici disklerle iletişim

gibi işlemlerden sorumludur.

Spark Streaming: Farklı kaynaklardan alınan gerçek zamanlı veriler üzerinde işlem yapmayı sağlayan bileşendir. Akan her veri için DStream (Discretized Stream) oluşturulur. DStream, üzerinde işlem yapılan mikro yığın (RDD) serisi olarak ifade edilebilir. Şekil 2.5. Spark Streaming'in yapısı'nda gösterildiği üzere akan veri belirli bir süre biriktirilerek veri yığını elde edilir. Daha sonra bu yığınlar Spark Engine tarafından alınarak işleme tabi tutulur.



Şekil 2.5. Spark Streaming'in yapısı

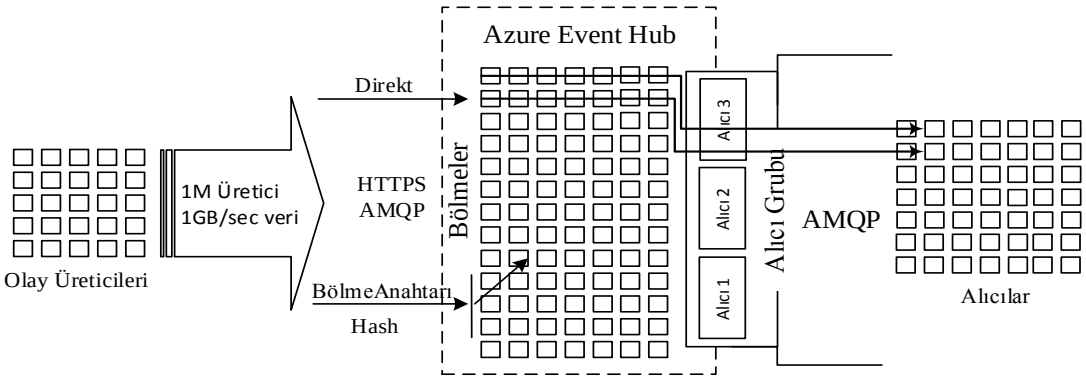
SparkSQL: SQL ve HQL (Hive Query Language) kullanarak sorgulamayı destekleyen Spark bileşenidir.

MLib: Sınıflandırma, regresyon, kümeleme, filtreleme gibi yetenekleri üzerinde barındıran makine öğrenmesi kütüphanesidir.

Microsoft Azure Event Hubs

Event Hub saniyede milyonlarca olayı anlık olarak alabilen yayımla-abone ol hizmetidir. Akış yeteneğine sahiptir; bu yetenek toplanan verilerin farklı uygulamalar tarafından olay akışı şeklinde tüketilmesine olanak verir. HTTP protokolü üzerinde REST API ile erişilebilir ve aynı zamanda AMQP desteği mevcuttur. Olaylar bölmeler içinde gruplandırılır ve her olay bölmesi sırayla kuyruğa eklenir. Bölmeler bir kez oluşturulduktan sonra tekrar değiştirilemez [30].

Olayları yayımlayanlara Event Publisher (Olay Üreticisi), tüketenlere ise Event Consumer (Olay Alıcısı) denir. Her alıcı bölme içerisindeki akış sırasını (offset) ayarlar, bu değer sıra numarası ya da zaman aralığı olabilir.



Şekil 2.6. Azure Event Hub

Akış sırasında problem gelmesi olasılığına karşı kontrol noktaları (checkpoint) belirlenebilir. Problem anında bir önceki kontrol noktasına dönülerek akış devam ettirilebilir. Bir olay maximum 256KB büyüklüğünde olabilir. Bölme sayısı ise maximum 32 tanedir.

Ücretlendirme, iletilen veri, olay sayısı ve olayların saklanacağı gün (ilk gün ücretsiz) üzerinden yapılır.

Microsoft Azure Stream Analytics

Hızlı geliştirme ve kurulma özelliklerinde; sensor, uygulama, cihaz vb gibi gerçek zamanlı üretilen veriler üzerinde analiz yapabilen yapıdır. Uygulama geliştirme sürecinde az uğraş ile kompleks yapılar kurulabilir.SQL tabanlı sözdizimine sahiptir. Anlık olarak milyonlarca olayı alarak gerçek zamanlı işleyebilir ve birbirleri arasında kıyaslama yapabilir. Düşük gecikmeli, yüksek veri kapasiteli ve esnektir. Veri kaybı olmaksızın tüm verileri işleyeceğini garanti eder [31].

Stream Analytics işlediği veri sayısı ve kullandığı donanım için saatlik olarak ücretlendirme yapar.

Amazon Kinesis Streams

Amazon Kinesis Streams gerçek zamanlı akan veri üzerinde analiz yapmaya yardımcı olur. Farklı kaynaklardan aldığı büyük hacimli verileri kolaylıkla iletebilir. Kinesis, Amazon EC2 gibi bir çok uygulama ile entegre olabilir. 50 KB boyutlarındaki olaylar üzerinden çalışabilir. Benzer yapılarda olduğu gibi burada da veri gönderenlere ‘producer’, tüketenlere ‘consumer’ denir. Kinesis üzerindeki kayıtlar, sıra numarası, bölme anahtarı ve verinin bir bölümünü içerecek şekilde dizayn edilir. Akış kayıtları oluşturulduğunda bir daha değişmez. Kayıtlar, Kinesis üzerinde en az 24 saat olacak şekilde tutulur [32].

Akış içerisindeki veri kaydı gruplarına ‘shard’ denir. Bir akış bir veya birden çok ‘shard’ dan oluşabilir. Ücretlendirme Microsoft Azure Event Hub gibi gönderilen veri trafiği, olay sayısı ve olayların saklanacağı gün (ilk gün ücretsiz) üzerinden yapılır.

S4

S4 sınırsız veri akımlarını işleyebilen, dağıtık, ölçeklenebilir, hata toleransı olan, takılıp/çıkarılabilir genel amaçlı bir hesaplama platformudur. 2010 yılında Yahoo tarafından başlatılmıştır ve 2011 yılından beri Apache Incubator projesi halindedir. S4 dayanıklı, ölçülebilir, bir merkezden yönetilmek zorunda olmayan, genişletilebilir, takım yönetimine sahip olma gibi özellikleri olan ve programcıların kolaylıkla geliştirme yapmasını sağlayabilen bir programdır. S4 ün çekirdek katmanı Java ile yazılmıştır. S4'ün çalışma şekli, büyük ölçekli veri akımlarını dinamik olarak işleyebilmek amacıyla modüler olarak tasarlanmıştır. S4 Yahoo'nun üretim sisteminin içine yerleştirilmiştir ve binlerce arama sorgusu yaparak diğer uygulamalar da iyi performans göstermiştir [33].

SQLstream s-Server

SQLstream, büyük ölçekli veri akımlarını gerçek zamanlı olarak işlemek için tasarlanmış bir diğer Büyük Veri platformudur.

SQLstream memory işlemcilerini kullandığı için çok hızlı çalışır, aynı zamanda "No Database" teknolojisi olarak adlandırılır. Veriler diske kaydedilmez. Bunun yerine, gelen veriler "StreamingSQL" sorguları kullanarak memory içerisinde işlenir. StreamingSQL, çok çekirdekli hesaplamanın avantajları kullanılarak standart SQL den geliştirilmiştir ve paralel olarak işlenmiş büyük verileri arşivlemek için kullanılır [34].

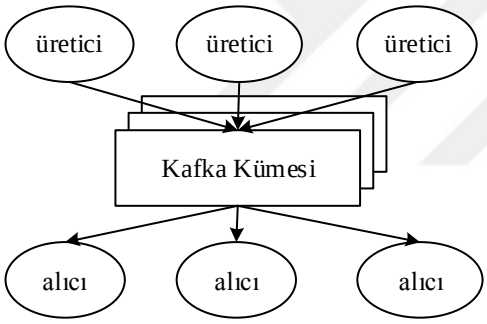
Splunk

Makine üretimi Büyük Veri'lerden, anlamlı veriler çıkartan gerçek zamanlı bir Büyük Veri platformudur. Heroku, Amazen, Senthub gibi birçok büyük firma tarafından kullanılmıştır. Splunk, Büyük Veri ile bulut teknolojilerini kombinleyerek kullanıcıların makine üretimi verilerini, bir web ara yüzü üzerinde aramasını, görüntülemesini ve analiz etmesini sağlar. Sonuçları grafikler, raporlar ya da uyarılar şeklinde sunar. Diğer akış işleme araçlarından çok farklıdır. Makine üretimi olan biçimlendirilmiş veya biçimlendirilmemiş verileri indeksler, gerçek zamanlı arama yapar, çözümlemeli sonuçlarını raporlar. Bu nedenle log dosyaları, Splunk için iyi uygulamalardır [35].

Apache Kafka

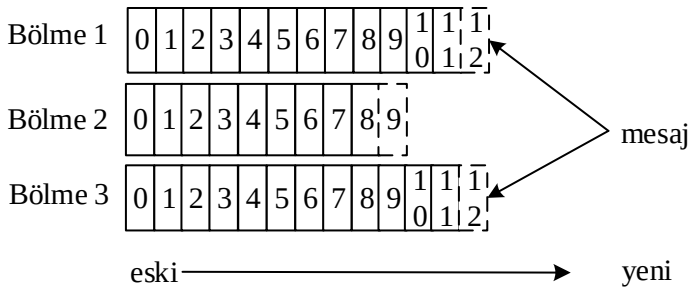
Apache Kafka yayınla-yakala (publish-subscribe/pub-sub) mekanizması üzerine kurulu, log kayıtlarına benzer yapıda kayıt tutan bir mesajlaşma sistemidir [36]. Yayınlanacak mesajın kategori ismine konu (topic) denir. Kafka'ya özgü temel kavramları kısaca anlatmak gerekirse;

- Küme (cluster) şeklinde çalışır; küme içerisinde bulunan her bir sunucuya ‘broker’ adı verilir.
- Mesaj gönderene üretici (producer); alana ise alıcı (consumer) denir.
- Her mesaj bir konu (topic) ile gönderilir; gönderen de dinleyen de aynı 'topic' üzerinden işlem yaptığı sürece iletişim mümkündür.
- Mesajlaşma işlemi TCP protokolü üzerinden gerçekleşir; bu sayede platform bağımsız olarak alıcı (consumer) yaratılabilir.



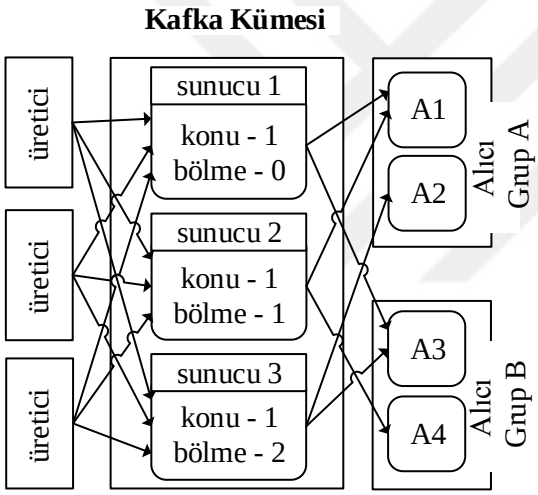
Şekil 2.7. Apache Kafka

Her bir ‘topic’ Şekil 2.8. Kafka log yapısı’ndaki gibi Kafka kümesi içerisinde parçalı loglar halinde tutulur:



Şekil 2.8. Kafka log yapısı

Kafka üzerinde her mesaj bir ‘topic’ ile ilişkilendirilir. Böylece her ‘topic’ bölümlenmiş (parititioned) bir veya birden fazla kayıt içererir. Kafka, mesajları gönderilme zamanına göre sıraya dizer ve bölmelere ayırır. Daha sonra her bölmeye ‘offset’ denilen sıra numarası atar. Bölmeye yazılan mesajın sıra bilgisi (offset) ve bölme (partition) bilgisi birkez atanır ve bir daha değişmez. Log bölmeleri Kafka üzerindeki sunuculara dağıtılır, kopyaları da diğer sunucularda tutulur. Her bölme lider olarak görev yapan bir sunucuya sahiptir, diğer sunucular takipçi gibi davranırlar. İlgili bölmenin tüm okuma ve yazma işlemlerinde lider sorumludur, takipçiler sadece kopya tutar. Lider işlem yapamaz hale geldiğinde takipçilerden birisi görevi devralır ve veri kaybı yaşanmadan süreç devam ettirilir. Bölmeler için farklı sunucular lider olarak çalışır; bunun sonucunda yük dengesi iyi bir şekilde dağıtılmış olur.



Şekil 2.9. Kafka kümesi çalışma prensibi

Kafka kümesi, kayıtları alıcıya gönderildikten sonra silmez, yapılandırma dosyasında belirtilen süre boyunda saklamaya devam eder. Alıcılar ilgili ‘topic’ i dinlemeye başladıklarında geçmişe dönük tüm kayıtları alırlar; bu sayede long polling gibi bir mekanizma kurulur.

SAP Hana

SAP Hana, memory içi mantıksal analiz yapabilen ve iş süreçlerinde, tahminsel analizlerde, hassas verilerin işlenmesinde gerçek zamanlı analizler yapmayı amaçlayan bir platformdur. SAP Hana veri tabanı, gerçek zamanlı platformun çekirdek kısmıdır. Diğer veri tabanı sistemlerinden biraz farklıdır. Operasyonel raporlama, veri setleri oluşturma,

Büyük Veri üzerindeki tahminsel ve yazılı analizler, Hana'ya özel gerçek zamanlı mantıksal analiz adımlarıdır. Hana, SAP tabanlı olan veya olmayan çok fazla uygulama ile çalışabilme özelliğine sahiptir [37].

2.1.3. İnteraktif analizde Büyük Veri araçları

Son yıllarda ortaya çıkan açık kaynak kodlu Büyük Veri sistemleri yığın işleme ve akış (stream) işleme özelliğinin yanı sıra interaktif analiz işlemi de yapmaktadır. İnteraktif analizde, veriler etkileşimli bir ortamda gösterilir ve kullanıcıların, bilgilerinin analizini kendilerinin yapmasına izin verir. Kullanıcı direkt olarak bilgisayara bağlanır ve gerçek zamanlı iletişim kurar. Veriler tablo veya grafikler şeklinde analiz edilebilir, gözden geçirilebilir ya da birbiriyle karşılaştırılabilir.

Google's Dremel

Google 2010 yılında Dermel adı verilen ve iç içe geçmiş (nested) verilerin analizini yapmaya yarayan bir interaktif analiz sistemi geliştirmiştir. Dremel, Apache Hadoop'dan daha farklı bir mimariye sahiptir ve Map/Reduce tabanlı hesaplamalar için çok iyi bir tamamlayıcıdır. Saniyeler içerisinde, trilyon satırdan fazla sorguyu çalıştırabilme kapasitesine sahiptir [38].

Apache Drill

Büyük Veri'nın interaktif analizinde kullanılabilen bir diğer dağıtık sistemdir. Google Dremel'e benzer. Drill birçok dildeki farklı sorguları, veri formatlarını ve veri kaynaklarını destekleyebilir. Drill de Dremel gibi, iç içe geçmiş verileri verimli bir şekilde kullanabilmek için özel olarak tasarlanmıştır. 10.000 ve üzerinde servise erişebilir, petabayt büyüklüğündeki verileri ve trilyonlarca kaydı saniyeler içerisinde işleyebilme kapasitesine sahiptir. Drill ve Dremel, depolama işlemi için HDFS, yığın işleme işini gerçekleştirmek için Map/Reduce fonksiyonlarını kullanır. Sütunlar halinde ya da dağıtık dosya sistemlerindeki verileri arayarak, petabayt büyüklüğündeki verileri taramak mümkündür. Drill, Dremel'in açık kaynak kodlu sürümüdür. Google BigQuery ihtiyacı için, Dremel-as-a-Service eklentisini sunmaktadır. Diğer şirketler, Büyük Veri araçlarını özel kullanım için tasarlayabilmektedir. Her Büyük Veri aracının bir odak noktası vardır.

Bazıları yığın işlemede, bazıları da gerçek zamanlı işlemlerde çok iyidir. Her Büyük Veri platformunun özel fonksiyonu vardır. İstatiksel analiz, makine öğrenmesi, veri akımı işleme bunlara örnek verilebilir [39].

2.2. NoSQL

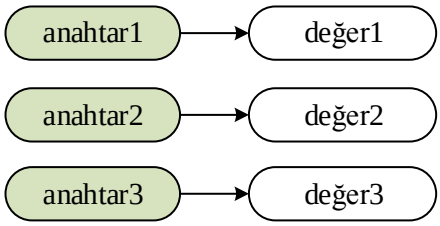
Dağıtık veri yönetimi ve veritabanı tasarımı için kullanılan yaygın yöntem NoSQL (Not only SQL) yapısına sahip bir veri tabanı kullanımıdır. Birçok NoSQL veritabanı şema bağımsız (schema-free) çalışabilme özelliğine sahiptir. Şema-bağımsız veri tabanlarının en büyük avantajı, veri yapılarının kolaylıkla değiştirilebilmesi ve bunların tabloya yeniden yazılmasının gerekmemesidir. Buna ek olarak, yapılandırılmış veriler heterojen bir şekilde depolandığı zaman dahi çok büyük bir esnekliğe sahiptir. Veri yönetim katmanında, veri doğru ve bütünleşik olmaya zorlanır. Apache Cassandra, Facebook patentli bir veritabanı olup, açık kaynak kodludur ve 2008 yılında yayınlanmıştır. Halen NoSQL veritabanları içerisinde yaygın olarak kullanılmaktadır. Twitter, LinkedIn ve NetFlix gibi şirketler de NoSQL veritabanı kullanmaktadır [40]. NoSQL veri tabanları 4 ayrı kategori altında toplanabilir.

2.2.1. Anahtar-değer şeklinde depolama yapan veri tabanları

Veri blok olarak anahtar ile ilişkilendirilerek kaydedilir. Bu tip veritabanlarının veri içeriği ile ilgili bir bilgisi yoktur; işlemleri anahtar üzerinden yaparlar. Aşağıda yaygın olarak kullanılan Anahtar-değer veri tabanları listelenmiştir. Şekil 2.10'da örnek bir anahtar-değer kaydı yapısı görülmektedir.

Anahtar-değer Veri Tabanları

- DynamoDB
- Azure Table Storage
- Riak
- Redis
- Aerospike
- FoundationDB
- LevelDB
- BerkeleyDB
- GenieDB
- MemcacheDB



Şekil 2.10. Anahtar-değer kaydı örneği

Anahtar-değer veri tabanlarının avantaj ve dezavantajları

Anahtar-Değer veritabanı genel olarak basit uygulama programlama arayüzlerine (API) sahiptir. Tüm işlemler tek bir *Anahtar* üzerinden yönetildiğinden dolayı tutarlılık ve bütünlük sağlaması kolaydır; uygulama geliştirici *Anahtar* verisini kullanarak kolayca ekleme, çıkarma ve güncelleme işlemlerini üç metot (put, remove, get) ile gerçekleştirebilir. Güncelleme durumunda iki farklı yaklaşım kullanılabilir; birincisi güncellemeleri kabul etme ve çözmek için kullanıcıya sorma, ikincisi ise son kaydın her zaman kazanacağı yaklaşımıdır. *Değer* kolonunda tutulacak verinin şema bilgisi kullanıcı tarafından belirlenir; böylece kullanıcı veriyi aldığında anlam çıkarma sorumluluğunu üstlenmiş olur. Yani veritabanı *Değer*'in formatı ve içeriği hakkında bilgi sahibi değildir. Bunun sonucu olarak verinin içeriği ile ilgili sorgulama ve indeksleme yapılamaz; kopyalanması ve çoğaltılması ise kolaydır (replication) [41].

2.2.2. Doküman tabanlı depolama yapan veri tabanları

Bu tip veri tabanlarında veriler genelde JSON veya XML formatında doküman olarak kaydedilirler. Doküman tipinde kaydedilen veriler, diğer sistemleri desteklemek amacıyla benzersiz belirteç/kimlik (Unique ID) ile kullanılırlar. Şekil 2.11'de döküman tabanlı kayıt örneği görülmektedir. Başlıca döküman tabanlı veri tabanları aşağıda listelenmiştir.

Doküman Tabanlı Veri Tabanları

- MongoDB
- CouchDB
- RethinkDB
- RavenDB
- Clusterpoint Server
- ThruDB
- Terrastore
- RaptorDB
- JasDB
- SisoDB
- SD

```
{
    “adı” : “adı1”
    “soyadı” : “soyadı1”
    “tecrübeler” : [
            {
                “ilk” : “tc1”
                “son” : “tc2”
            }
        ]
}
```

Şekil 2.11. Doküman tabanlı kayıt

Döküman tabanlı veri tabanlarının avantaj ve dezavantajları

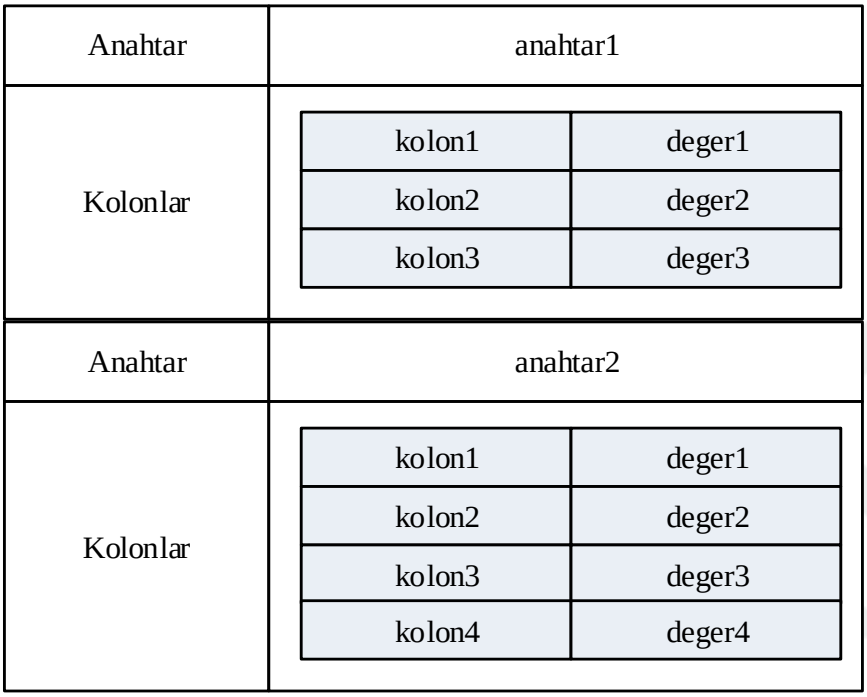
Uygulama geliştiricisi kaydedeceği veri formatını belirleyebilir. Formatların değiştirilebilir olması, programlama dillerinin kullandığı formatlar kullanıldığında, kaydedilen verinin uygulama içerisinde direk eşlenebilir ve kullanılabilir olması, yeni format tipleri ile koordinasyon problemi olmaması gibi özellikler büyük esneklik sağlar. Bunlara ek olarak veri kopyalama ve çoğaltma (replication) esnasında doküman formatları değiştirilebilir; veritabanı veri içeriği hakkında bilgi sahibidir ve onun üzerinden işlem yapabilir. Dolayısı ile dokümanın herhangi bir alanına göre indeksleme ve sorgulama yapılabilmektedir. İndeksleme sisteme ek yük getirir; çoklu indekslenmeye sahip sistemlerde indekslerin yazılması ve güncellemesi maliyetli olabilmektedir. Doküman boyutunun büyük olması durumundam ise veri, binary formata dönüştürülerek kaydedilebilir [41].

2.2.3. Kolon tabanlı depolama yapan veri tabanları

Genel olarak farklı makinelere dağıtılmış veriler için kullanılır; nesne yapısı, ilişkisel veritabanı mimarisi ile benzerlik taşımaktadır. Anahtar-Değer veritabanı ile ilişkisel veritabanı modellerinin mantıksal olarak birleştirilmesiyle oluşmuştur. 2 kolondan oluşmakta; ilk kolonu benzersiz anahtar, ikinci kolon ise kendi içerisinde kolon yapısına sahip olacak şekilde veri tutmaktadır. Değer anahtar ile ilişkilendirilmekte, anahtar verinin kümedeki(cluster) yerini belirlemektedir. Şekil 2.12’ de kolon tabanlı kayıt örneği görülmektedir.

Kolon Tabanlı Veri Tabanları

- HBase
- Cassandra
- Hybertable
- Accumulo
- Amazon SimpleDB
- Cloudata
- HPCC
- Stratosphere
- Druid



Şekil 2.12. Kolon tabanlı kayıt

Kolon tabanlı veri tabanı avantaj ve dezavantajları

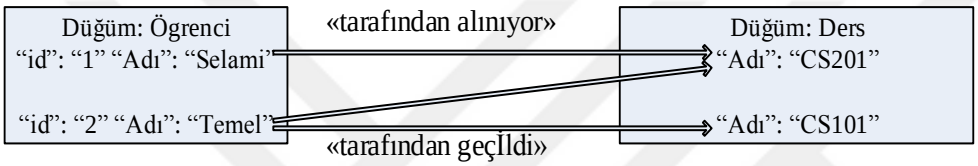
Veri tek bölmede tutularak Anahtar ile ilişkilendirilmektedir ve veritabanı verinin içeriği hakkıda bilgi sahibi değildir. Bu sebeple indeksleme ve sorgulama işlemleri için uygun değildir. Kolon yapısının desteklenebilmesi için veri modeli, denormalize edilip minimum seviyede ilişkiler kurulur, uygulama geliştirici tarafından değiştirilebilir ve uygulama içerisinde direk kullanılabilecek hale getirilebilir. İşlemler atomik olarak yapıldığından dolayı veri kopyalama, güncelleme, silme ve ekleme işlemleri hızlı gerçekleşmektedir. Dolayısı ile tutarlılık ve bütünlük problemi yoktur; kopyalama ve çoğaltma (replication) kolayca yapılabilir [41].

2.2.4. Grafik tabanlı depolama yapan veri tabanları

Veriler düğüm olacak şekilde ilişkileri ile birlikte kaydedilir; birbirleri ile ilişkili verileri optimize eder. İlişkilerin özellikleri olabilir, yönleri ve isimleri vardır. Şekil 2.13'de grafik tabanlı örnek bir kayıt görülmektedir.

Grafik Tabanlı Veri Tabanları

- Neo4J
- Infinite Graph
- InfoGrid
- HyperGraphDB
- GraphBase
- VertexDB
- FlockDB
- Meronymy



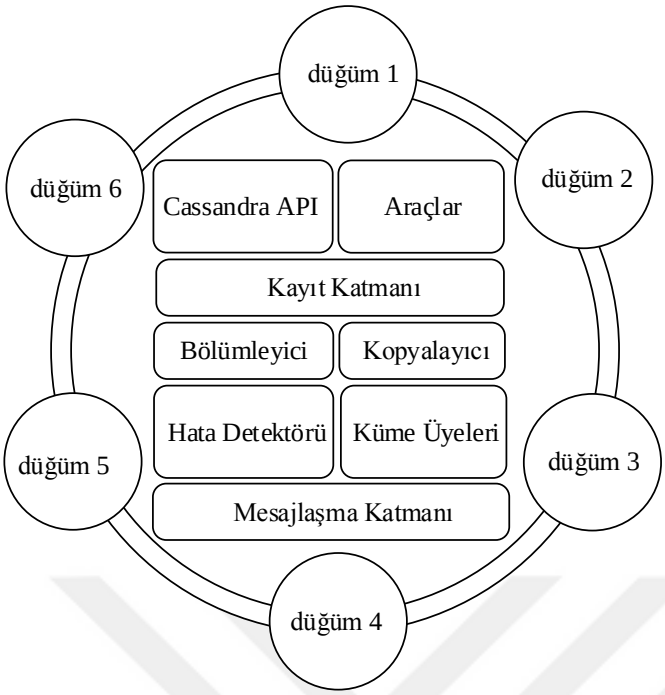
Şekil 2.13. Grafik tabanlı kayıt

Grafik tabanlı veri tabanı avantaj ve dezavantajları

Grafik tabanlı veri tabanları ilk kayıt oluşturma ve güncelleme sırasında diğer veritabanı tiplerine göre yavaş kalmaktadır. Uygulamanın sahip olduğu veri yapısı güncellenebilir ve değiştirilebilir yapısı ile geliştiricilere büyük kolaylık sağlar. Yapılan güncelleme ve oluşturma işlemleri sonucunda linkler kaydedildiğinden sorgulama işlemleri çok hızlı gerçekleşir [41].

2.3. Apache Cassandra

Apache Cassandra [42] kolon tabanlı NoSQL veri tabanı mimarisine sahip, ölçeklenebilir, açık kaynak kodlu ve dağıtık bir veri tabanı sistemidir. Cassandra dağıtık mimarisi, efendi-köle kavramına sahip değildir; sunucular birbiri ile gossip protokolü ile haberleşmektedir.



Şekil 2.14. Cassandra küme mimarisi

Cassandra mimarisinde veri tabanı sunucuları olarak kullanılan düğümler birleşerek kümeyi oluşturur; dolayısı ile veriler birden fazla sunucuda saklanır. Bu sayede erişilemeyen sunucularda bulunan verilere erişilebilir. Sorgulama işlemleri Cassandra Query Language (CQL) kullanılarak yapılır.

2.3.1. Cassandra küme mimarisi bileşenleri

Gossip: Küme içerisinde bulunan sunucuların tesbiti, yer ve durum bilgilerinin paylaşılmasını sağlayan protokoldür [42].

Partitioner (Bölümleyici): Verinin sunucular arası gönderimini ve hangi sunucunun ilk kopyaya sahip olacağını belirler [42].

Replication factor (Kopyalama faktörü): Bir verinin kaç kopyası olacağını ve hangi sunucuların bu kopyalara sahip olacağının belirlendiği kısımdır [42].

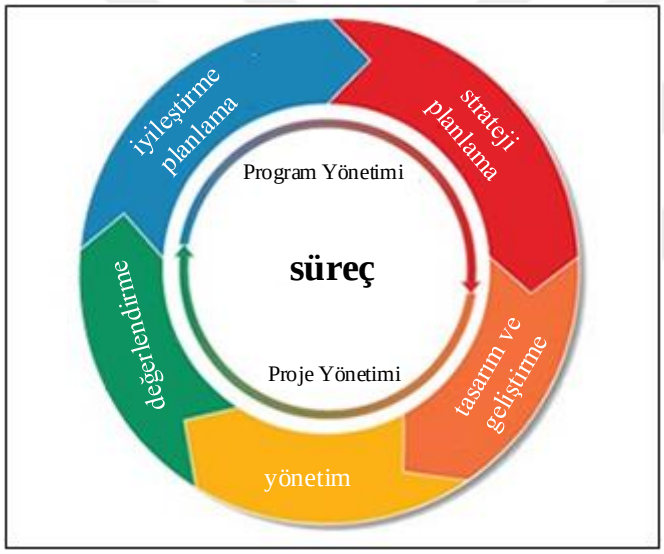
Replica placement strategy (Kopya yerleştirme stratejisi): Cassanda bir verinin kopyasını küme içerisinde bulunan sunuculara da yazar; ancak sunucunun kapasitesi, bulunduğu

lokasyon gibi sebeplerden ötürü hangi sunucuların kopyaları saklamak için kullanılacağı ağ topolojisi stratejisi belirlenmelidir [42].

Snitch: Kopyalama stratejisi topolojisini ve sunucuları belirleyen mekanizmadır [42].

3. SİSTEM MİMARİSİ VE BİLEŞENLERİ

Sistem tasarımındaki ana hedef, güvenlik sistemlerinin kurumsal düzeyde iyileşmesine, süreç ve teknolojilerin tümleşik çalışmasına, tasarım ve gelişimine yardımcı olmaktır. Yaygın erişilebilirlik, kullanım kolaylığı, kullanıcı dostu arayüzü bu teknoloji platformunun önemli özellikleridir. Sistem, bilginin kullanımı ve yaşam döngüsü boyunca daha verimli ve etkin biçimde yürütülmesini sağlamalıdır. Aynı zamanda teknoloji ilerlemeye devam ettikçe, tasarlanan sistemin ve yeteneklerinin geliştirilmesi gerektiği kabul edilmektedir. Bu nedenle Şekil 3.1. Sistem yaşam döngüsü'de belirtilen süreç kullanılarak esnek ve geliştirilebilir mimari elde etmek hedeflerden bir tanesidir.



Şekil 3.1. Sistem yaşam döngüsü

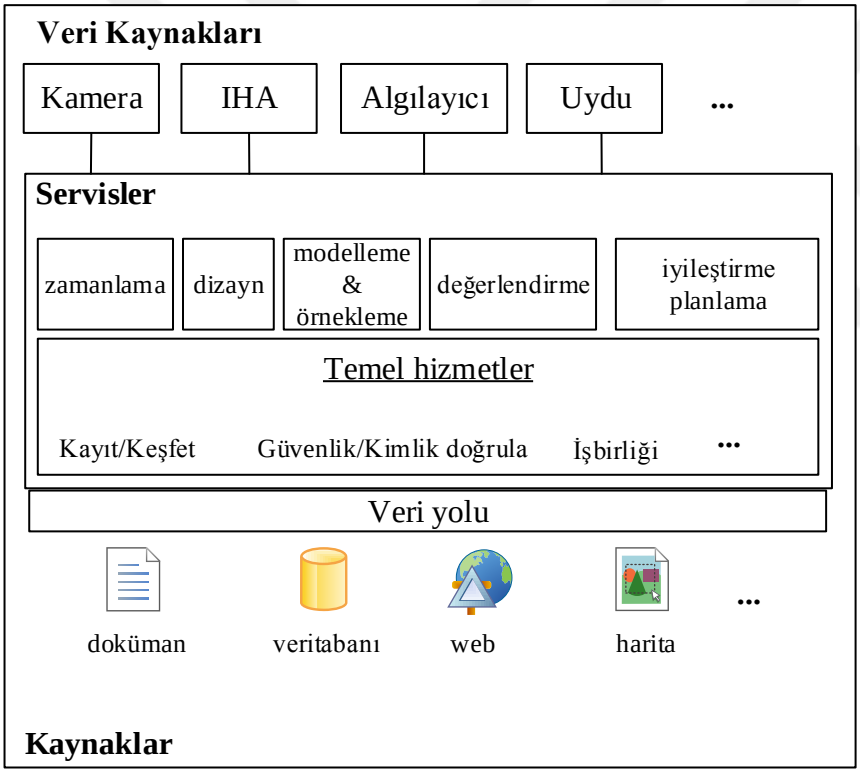
Tasarımın odak noktası birçok olay, aktör ve durum üzerinedir. Hedef bu bilgilerle, çevre ve kullanıcı etkileşimini daha akıllı ve daha verimli hale getirmektir. Özet olarak, anahtar yenilik hedeflerini sıralamak gerekirse:

- Kullanıcı etkileşimi ile proaktif karar destek.
- Basit, kullanışlı insan makine arayüzü.
- Çoklu model etkileşimi.
- Misyon etkinliği artarken ilgisiz bilgi yükünde azalma.
- Genel ve ortak amaçlı mimarinin farklı alanlarda kullanımı.

Boru hattı geliştirme için birçok teknoloji, mimari ve altyapı mevcuttur. Bir boru hattı mekanizmasının;

- Farklı veri kaynaklarıyla çalışabilmesi,
- Veri iletiminin düşük gecikmeli olması,
- Veri iletimi ve kaydı sırasında veri kaybının önlenmesi,
- Veri işleme hızının yüksek olması,
- Sonuçları okunabilir ve anlaşılabilir şekilde sunması gerekir.

Mimari yapılar değerlendirildiğinde Şekil 3.2'de görülen genel mimari yapısı ortaya çıkmaktadır.



Şekil 3.2. Genel mimari yapı

Anlatılan hedeflerin gerçekleştirilebilmesi için teknik ve teknoloji seçimleri oldukça önemlidir. Sınır güvenliği gibi kritik önem taşıyan sistemler hem gerçek zamanlı hem de yığın veri üzerinde analiz sonuçlarına ihtiyaç duyarlar. Bu sebeple gerçek zamanlı ve yığın veri analizi için en uygun teknoloji olan Spark seçilmiştir. Spark seçimi ile iki farklı amaç için tek platformda uygulama geliştirme olanağı elde edilmiştir.

Veri kaynaktan hedefe gönderilirken hızlı, kayıpsız, sıralı ve hata toleranslı şekilde iletilmelidir. Microsoft ve Amazon'un ürünleri de Kafka kadar güçlü olmasına rağmen ücretli olduklarından dolayı tercih sebebi olmamışlardır. Sonuç olarak Kafka veri iletim hattı olarak seçilmiştir.

Sensor verileri bu çalışma için kritik öneme sahiptir ve atomik olarak işlem görecektir. Kopyalama, yazma ve silme işlemlerinin hızlı olması ve kolonlar üzerinden işlem yapabilmesinden dolayı, veri saklama katmanı için, Cassandra NoSQL veri tabanı seçilmiştir. Bunlara ek olarak Cassandra, Spark ile kolaylıkla birleştirilebilir. Bu konu ile alakalı detaylı açıklamalara Bölüm 2.3'de yer verilmiştir.

3.1. Mimari Tasarım

Veri, alınmasından sunulmasına kadarki her aşamada işleme tabi tutulmalı, yani büyük iş paketleri daha küçük parçalara bölünerek aşama aşama kurgulanmalıdır. Bu mantık çerçevesinde sistemin ana iskeleti Olay-Tabanlı Mimari (OTM/ Event-Driven Architecture - EDA) [43] ile tasarlanmıştır. Bunun sonucunda, her bileşen ayrı ayrı ele alınarak gevşek bağlanmış ve tek sorumluluk prensibi ile kodlanmış bir yapı elde edilmiştir.

Bu çalışma ile gerçek zamanlı akan verileri ve yığın verilerini işleyebilmek için dağıtık, ölçeklendirilebilir, duyarlı, tutarlı, dayanıklı, düşük gecikmeli bir boru hattı geliştirme amaçlanmıştır. Geliştirilen sisteme ait adımlar sırasıyla aşağıda olduğu gibidir.

Adım 1: Apache Kafka ile farklı kaynaklardan alınan veriler, bir konu (topic) ile eşleştirilerek gönderilir.

Adım 2: Apache Spark ile Apache Kafka birleştirilerek Spark üzerinde mikro yığın serileri (DStreams – RDD serisi) oluşturulur, daha sonra ise RDD üzerinde Map/Reduce işlemleri yapılır.

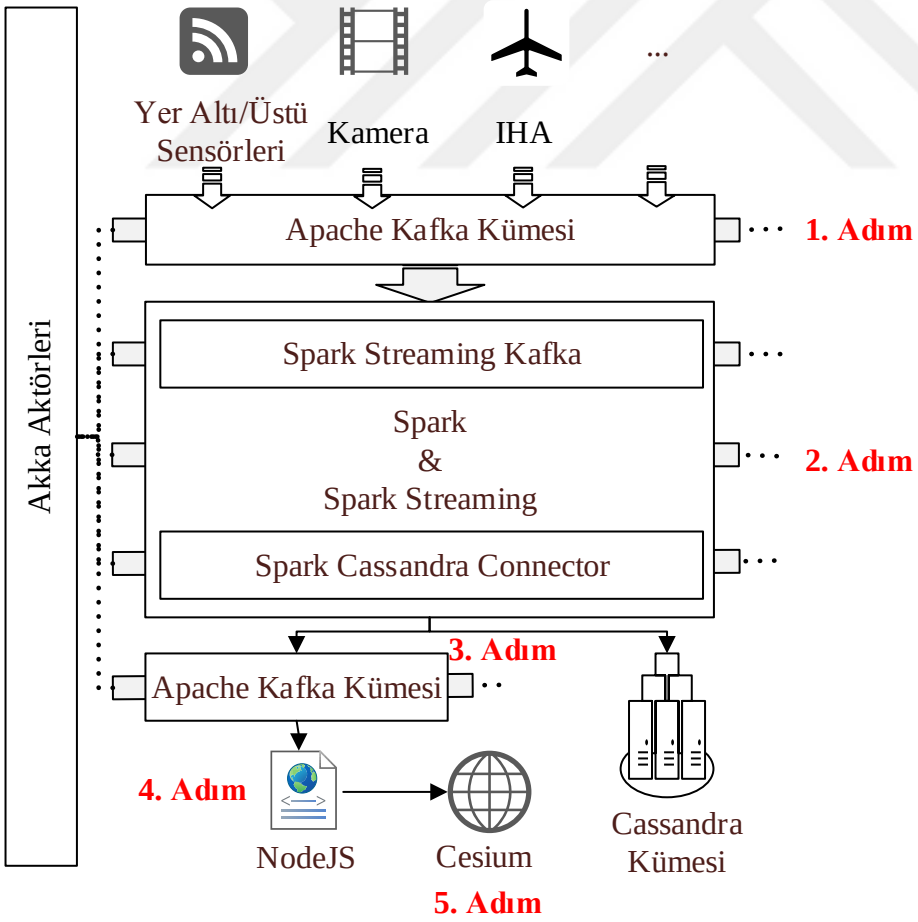
Adım 3: Spark'ta işlenen veriler Cassandra'ya kaydedilir ve eş zamanlı olarak Kafka ile yayınlanır.

Adım 4: Web sunucusu ve Kafka İstemcisi (Kafka Consumer) olarak görev yapan sunucu

(NodeJS ile geliştirilmiş), yayınlanan verileri alır ve Socket.IO ile web sayfasına gönderir. NodeJS, sunucu tarafında (server side) JavaScript programlama dili kullanılarak uygulama geliştirilmesini sağlayan, açık kaynak kodlu, olay tabanlı, bloklanmayan Girdi/Çıktı (I/O) modeline sahip mekanizmadır. Asenkron işlemler için elverişli, zaman ve kaynak kullanımı oldukça düşük, performansı ise yüksektir. WebSocket ve Socket.IO'yu destekler; bu sebeple yüksek trafikli işlemler gerektiren gerçek zamanlı uygulamalar için oldukça kullanışlıdır.

Adım 5: Web sayfası tarafından alınan veri 3 boyutlu Cesium nesnesine dönüştürülerek gösterilir. Cesium, JavaScript ile geliştirilmiş, açık kaynak kodlu ve 3 boyutlu çalışmaya olanak veren Coğrafi Bilgi Sistemi kütüphanesidir.

OTM'de veri gönderen ve veri alan bileşenler arasında doğrudan bir ilişki yoktur. Şekil 3.3'de görüldüğü üzere iş akışı, Kafka üzerinden iletilen olaylar ile gerçekleşir.



Şekil 3.3. Tasarlanan sistem mimarisi

3.2. Gerçek Zamanlı Veri Akışı

Gerçek zamanlı işlemlerde akan verinin büyüklüğünü ve frekansını tahmin etmek zordur. Dolayısı ile verinin iletilmesi esnasında veri gönderen ile alan arasında darboğaz/tıkanıklık (bottleneck) yaşanabilir. Buna istinaden Jonas Bonér, Dave Farley, Roland Kuhn ve Martin Thompson 2014 yılında yayınladıkları Reaktif Manifestosu'nda (The Reactive Manifesto) [44] bloklanmayan (non-blocking) geri basınç (back pressure) ile asenkron akış işleme (asynchronous stream processing) yaklaşımını ortaya atmışlardır. Manifestonun yayınlanmasından sonra Reaktif Akışları destekleyen birçok sistem ve araç geliştirilmiştir. Bunlardan bir tanesi de Akka ile geliştirilen Akka Akışları'dır (Akka Streams).

Kaynaklardan veri alma için kodladığımız TCP İstemci/Sunucu modelinde Akka Akışları'ndan faydalanılmıştır.

```
trait Server extends Actor
object TcpServer {
  def props(handlerProps: HandlerProps): Props =
    Props(classOf[TcpServer], handlerProps)
}
class TcpServer(handlerProps: HandlerProps) extends Server {
  IO(Tcp)  !  Tcp.Bind(self,  new  InetSocketAddress(ApplicationSettings.appHostName,
ApplicationSettings.appPort))
  override def receive = {
    case Tcp.CommandFailed(_: Tcp.Bind) => context stop self
    case Tcp.Connected(remote, local) =>
      val handler = context.actorOf(handlerProps.props(sender))
      sender ! Tcp.Register(handler)
  }
}
```

Şekil 3.4. Akka tabanlı TCP istemci/sunucu kod örneği

TCP sunucusu, verileri hızlı bir şekilde alır ve sonrasında Kafka ile yayınlar. Sistem bütünlüğü ve tutarlılığı için veri alma hızı ile gönderme hızı arasındaki fark minimum olmalıdır. Bu yeteneğin kazanılması için Kafka Üreticisi (Kafka Producer) ve Kafka İstemcisi (Kafka Consumer) sınıfı aktör tabanlı işlem yapan Akka kütüphanesi ile sarmalanarak KafkaAkka Üreticisi (KafkaAkkaProducer) ve KafkaAkka İstemcisi (KafkaAkkaConsumer) sınıfları kodlanmıştır.

```
object KafkaHandlerProps extends HandlerProps{
  def props(connection: ActorRef) = Props(classOf[KafkaHandler], connection) }
class KafkaHandler(connection: ActorRef) extends Handler(connection) with DataPublisher{
  def received(data: String): Unit = {
    streamingPublisher(data)
    connection ! Write(ByteString(data + "\n"))   }
}
trait HandlerProps {
  def props(connection: ActorRef): Props }
abstract class Handler(val connection: ActorRef) extends Actor {
  def receive: Receive = {
    case Received(data) =>
      data.utf8String.trim match {
        case abort() => connection ! Abort
        case confirmedClose() => connection ! ConfirmedClose
        case close() => connection ! Close
        case str => received(str)
      }
    case PeerClosed =>
      ...
    case ErrorClosed =>
      ...
    case Closed =>
      ...
    case ConfirmedClosed =>
      ...
    case Aborted =>
      ...
  }
}
```

Şekil 3.5. Akka tabanlı Kafka üreticisi/tüketicisi kod örneği

Uygulama içerisindeki tüm mesajlaşma işlemleri, bu sınıflardan türetilen nesneler vasıtasıyla yapılır.

3.3. Modelleme ve Temizleme

Sınır güvenliği söz konusu olduğunda sadece gerçek zamanlı işlem yapmak yeterli olmayacaktır. Yığın veri üzerinde çalışmak; alakasız olan verileri çıkarma, gruplandırma ve gruplar üzerinden anlam çıkarma gibi oldukça faydalı bilgiler elde edilmesine yardımcı olmaktadır. Sisteme bu yeteneği kazandırmak için, veri modelleme ve temizleme sürecinde, Nathan Marz tarafından ortaya konan Lambda Mimarisi temel alınmıştır [45]. Bölüm 3.3.1'de detaylı açıklamaya yer verilmiştir. Araç olarak ise, büyük ölçekli veriler üzerinde dağıtık hesaplama yapmayı destekleyen Apache Spark tercih edilmiştir. Gerçek zamanlı ve yığın veri işleme yeteneklerine sahip olan bu araç, istekleri kuyrukta tutar ve aksiyon fonksiyonu (action function) çağrılana kadar bekletmeye devam eder (lazy evaluation), işlem sonuçlarını belleğe yazarak sonraki işlemlerin hızlı gerçekleşmesini

sağlar. Spark Streaming ise farklı kaynaklardan alınan gerçek zamanlı veriler üzerinde işlem yapmayı sağlayan bileşendir. Spark Streaming, akan her veri için DStream (Discretized Stream) oluşturur. DStream, üzerinde işlem yapılan mikro yığın (RDD) serisi olarak ifade edilebilir.

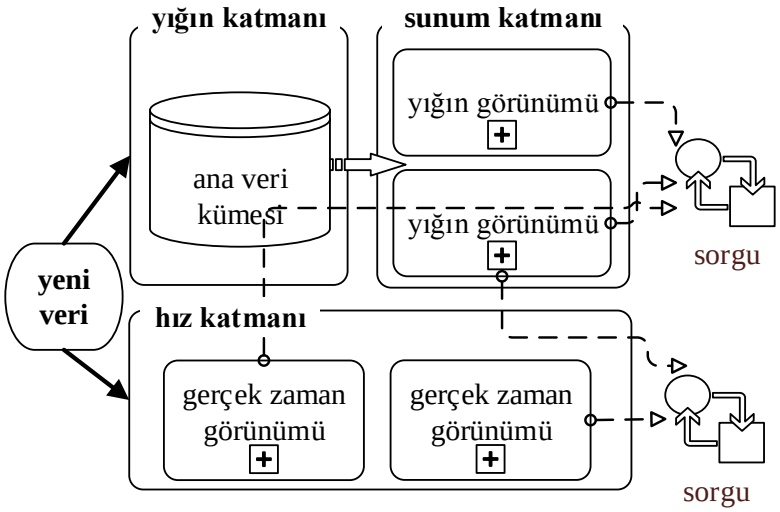
```
val sparkConf = {
   val conf = new SparkConf()
   ...
   conf  }
 ssc = new StreamingContext(sparkConf, ApplicationSettings.batchInterval)
 val topicsSet = ApplicationSettings.tcpStreamingTopic.split(",").toSet
 val kafkaParams = Map[String, String](
   ...       )
 val streams = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder](
   ssc, kafkaParams, topicsSet)
```

Şekil 3.6. Spark – Kafka entegrasyonu kod örneği

Spark Streaming birçok mesajlaşma sistemi ile kolaylıkla entegre edilebilir ve aynı anda birden fazla konu/başlık (topic) üzerinden veri alabilir. Bunlara ek olarak, gerçek zamanlı akan veriyi belirli bir süre biriktirdikten sonra işlenmesine olanak sağlayan yığın aralığı (batch interval) konfigürasyonu da mevcuttur. Mesajlaşma sistemi olan Kafka ile Spark arasında veri akışı sağlanabilmesi için Alıcı Tabanlı Yaklaşım (Receiver Based Approach) ve Direkt Yaklaşım (Direct Approach – No Receivers) olmak üzere iki yöntem mevcuttur [46]. Bu çalışmada sıfır veri kaybı, kolay kurulum ve özel ayar gerektirmeden paralel çalışma özelliklerinden dolayı Direkt Yaklaşım seçilmiştir.

3.3.1. Lambda mimarisi

Lamda Mimarisi (LM) akış işleme (stream processing) ve yığın işleme (batch processing) metodlarının avantajlarını kullanarak büyük miktardaki verinin işlenmesi için tasarlanmış veri işleme (data processing) mimarisidir [47]. Yığın katmanı (batch layer), hız katmanı (speed layer) ve sunum katmanı (serving layer) olmak üzere 3 ana bileşenden oluşur.



Şekil 3.7. Lamba Mimarisi

Şekil 3.7' de görülen Lambda Mimarisi'nde sisteme giren her veri hem yığın katmanına hem de hız katmanına gönderilir. Yığın katmanının iki temel görevi vardır. Bu görevlerden biri ana veri kümesini kontrol etmek, diğeri ise yığın parçacıkları oluşturmaktır. Yığın parçacıkları, sunum katmanında dizinlenerek sorgulamalar için hazır hale getirilir. Hız katmanı, sunum katmanındaki güncelleme işlemlerinden kaynaklanan büyük gecikmeleri dengeler ve son gelen veriler üzerinde işlem yapar. Hem yığın katmanı hem de hız katmanı ya da her ikisi birden kullanılarak sorgulamalar yapılabilir.

3.4. Sistemde Veri Saklama ve Sunma

Spark Cassandra Connector kütüphanesi ile Cassandra tabloları Spark RDD'leri, Spark RDD'leri ise Cassandra tablolarıymış gibi yönetilebilir. Dolayısı ile Map/Reduce işlemleri sonucu oluşan RDD'ler, Cassandra veri tabanına yazılabilir. Kodlanan Spark uygulaması da bu yetenekleri kullanarak RDD'leri kaydeder.

Tezde, işlenmiş verilerin son kullanıcılara bilgi amaçlı gösterimi de hedeflenmiştir. Bu nedenle işlenmiş veriler kaydedilirken, aynı anda da Kafka üzerinden yayınlanmaktadır. Kütüphaneler yığını olan NodeJS, bu çalışmada web sunucusu işlevi görmesinin yanı sıra Kafka istemcisi olarak da görev yapar. Kafka içerisinde kayıtlar alıcıya gönderildikten sonra silinmez konfigürasyon dosyasında belirtilen süre boyunca saklanmaya devam eder. Bu sayede 'long polling' gibi her kullanıcının sahip olabileceği bir veri seti mekanizması kurulur. İstemciler ilgili 'topic'i dinlemeye başladıklarında geçmişe dönük tüm kayıtları

alırlar. Yani NodeJS istemcisi, Kafka ile bağlantı kurduğunda geçmişe dönük işlenmiş verileri alır. Web sunucu ile istemciler arasında haberleşme modeli olarak Socket.IO seçilmiştir. Web istemcisi, Socket.IO üzerinden aldığı verileri, 3 boyutlu olarak CesiumJS nesnesine dönüştürür ve zaman serisi şeklinde tutarak verilerin geçmişe dönük incelenmesine olanak sağlar.

3.5. Gerçekleme

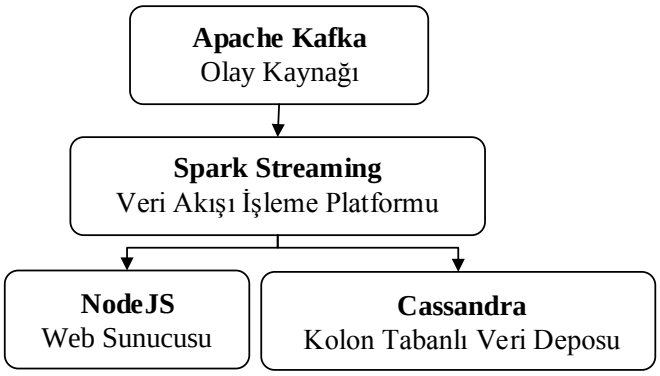
Büyük sistemlerin ihtiyaçlarını karşılamak için tasarlanan mimari;

- hata toleranslı,
- donanım ve insan kaynaklı hatalara karşı kararlı,
- iş yüklerini ve farklı kullanım amaçlarını geniş yelpazede destekleyici,
- düşük gecikme süreli okumalı ve güncelleyebilmeli,
- ölçeklenebilir ve genişletilebilir,

yapıda olmalıdır. Bu sebeple tezde sınır güvenliğini destekleyecek alternatif bir altyapı oluşturmak öncelikli olarak ele alınmıştır. Tezde analiz sonuçlarının doğruluğu ve performans değerlendirmesi ikincil önceliğe sahiptir. Sistemin iş akışı Şekil 3.8'de görülmektedir. Bunlar sırasıyla;

- Kafka'dan olay oku
- Verileri anlaşılabilir hale getir
- Gereksiz olayları filtrele
- İlgili alanlarla projeksiyon oluştur
- Kaydet ve sun

şeklindedir.



Şekil 3.8. Sistem iş akışı

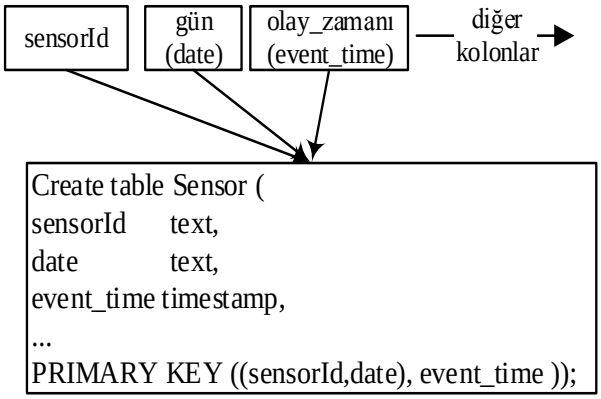
Sistemin sahip olduğu yazılım konfigürasyonları Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Yazılım konfigürasyonları

Bileşenler	Versiyonlar
Scala	2.10.4
Apache Spark	1.4.1
Cassandra	2.2.1
Apache Kafka	2.10-0.8.2.1
Spark Cassandra Connector	1.4.1
JavaCV & JavaCpp	1.1
Gson	2.4
OpenCV	3.0
FFMPEG	2.8.1
NodeJS	0.10.25

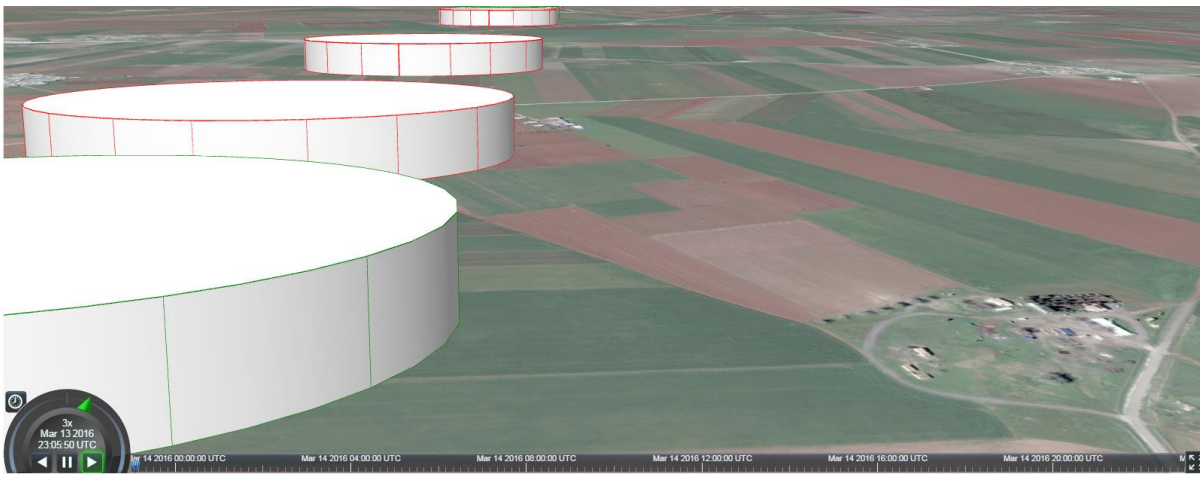
Bir bölgenin güvenliğinin sağlanması değerlendirildiğinde sensor, kamera veya İHA üzerinden alınan verilerin analizi büyük önem taşır. Dolayısı ile kodlanan boru hattının işlevselliğinin test edilebilmesi için, text ve görüntü analizi amaçlı, en az iki farklı prensibe sahip veri işlenmelidir. Bu sebeple birincil kaynak olarak JSON formatında sensor verisi üreten TCP istemcisi; ikincil kaynak olarak ise, görüntü analizi maksadıyla, video çerçevelerinin (frame) gönderilmesi olarak seçilmiştir.

JSON formatında gönderilen sensor verilerini işlemek için veri; gönderilme zamanı, sensor kimliği, yerleşimi, statüsü (telemetri bilgilerinden sadece kalan pil süresi), hareket tespit edip-etmediği, eğer tespit etti ise tespit zamanını içerecek şekilde yeniden modellenir. Modellenen veriler sensor kimliğine göre gruplandırılır ve sonrasında gönderilme zamanına göre sıralanır (Map). Zamana göre ardışık hale getirilen verilerden değişiklik (statü ve hareket tespit bilgisi) gözlemlenmeyenler temizlenerek rafine bir veri seti elde edilir (Reduce). Şekil 3.9'da görüldüğü üzere kimlik (id) ve gün bilgisine, zaman bilgisi eklenerek birincil anahtar (primary key) elde edilir. Bu sayede her sensor için günlük bir satır kayıt oluşturularak sorgulama işlemlerinin hızlı olması sağlanır. Görüntü verisi kayıtları için ise gün ve zaman bilgileri birleştirilerek birincil anahtar oluşturulur ve sonrasında kaydedilir.



Şekil 3.9. Sensor tablosu

JavaCV, JavaCPP ile sarmalanan bilgisayar görüşü alanında geliştirilmiş kütüphanelerin Java ve Android platformlarında kullanmasını sağlar [48]. Bu çalışmada, byte dizisi şeklinde alınan görüntü verisi analizinde, JavaCV kullanılmıştır. Her bir kare içerisinde bulunan yüzleri bulmak amaçlandığından, kareler arasında doğrulama veya gruplama gibi bir ihtiyaç yoktur. H.264 ile kodlanmış, 23 fps kare hızlı, 1280x720 çözünürlüklü videodan gönderilen kareler, Haar Özellikleri Tabanlı [49] yüz algılama sınıflandırıcısı sürecinden, tek tek geçirilir. İnsan yüzü bulunan kareler saklanır, aksi durumda olanlar ise silinir.



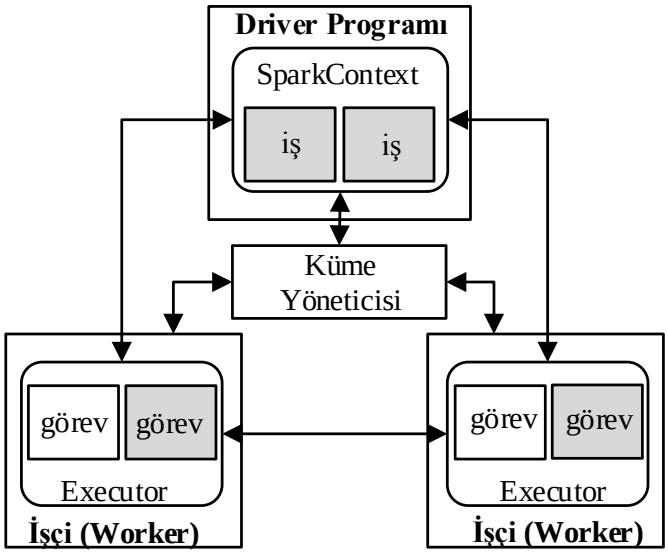
Resim 3.1. Harita ekran görüntüsü

Cesium üzerinde gösterilen sensor verileri, kapsama alanları ile birlikte üç boyutlu hale getirilir ve gönderilme zamanına göre zaman serisi şeklinde tutulur. Güncelleme yapılanlarda ise değişiklikler, ilgili zamanla eşleştirilerek nesneye eklenir. Hareket tespit eden sensorlar kırmızı, etmeyenler ise yeşil renkteki çizgilerle ifade edilir.

4. BULGULAR VE TARTIŞMA

Cesium sensorler için özelleşmiş bir modüle sahiptir. Farklı şekillerle ifade edilebilen üç boyutlu sensor verileri zaman serisi şeklinde de bellekte saklanabilir. Bu yeteneğiyle büyük kolaylık sağlamasına rağmen, WebGL temelli olduğundan, WebGL destekli tarayıcılar üzerinde çalıştırma zorunluluğu vardır.

Spark güçlü, hızlı ve sağlam bir platformdur. Ek olarak fonksiyonel programlama dilleriyle uygun çalışabilmesi de tercih edilme sebeplerinden birisidir. Fakat karışık uygulamalarda problemlerle karşılaşılması kaçınılmazdır. Geliştirme sürecinde, Spark'ın nesne serialization ve yaşam döngüsü yönetiminde bir problemle karşılaşılmıştır. Uygulama yığın izinde (application stack trace) *java.io.NotSerializableException* hatası ile karşılaşılmıştır. Bu hata, kodun bir kısmı Spark 'driver' üzerinde, diğer bir kısmının da 'executor' üzerinde çalışmasından kaynaklanır. 'Driver' programı olarak adlandırılan SparkContext nesnesi tarafından yönetilen Spark uygulamaları, küme içerisinde bağımsız işlem setleri şeklinde çalışırmaktadır.



Şekil 4.1. Spark küme çalışma mantığı

Yukarıdaki şekilden anlaşılacağı gibi 'driver' üzerinde oluşturulan işler, işçilere görev olarak dağıtılır. İş sayısı kurgulanan mantığa, görev sayısı ise veri bölümleme ile ilgilidir. Yani RDD işlemleri 'driver' üzerinde, bölümlenmiş RDD'ler 'executor' üzerinde çalışır. Bunların sonucu olarak Map/Reduce işlemlerinden geçen verilerin Kafka ile gönderimi

için, her bölmede (partition) Kafka nesnesi oluşturmak zorunda kalınmıştır. Matematiksel olarak ifade etmek gerekirse; saniyede 1000 olaya, 2 sn yığın aralığına ve 16 bölmeye sahip olduğumuzu düşünelim. Kafka sadece 125 mesaj gönderecek olmasına rağmen, her 2 saniyede 16 kez yaratılmış olacaktır. Bu işlem maliyetinden kurtulmak için sonraki çalışmada Serializable arayüzünden türeyen Kafka Üretici sınıfı kodlanmalı ya da ilgili çalışma varsa incelenmelidir.

Literatürde bulunan çalışmalar incelendiğinde, gerçek zamanlı ve yığın veri işleme için önerilmiş birçok çalışma ile karşılaşılacaktır; ancak uçtan-uca çözüm sunanlar oldukça azdır. Bu çalışmanın diğer çalışmalardan en büyük farkı, veri işleme sürecindeki her aşamaya çözüm üretmiş olmasıdır. SPEEDD [50] (Scalable ProactivE Event-Driven Decision) projesinin altyapı ve kullanılacak araçları belirlemek maksadıyla yapılmış olan çalışma bazı yönleriyle benzerlik taşımaktadır. Verinin kaynaktan alınması ve işlenmesi adımlarındaki teknolojiler incelenmiş; işlenen verilerin daha sonra makine öğrenmesi için kullanılacağı belirtilmiştir; ancak hangi veri tabanının kullanılacağı bilgisi bulunmamaktadır. Çalışma içerisinde; mesajlaşma platformları üzerine yapılan çalışma sonucu Kafka, veri işleme platformları üzerine yapılan çalışma sonucu ise Spark seçildiği bilgisine yer verilmiştir. Diğer bir çalışmada ise sensor verilerinin alınma, işlenme ve kaydedilme aşamalarında kullanılabilecek araçlar hakkında bilgiler verilmiştir [51]. Her aşama ayrı ayrı incelenmiş ve alternatifler hakkında kısa da olsa anlatım yapılmıştır. Sensorların haberleşme mekanizmaları hakkında detaylı bilgiler verilmiş olmasına rağmen, veri işleme aşamalarında kullanılabilecek araçlar hakkında teknik detaylara fazla yer verilmemiştir.

Model oluşturma ve çözüm üretme gerektiğinden dolayı Büyük Veri ile ilgili çalışmalarda incelenmiştir. Gerçek Zamanlı Büyük Veri İşleme için Çok-Temsilcili Mimari (Multi-Agent Mimari - MAM) [52] konulu çalışmada; Büyük Veri metodolojileri, yaklaşımları, zaman içerisinde değişikliğe uğrayan özellikleri hakkında bilgi vermektedir. Bunların yanı sıra Lamda Mimarisi detaylı olarak incelenerek MAM (MAS / Multi-Agent System) mimarisine sahip sistemlerin sahip olduğu yetenekler anlatılmıştır. Literatürde, Büyük Veri teknolojilerinden Hadoop'un görüntü işleme amaçlı kullanılması ile ilgili çalışma da mevcuttur. Videodan alınan verinin Hadoop kümesi üzerinde paralel olarak işlenmesinin amaçlandığı çalışmada [53]; okunan kareler üzerinde JavaCV kütüphanesi kullanılarak yüz

algılama ve hareket algılama algoritmaları koşulmuştur. Ayrıca küme içerisindeki köle (slave) sayısı değişikliğine göre performans ölçümlerine de yer verilmiştir.

Benzer konu ve teknik altyapıya sahip çalışmalar incelendiğinde, bu çalışmanın daha kapsamlı ve tutarlı çözümler üretmeye yönelik olduğu anlaşılacaktır. Seçilen tez konusu ve kapsamı, farklı amaçlar için hazırlanmış birçok çalışmanın içeriğine sahiptir. Ayrıca tercih edilen araçların seçiminde, tasarım ve geliştirme süresince, birçok çalışma ve araç incelenmiş; en uygun olanların belirlenmesine gayret edilmiştir. Bu açılardan bakıldığında elde edilen bilgiler büyük önem taşımaktadır.

5. SONUÇ VE ÖNERİLER

Güvenlik, insan yaşamında yasal düzenin sekteye uğramadan yürütülmesi, kişilerin hür, huzurlu ve korkusuzca yaşaması anlamına gelmektedir. Küreselleşme, ulaşım, iletişim ve bilgi teknolojilerindeki gelişmeler terörizmi, uluslararası platforma taşıyarak günümüz dünyasının en önemli problemi haline gelmiştir. Artan terör ve diğer olaylar, sınır güvenliği kavramının önemini bir kat daha artırmış ve yenilikçi önlemler alınmasını gerekli kılmıştır.

Akıllı sistemlere dayalı uygulamalar; veri desenlerini, şartlara göre gerçekleşen davranışları ve karmaşık analiz gerektiren durumları yorumlamak için sınır güvenliğini sağlamada güvenlik sistemlerinin yapı taşları haline gelmektedir. Bu tür uygulamalar anlamsal yorumun çağdaş bilişsel modellerini kullanarak yeni protokoller ve yaklaşımlar ortaya koymaktadır. Bu bağlamda gelişen ve değişen dünya düzenine uygun akademik ve endüstriyel çalışmalar, milli savunma yeteneklerini güçlendirmek için yeni çözümler üretebilir.

Bu tezde, sınır güvenliğinin sağlanmasına yardımcı olacak tutarlı, dayanıklı, dağıtık ve ölçeklendirilebilir bir sistem geliştirilmiştir. Analiz aşamasında, ilgili alanların içerdiği problemler tespit edilmiş ve nasıl çözülecekleri hakkında öngörüler yapılmıştır. Analiz, araştırma, geliştirme ve idame ettirme süreçlerinin sonucunda elde edilen sistem, gerçek zamanlı akan ve yığın veri üzerinde analiz gerektiren uygulamalar için, birlikte çalışabilen ve birbirini tamamlayan teknolojiler bütünü olarak düşünülebilir. Verinin kaynaklardan alınıp, son kullanıcıya gösterilmesine kadar olan gerekli her aşama ayrı ayrı değerlendirilmiştir. Büyük resim içerisindeki her kare, ayrı ayrı çözümlenmiş; ancak genel gereksinim ve hükümlere ters düşecek yaklaşımlardan kaçınılmıştır. Tasarlanan mimari ve altyapının belirlenmesi safhasında birçok kavram, mekanizma ve yaklaşım incelenmiştir. Uygun görülenler temel alınarak sistem içerisinde karşıt kalıp (anti-pattern) oluşmamasına özen gösterilmiştir.

Tez kapsamında, akan veri ve yığın veri üzerinde işlem yapmayı sağlayan Lambda Mimarisi temel alınarak hem gerçek zamanlı hem de geçmişe dönük veriler üzerinde analiz yapılmasına olanak sağlayan bir yapı elde edilmiştir. Zamanın ve kaynakların verimli

kullanılabilmesi için dağıtık bir sistem geliştirilmiş ve buna ek olarak sistem içerisinde gerçekleşebilecek tıkanıklıkların önüne geçmek için reaktif bir düzen sağlanmıştır. Mevcut sistemlerin sahip oldukları veri işleme yetenekleri düşünüldüğünde bu çalışma, yeni yaklaşımlar ve çözümler geliştirilmesine olanak sağlayan erken uyarı sistemi olabilecek bir tasarıma sahiptir.

Geliştirilen sistemin test edilebilmesi için örnek senaryolar oluşturulup elde edilen sonuçlar gözlemlenmiştir. Sensor verileri sisteme gönderilerek hedeflenen analizler yapılmış, hareket tespit edenlerin sistem tarafından bulunarak son kullanıcıya bilgi verdiği belirlenmiştir. Video verisi sisteme gönderildiğinde ise her karenin ayrı fiziksel makine tarafından işlenerek paralel çalışma prensibinin gerçekleştirildiği bilgisine ulaşılmıştır. Kodlanan web uygulamasındaki 3 boyutlu destek, uyarı mekanizması, zaman serisi şeklinde çalışma gibi yetenekler düşünüldüğünde, bu çalışmanın kapsamı ve yararı konusunda fikir sahibi olmak mümkündür.

KAYNAKLAR

1. Aksu, M. ve Turhan, F. (2012). Yeni Tehditler, Güvenliğin Genişleme Boyutları ve İnsani Güvenlik. *Uluslararası Alanya İşletme Fakültesi Dergisi*, 4(2), 69-80.

2. İnternet: Lockheed Martin. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.lockheedmartin.com%2Fus%2Fwho-we-are.html&date=2016-06-02, Son Erişim Tarihi: 02.06.2016.

3. İnternet: Persistent Threat Detection System (PTDS). URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.lockheedmartin.com%2Fus%2Fproducts%2Flighter-than-air-vehicles%2Fptds.html&date=2016-06-02, Son Erişim Tarihi: 02.06.2016.

4. İnternet: ASELSAN. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.aselsan.com.tr%2Fen-us%2Fabout-us%2FPages%2FDefault.aspx&date=2016-06-02, Son Erişim Tarihi: 02.06.2016.

5. İnternet: HAVELSAN. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.havelsan.com.tr%2F&date=2016-06-02, Son Erişim Tarihi: 02.06.2016.

6. İnternet: Poseidon. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.poseidon.com.tr%2F&date=2016-06-02, Son Erişim Tarihi: 02.06.2016.

7. Lilien, T., Othmane L., Angin, P., DeCarlo, A., Salih, M. Ve Bhargava, B. (2014). A Simulation Study of Ad Hoc Networking of UAVs with Opportunistic Resource Utilization Networks. *Journal of Network and Computer Applications*, 1(38), 3-15.

8. Choi, D., Ko, B. ve Kim, P. (2014). Text analysis for detecting terrorism-related articles on the web. *Journal of Network and Computer Applications*, 1(38), 16-21.

9. Ogiela, L. ve Ogiela, M. (2014). Cognitive systems and bio-inspired computing in Homeland Security. *Journal of Network and Computer Applications*, 1(38), 34-42.

10. Seo, S., Gupta, A., Sallam, A., Bertino, W. ve Yim, K. (2014). Detecting mobile malware threats to Homeland Security through static analysis. *Journal of Network and Computer Applications*, 1(38), 43-53.

11. Irgan, K., Unsalan, C. ve Baydere, S. (2014). Low-cost prioritization of image blocks in wireless sensor networks for border surveillance *Journal of Network and Computer Applications*, 1(38), 54-64.

12. Eker, G. ve Yılmaz, G. (2013). Kablosuz Algılayıcı Ağlar Kullanılarak Belirlenen Bir Bölgenin Çevre Güvenliğinin Sağlanması. *TBV Bilgisayar Bilimleri ve Mühendisliği Dergisi*, 6(1), 7-10.

13. Demchenko, Y., Laat de, C. ve Membrey P. (2014, 13-18 Mayıs). *Defining Architecture Components of the Big Data Ecosystem.* Paper presented at Collaboration Technologies and Systems (CTS), 2014 International Conference, Minneapolis.

14. İnternet: Leveraging Data for Agile Business. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fidc-cema.com%2Feng%2Fevents%2F50534-idc-big-data-and-business-analytics-forum-2013%3Fg_clang%3DENG&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

15. İnternet: Veri Madenciliği. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fweb.itu.edu.tr%2F%7Esgunduz%2Fcourses%2Fverimaden%2Fslides%2Fd1.pdf&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

16. İnternet: Classical and Advanced Techniques for Optimization. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fnptel.ac.in%2Fcourses%2FWebcourse-contents%2FIISc-BANG%2FOPTIMIZATION%2520METHODS%2Fpdf%2FModule_1%2FM1L4slides.pdf&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

17. İnternet: Data Mining Techniques. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.statsoft.com%2FTextbook%2FData-Mining-Techniques&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

18. Pop, D. (2016). Machine Learning and Cloud Computing: Survey of Distributed and SaaS Solutions. *Cornell University Library*, 1(1603.08767), 1-12.

19. Yavuz, G., Aytekin, S. ve Akçay, M. (2012). Apache Hadoop ve Dağıtık Sistemler Üzerindeki Rolü. *Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 1302-3055(25), 43-54.

20. Isard, M., Budiu, M., Yu, Y., Birrell, A. ve Fetterly, D. (2007). Dryad: distributed data-parallel programs from sequential building blocks. *ACM Digital Library*, 41(3), 59-72.

21. İnternet: Apache Mahout. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fen.wikipedia.org%2Fwiki%2FApache_Mahout&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

22. İnternet: Pentaho Business Analyst. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.pentaho.com%2Fproduct%2Fbig-data-analytics&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

23. İnternet: Skytree Server. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.skytree.net%2Fproducts-services%2Fskytree-server%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

24. İnternet: Tableau. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.tableausoftware.com%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

25. İnternet: Karmasphere Studio and Analyst. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.karmasphere.com%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

26. İnternet: Talend Open Studio 2014. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.talend.com%2Fproducts%2Ftalend-open-studio&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

27. Chen, Y., Alspaugh, S. ve Katz R. (2012). Interactive analytical processing in big data systems: a cross-industry study of MapReduce workloads. *ACM Digital Library*, 5(12), 1802-1813.

28. İnternet: Storm 2014. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fstorm.incubator.apache.org%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

29. Zaharia, M., Das, T., Li, H., Shenker, S. ve Stoica, I. (2012, 12-15 Haziran). *Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters.* Paper presented at HotCloud'12 Proceedings of the 4th USENIX conference on Hot Topics in Cloud Computing, Berkeley.

30. İnternet: Azure Event Hubs. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fazure.microsoft.com%2Ftr-tr%2Fservices%2Fevent-hubs%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

31. İnternet: Azure Stream Analytics. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fazure.microsoft.com%2Fen-us%2Fservices%2Fstream-analytics%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

32. İnternet: Kinesis. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fdocs.aws.amazon.com%2Fkinesis%2Flatest%2Fdev%2Fkey-concepts.html&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

33. İnternet: S4. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.slideshare.net%2Fabifet%2Fmining-bigdata&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

34. İnternet: Sqlstream. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.sqlstream.com%2Fproducts%2Fserver%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

35. İnternet: Splunk. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.splunk.com%2Fview%2Fbig-data%2FSP-CAAAGFH&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

36. Kreps, J., Narkhede, N. ve Rao, J. (2011, 12-16 Haziran). *Kafka: A distributed messaging system for log processing.* Paper presented at 6th International Workshop on Networking Meets Databases, Atina.

37. İnternet: SAP Hana. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.saphana.com%2Fwelcome&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

38. Melnik, S., Gubarev, A., Long, J., Romer, G., Shivakumar, S., Tolton, M. ve Vassilakis, T. (2012). Dremel: Interactive Analysis of Web-Scale Datasets. *ACM Digital Library*, 3(1-2), 330-339.

39. İnternet: Apache Drill. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fincubator.apache.org%2Fdrill%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

40. Gortan, I. ve Klein, J. (2014). Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems. *IEEE Software*, 32(3), 78-85.

41. Gortan, I. ve Klein, J. (2014, 5-9 Mayıs). *Big Data; Architectures and Technologies.* Paper presented at SEI Conference SATURN2014, Portland.

42. İnternet: Apache Cassandra. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fdocs.datastax.com%2Fen%2Fcassandra%2F2.1%2Fcassandra%2Farchitecture%2FarchitectureIntro_c.html&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

43. Engel, Y. ve Etzion, O. (2011, 11-15 Temmuz). *Towards proactive event-driven computing.* Paper presented at 5th ACM international conference on Distributed event-based system, New York.

44. İnternet: The Reactive Manifesto. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.reactivemanifesto.org%2F&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

45. Marz, N. ve Warren, J. (2015). *Big Data principles and best practices of scalable realtime data systems*.(1). New York/Amerika: Manning Publications, 14-14.

46. İnternet: Spark Streaming + Kafka Entegrasyonu. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fspark.apache.org%2Fdocs%2Flatest%2Fstreaming-kafka-integration.html&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

47. Kiran, M., Murphy, P., Monga, I., Dugan J. and Baveja, S. (2015, 29 Ekim - 01 Kasım). *Lambda architecture for cost-effective batch and speed big data processing.* Paper presented at Big Data (Big Data), 2015 IEEE International Conference, Santa Clara.

48. İnternet: JavaCV. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fgithub.com%2Fbytedeco%2Fjavacv&date=2016-05-20, Son Erişim Tarihi: 20.05.2016.

49. İnternet: Haar Özellikleri Tabanlı Sınıflandırma. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fdocs.opencv.org%2F2.4%2Fmodules%2Fobjdetect%2Fdoc%2Fcascade_classification.html&date=2016-05-20, Son Erişim Tarihi: 20.05.2016.

50. İnternet: The Architecture Design of the SPEEDD Prototype. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.speedd-project.eu%2Fsites%2Fdefault%2Ffiles%2FD6.1-Architecture_Design_of_SPEEDD_Prototype-v1.0a.pdf&date=2016-05-17, Son Erişim Tarihi: 17.05.2016.

51. Ronkainen, J. ve Livari, A. (2015). Designing a Data Management Pipeline for Pervasive Sensor Communication Systems. *Science Direct*, 56(1), 183-188.

52. Twardowski, B. ve Ryzko, D. (2014, 11-14 Ağustos). *Multi-agent Architecture for Real-Time Big Data Processing.* Paper presented at Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences, Warsaw.

53. Tan, H. ve Lidong, C. (2014, 14-18 Temmuz). *An Approach For Fast and Parallel Video Processing on Apache Hadoop Clusters.* Paper presented at 2014 IEEE International Conference on Multimedia and Expo (ICME), Chengdu.

54. İnternet: HDFS. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fhortonworks.com%2Fhadoop%2Fhdfs%2F&date=2016-05-20, Son Erişim Tarihi: 20.05.2016.

55. İnternet: Druid. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fdruid.io%2F&date=2016-05-20, Son Erişim Tarihi: 20.05.2016.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : Fatih, AYDEMİR
Uyruğu : T.C.
Doğum tarihi ve yeri : 31.05.1986, Sivas
Medeni hali : Evli
Telefon : 0 (544) 5853105
Faks : 0 (312) 2663550
E-Posta : faydemir@stm.com.tr

Eğitim

Derece	Okul/Program	Mezuniyet tarihi
Yüksek Lisans	Gazi Üniversitesi/Bilgisayar Mühendisliği	Halen
Lisans	9 Eylül Üniversitesi/Bilgisayar Mühendisliği	2010
Lise	Mehmet Erdoğan Anadolu Lisesi	2004

İş Deneyimi

Yıl	Çalıştığı Yer	Görev
2010-2011	Başar Bilgisayar Sist. Ltd. Şti. (Başarsoft)	Yazılım Mühendisi
2011-...	STM A.Ş.	Yazılım Mühendisi

Yabancı Dil

İngilizce

Hobiler

Yüzme, Uzakdoğu Dövüş Sporları

GAZİ GELECEKTİR...