VERİ BİLİMİ DERGİSİ
www.dergipark.gov.tr/veri

Dağıtık Mesajlaşma Altyapısı Kullanılarak Büyük Boyutlu Verilerin Gerçek Zamanlı Olarak İşlenmesi

Ahmet TOPRAK[1]*, Abdül Halim ZAİM[2]

[1]*İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği A.B.D., İstanbul*
[2]*İstanbul Ticaret Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul*

Makale Bilgisi

Başvuru:
17/07/2020
Kabul:
30/11/2020

Özet

Günümüzde IoT (Internet of Things – Nesnelerin İnterneti) cihazların kullanımındaki artış beraberinde yüksek yoğunluklu ve farklı çeşitte verilerin oluşmasına sebep olmuştur. IoT cihazlarından toplanan bu verilerin formatları, şekilleri ve yoğunlukları birbirinden tamamen farklıdır. Bu verilerin anlık olarak işlenmesi ve ilgili kullanıcıya anlık olarak iletilmesi gerekmektedir. Bu makalede, IoT cihazlarından elde edilen verilerin işlenmesi ve son kullanıcıya anlık olarak iletilmesi amacıyla bir model tasarlanmıştır. Çalışmada öncelikli olarak IoT cihazlarından toplanan yapısal olmayan veriler veri ön işleme adımlarına tabi tutulmuştur. Veri ön işleme adımları sonrası elde edilen verilerden anlamlı kelimeler tespit edilmiştir. Bu amaçla TF-IDF (Term Frequency-Inverse Document Frequency) metrikleri uygulanmıştır. Anlamlı kelime tespiti sonrası her anlamlı kelime konusuna göre verileri anlık işlemek amacıyla RabbitMQ dağıtık mesaj işleme kuyruğuna yönlendirilmiştir. Böylece mesajların iletilmesi garanti altına alınmıştır. RabbitMQ kuyruğuna iletilen mesajların anlık olarak alınması ve işlenmesi amacıyla Apache Storm topolojisi kullanılmıştır. Garantili mesaj işleme alt yapısı kullanan Apache Storm topolojisi, mesajları RabbitMQ dağıtık kuyruklama teknolojisi üzerinden okuyup, yapması gereken işlem ve hesaplamaları yaptıktan sonra çıktıları Elasticsearch içerisinde saklamıştır. Apache Storm topolojisi içerisinde üretilen sonuçlar daha sonra REST (Representational State Transfer) mimarisi kullanılarak son kullanıcı ile paylaşılmıştır.

Anahtar Kelimeler: *Nesnelerin İnterneti, Büyük Veri, RabbitMQ, TF-IDF Metrikleri, Apache Storm*

Real-Time Processing of Large Data Using a Distributed Messaging Infrastructure

Abstract

Today, the increase in the use of IoT (Internet of Things) devices has led to the formation of high density and different kinds of data. The formats, shapes and densities of this data collected from IoT devices are completely different from each other. This data needs to be processed instantaneously and transmitted instantly to the user concerned. In this article, a model is designed to process data obtained from IoT devices and transmit them to the end user instantly. In this study, non-structural data collected from IoT devices were subjected to data preprocessing steps. Significant words were determined from the data obtained after the

* İletişim e-posta: ahmetoprak190363@gmail.com
** Bu çalışmanın bir kısmı III. International Conference on Data Science and Applications 2020'de sözlü olarak sunulmuştur.

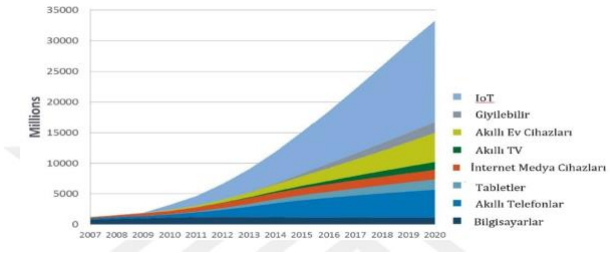
data pre-processing steps. For this purpose, TF-IDF (Term Frequency-Inverse Document Frequency) metrics were applied. After meaningful word detection, the subject of each meaningful word was redirected to the RabbitMQ distributed message processing queue to process the data instantly. This ensures that messages are delivered. Apache Storm topology is used to receive and process the messages transmitted to the RabbitMQ queue instantly. Using the guaranteed message processing infrastructure, the Apache Storm topology will read the messages through RabbitMQ distributed queuing technology and store the printouts in Elasticsearch after performing the necessary operations and calculations. The results generated in the Apache Storm topology will then be shared with the end user using the REST (Representational State Transfer) architecture.

Keywords: *Internet of Things, Big Data, RabbitMQ, TF-IDF Metrics, Apache Storm*

1 Giriş

Günümüzde teknoloji artık çok farklı bir noktaya gelmiştir. Anlık olarak üretilen veri miktarı oldukça yüksek seviyelere ulaşmıştır. Durum böyle olunca da, bu yüksek miktarlı veriyi işlemek anlamladırmak ve bilgiye dönüştürmek zorlaşmıştır. Veriyi raporlama, son kullanıcıya iletmek için yeni teknik ve yöntemler geliştirilmiştir. IoT cihazlarının artık daha çok kullanılmasıyla birlikte, toplumlar yeni yöntem ve arayışlara girmişlerdir. IOT, Internet of Things ifadesinin kısaltmasıdır ve "Nesneler İnterneti" anlamına gelir. İlk kez 1999 yılında Kevin Ashton tarafından hazırlanan bir sunumda bu kavram kullanılmıştır. Nesnelerin İnterneti kavramı; insan etkisine ihtiyaç duymadan, insanların beklentilerini ve ihtiyaçlarını karşılamak amacıyla ve interneti kullanan teknolojik aletler vasıtasıyla, aralarında veri alışverişi yapan sistemleri kapsamaktadır[1].

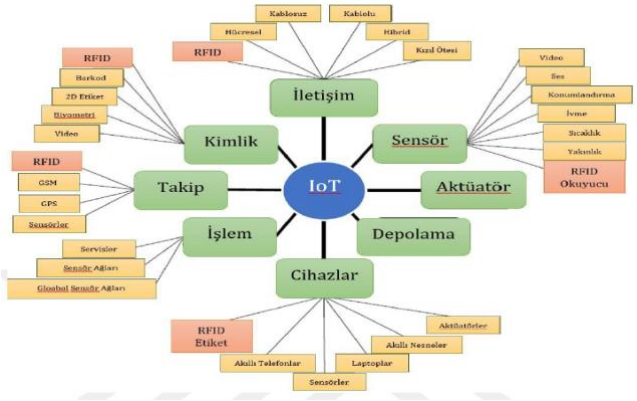
Ağ üzerindeki cihaz sayısının artması ile aşağıdaki Şekil 1'den de görüleceği üzere veri trafiğinde ciddi artış ve güvenlik riskleri oluşmaktadır. 2025'e kadar İnternete bağlı cihaz sayısının 70 milyarı geçmesi beklenmektedir. Şekil 1'de küresel internet cihazları ağ tahmini grafiği verilmiştir.



Şekil 1. Küresel internet cihazları ağ tahmini[2]

IoT teknolojisi, geniş kapsamlı alanlardan veri toplamak aynı zamanda o alanları kontrol edebilmek amacıyla M2M (Machine to Machine-Makineden Makineye) ve H2M (Human to Machine-İnsandan Makineye) haberleşmelerine imkân tanımaktadır. IoT teknolojisinin, akıllı kontrol sistemleri, yapay zeka uygulamaları ile birlikte kullanıldığı uygulamalar insan hayatına, konforuna, güvenliğine, sanayi ve tarım sektörüne ve diğer alanlara çok işlevler ve faydalar sağlayacaktır[3].

Nesnelerin İnterneti sekiz ana bileşenden oluşmaktadır. Bu ana bileşenlerin bütünlüğü, gizliliği, orijinalliği; sistemlerin değerlendirilmesi adına incelenmesi gereken konulardır. Aşağıda Şekil 2'de IoT bileşenleri verilmiştir.



Şekil 2. IoT bileşenleri[4]

Bu çalışmada, IoT cihazlarından toplanan verilerin anlık gerçek zamanlı olarak işlenmesi amacıyla bir iş akışı önerilmiş ve deneysel olarak sınanmıştır. Öncelikle IoT cihazlarından veriler yapısal olmayan bir formatta alınmıştır. Yapısal olmayan bu verilerin tek düzene getirilmesi amacıyla veri ön işleme adımlarına tabi tutulmuştur. Veri ön işleme sonrası elde edilen kelimelerin ağırlıklarını tespit etmek amacıyla TF-IDF metrikleri kullanılmıştır. Daha sonra bu kelimeler kullanılarak, anlam ağırlıklarına

göre RabbitMQ dağıtık mesaj kuyruklarına yönlendirilmiştir.

RabbitMQ kuyruklarına iletilen mesajların anlık olarak işlenmesi amacıyla Apache Storm kullanılmıştır. Apache Storm, veri ön işleme adımlarından geçirilip tek düzene getirilen verilerin anlık olarak analizini gerçekleştirerek, elde ettiği sonuçları Elasticsearch veritabanına kaydetmektedir. Bu işlemlerin sonrasında, işlenen verileri Elasticsearch veritabanından alıp kullanıcıya sunmak amacıyla REST API kullanılmıştır.

Makalenin ikinci bölümünde, benzer ya da bu çalışmada kullanılan adımları kapsayan çalışmalara değinilmiştir. Üçüncü bölümde, genel sistem topolojisi detaylı olarak açıklanmıştır. Dördüncü bölümde, üçüncü bölümde bahsedilen topoloji adımları açıklanmış, son bölümde ise sonuç ve gelecek çalışmalara değinilmiştir.

2 Literatür Taraması

IoT sistemleri ile ilgili geçmişten günümüze çalışmalar yapılmış ve bu çalışmalar farklı analiz ve teknikler içererek günümüze kadar gelmiştir. Bu çalışmaların en yenilerinden biri 2019 yılındaki Karen R. Sollins'in [5] çalışmasıdır. Bu çalışmada, güvenlik ve gizlilik gereklilikleri ile verilerin yenilikçi kullanımlarının kesiştiği noktada büyük verilerin toplanması, kullanılması ve yönetimindeki çatışma ele alınmıştır. Çalışmada, IoT cihazlarından toplanan verilerin incelenmesi sırasında, verilerin güvenlik ve gizlilik sınırlarının yeniden çizilmesi için yinelenen bir gözden geçirme ve yeniden tasarım süreci önerilmektedir.

IoT cihazları üzerinde büyük veri analizi yapan bir diğer çalışma da, Umar Ahsan ve ekibinin 2018 yılında yaptıkları [6] çalışmadır. Bu çalışmada, milyarlarca IoT cihazından anlık olarak elde edilen verilerin işlenmesi ile ilgili tartışılmaktadır. Makale, IoT protokolleri ve mimari yapısını, IoT'deki büyük verilerin rolünü gözden geçirmeyi amaçlamaktadır.

Son zamanlarda, IoT uygulamaları akıllı şehirler, hastaneler gibi konulara oldukça odaklanmaktadır. Bu çalışmalardan biri de, Victoria Moreno-Cano ve ekibinin 2015 yılındaki [7] çalışmasıdır. Bu çalışma, büyük verilerin akıllı şehirler için yararlarını ve algılanan verilerden bilgi keşfinin potansiyelini analiz etmektedir. Bu çalışmada, akıllı şehirleri iki senaryosu içerisinde, büyük veri analizi uygulamalarından bazı örnekler sunulmaktadır. Bunlardan biri de Murcia Üniversitesi SmartCampus'ta sunulan hizmetler olup detaylı olarak açıklanmaktadır. İkinci örnek ise, binlerce transit kart işleminin yapılması gereken bir tramvay servis senaryosuna odaklanmıştır. Her iki senaryoda da en uygun büyük veri tekniklerini uyguladıktan sonra elde edilen sonuçlar, binalarda enerji tüketimi ve konfor ve akıllı şehirler bağlamında ulaşım sıkışıklığı yönetimi gibi verimli hizmetlerin nasıl sağlanabileceğini göstermektedir.

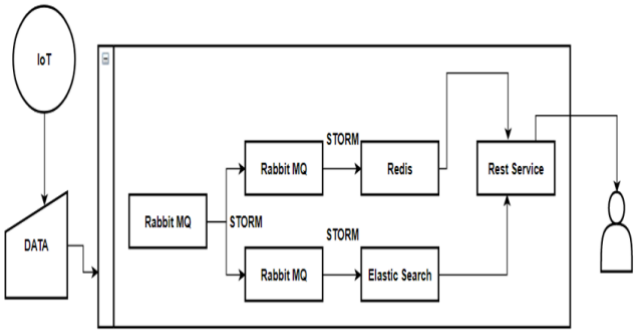
IoT uygulamalarının akıllı evler üzerindeki çalışmalarından biri de Muhammad Rizwan Bashir ve ekibinin 2017 yılı [8] çalışmasıdır. Bu çalışmada, Büyük Veri Analitiği alanındaki araştırma boşluğunu doldurmak için entegre bir IoT Büyük Veri Analizi (IBDA) çerçevesine ihtiyaç olduğu savunulmaktadır. Bu amaçla, akıllı binaya yerleştirilmiş IoT sensörlerinden üretilen gerçek zamanlı verilerin depolanması ve analizi için bir IBDA modeli tasarlanmıştır. Elde edilen sonuçlara göre, önerilen çerçevenin amaca uygun olduğunu ve akıllı binalar için IoT özellikli Büyük Veri Analitiğinin yararlı göründüğü belirtilmektedir.

Mehdi Mohammadi ve ekibinin 2018 yılında yaptıkları [9] çalışmada, IoT alanındaki analitiği ve öğrenmeyi kolaylaştırmak için derin öğrenme gibi gelişmiş bir makine öğrenme teknikleri sınıfının kullanımı hakkında kapsamlı bir genel bakış sunulmaktadır. Çalışmaya göre, IoT cihazlarından toplanan verilerin işlenmesi ve anlamlandırılması noktasında, derin öğrenme ile istenen analitiğe ulaşmak umut verici bir yaklaşım olarak görülmektedir.

IoT cihazlarından toplanan verilerin anlık olarak işlenmesi amacıyla yapılan Türkçe çalışmalar da bulunmaktadır. Bu çalışmalardan biri de Devrim Barış Acar ve ekibinin [10] çalışmasıdır. Bu çalışmada, IoT verilerinin işlenmesi için büyük veri mimarisi bileşenlerinin nasıl kullanıldığı, veri işleme hattı aşamaları üzerinde son bir yılda karşılaşılan problemler ve bu problemler özelinde geliştirilen çözümler açıklanmıştır.

3 Yöntem

IoT cihazlarından toplanan veriler sisteme dahil edilmeden önce veri ön işleme adımından geçirilir. Ön işleme adımından geçen kelimeler için daha sonra kelime ağırlıklandırma işlemi yapılır. Daha sonra konularına göre ilgili RabbitMQ dağıtık mesaj kuyruğuna yönlendirilir. Sonrasında bu mesajlar Apache Storm tarafından kuyruktan alınır ve anlık olarak işlenir. Son olarak ise mesajlar Elasticsearch veritabanına kaydedilir. Aşağıda sırasıyla uygulanan yöntemler açıklanmaktadır.

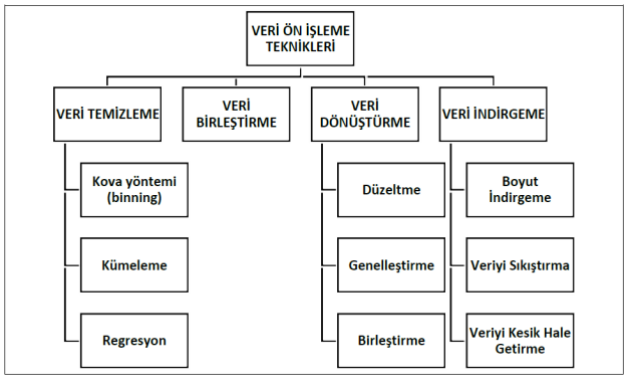


Şekil 3. Algoritma yaşam döngüsü

3.1 Veri ön işleme

İyi bir araştırma yapmak için kaliteli verilerle işlem yapılmalıdır. Bir verinin kalitesi o verilerle yapılacak olan çalışmanın başarı oranının artmasına olanak sağlamakta ve çalışmayı gelişime açık kılmaktadır. Verilerin özünün anlaşılması ve daha anlamlı bir veri analizi yapılabilmesi açısından veri ön işleme önemli bir kısımdır. Doğal dil işleme çalışmalarında ise, ön işleme teknikleri sisteme verilen dokümanlara uygulanarak, doküman içindeki kelimelerin standart bir forma dönüştürülmesi sağlanır. Veri ön işleme süreci ile ilgili yoğun çalışmalar yapılmaktadır. Bu çalışmalardan D. Tanasa [11] ve V. Chitraa [12] çalışmaları en bilinenleridir. Veri ön işleme süreci, yapılan çalışmanın kalitesine doğrudan etki ettiği için oldukça önemlidir. Veri ön işleme süreçleri genellikle veri temizleme (Hatalı verilerin dokümandan çıkarılması), veri birleştirme (Farklı kaynaklardan elde edilen verilerin bütünsel bir yapı haline getirilmesi), veri küçültme (Verinin boyutunun küçük parçalara ayrılması) ve veri dönüştürme(Verinin normalizasyon işlemine tabi tutulması) adımlarından oluşur.

Aşağıda Şekil 4'te veri ön işleme yöntemlerinin genel şeması verilmiştir.



Şekil 4. Veri ön işleme yöntemlerinin genel şeması

3.2 Anlamlı kelime tespiti

IoT cihazlarından toplanan sinyal bilgileri veri ön işleme adımına tabi tutulduktan sonra elde edilen veriler RabbitMQ mesaj kuyruklarına konularına göre gönderilmektedir. Bu işlemin amacı; tüm sinyalleri aynı kuyruğa gönderdiğimizde, dar boğaz oluşacak ve sistem üzerine yük bindiğinde veya farklı bir nedenden ötürü sistem çöktüğünde tüm süreç kesilmeyecek, sadece ilgili RabbitMQ kuyruğu devre dışı kalacaktır.

Çalışmada, anlamlı kelime tespiti amacıyla TF-IDF metrikleri kullanılmıştır. TF-IDF metrikleri bir doküman içerisindeki her bir kelimenin ilgili dokümandaki önemini tespit etmek amacıyla uygulanan matematiksel yöntemdir. TF-IDF metrikleri, TF(Term Frequency) ve IDF(Inverse Document Frequency) hesaplamalarından oluşur.

TF, bir terimin doküman içerisindeki frekansıdır. Bir kelime ilgili dokümanda ne kadar fazla geçiyorsa TF değeri o denli yüksektir.

IDF ise, dokümanlar arası sıklığı temsil eder. Bir kelimenin geçtiği doküman sayısının, toplam doküman sayısına logaritmik olarak bölünmesiyle elde edilir. Bu hesaplama sonucunda şu yorum yapılabilir. Bir kelime tüm dokümanlarda geçiyorsa içerdiği bilgi miktarı azdır denilebilir. TF-IDF metriklerini örnek üzerinden açıklamak gerekirse;

TF (Term Frequency): "V1" adında 1000 kelimeye sahip bir dokümanımız olsun. Bu doküman içerisinde "ICONDATA" kelimesinin 10 kez geçtiğini varsayarsak, "ICONDATA" kelimesinin TF değeri aşağıdaki gibi hesaplanır.

$$\mathrm{TF} = \frac{10}{1000} = 0.001$$

IDF (Inverse Document Frequency): 10 adet dokümanımız olsun. Bu dokümanların 5 tanesinde "ICONDATA" kelimesinin geçtiğini varsayarsak, "ICONDATA" kelimesinin IDF değeri aşağıdaki gibi hesaplanır.

$$\mathrm{IDF} = \log_e \frac{10}{5} = 0.3010$$

TF-IDF (Term Frequency—Inverse Document Frequency): TF ve IDF değerleri hesaplandıktan sonra TF-IDF hesaplaması yapılır. TF-IDF değeri, TF ve IDF değerlerinin çarpımına eşittir. Yukarıda elde

ettiğimiz TF ve IDF değerlerini baz alarak TF-IDF değerini hesaplarsak, sonuç aşağıdaki gibi olacaktır.

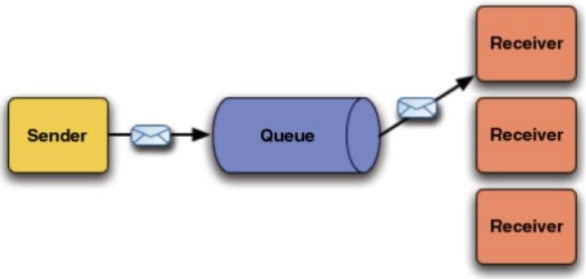
$$TF - IDF = 0.002 * 0.3010 = 0.000301$$

3.3 Verilerin mesaj kuyruğuna atanması

TF-IDF metrikleri ile ağırlıklandırılan veriler konularına göre RabbitMQ mesaj kuyruklarına gönderilmiştir.

RabbitMQ, Erlang dili ile açık kaynak olarak geliştirilmiş bir mesaj kuyruk yapısıdır [13]. Yani bir uygulamadan bir mesajı alıp bir başka uygulamaya sırası geldiğinde ileten sistemdir. Mesajlaşma protokolleri AMQP (Advanced Message Queuing Protocol), STOMP (Simple Text Oriented Message Protocol) ve MQTT'yi (Message Queuing Telemetry Transport) desteklemektedir. RabbitMQ, yönlendirme için kullanılabilecek birden fazla tür içermektedir.

Aşağıda Şekil 5'te RabbitMQ çalışma yaşam döngüsü verilmiştir.



Şekil 5. RabbitMQ çalışma yaşam döngüsü[13]

Sunucu, RabbitMQ sunucusuna bir mesaj gönderir. RabbitMQ dinleyici servisi bu mesajı ilgili kuyruğa yönlendirir. Sonrasında başka bir dinleyici servis bu kuyruğu dinler ve FIFO (First in First out) mantığıyla çalışan kuyruktaki bu mesajlar tüketilerek süreç sonlandırılır. Sahip olduğu Web Management Interface ile de mevcut kuyrukları görüntüleyebilme, tekrar kuyruğa alma, silme, tasfiye etme gibi daha bir çok işlemi yapabilmemizi sağlar.

Çalışmamızda RabbitMQ kullanmadan önce RabbitMQ gibi mesaj kuyruğu yönetim modeli olan KafkaMQ kullanılmıştır. Ancak yaptığımız incelemelerde KafkaMQ aşağıda belirtilen maddelere çözüm sağlamadığı için kullanımı devre dışı bırakılmıştır.

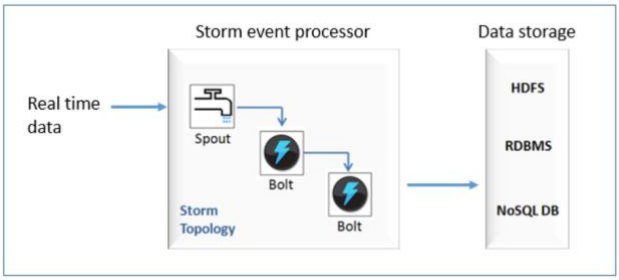
- KafkaMQ mesajın iletilmesini garanti etmez.
- KafkaMQ ile sadece Native Kafka Protocol protokolü kullanılmaktadır.

Geliştirme ortamınız .NET ise piyasadaki SDK'lardan birini kullanmak zorundasınız. Resmi olarak destek verilmemektedir[14].

3.4 Verilerin gerçek zamanlı işlenmesi

Veri ön işleme adımları sonrası RabbitMQ mesaj kuyruklarına gönderilen verilerin anlık olarak analiz edilmesi amacıyla çalışmamızda Apache Storm altyapısı kullanılmıştır. Apache Storm, ücretsiz ve açık kaynak dağıtılmış gerçek zamanlı hesaplama sistemidir[15]. Storm, Hadoop'un toplu işlemde yaptıklarını gerçek zamanlı olarak ve sınırsız veri akışlarını güvenilir şekilde işlemede kolaylık sağlar. Kullanımı karmaşık değildir. Storm ayrıca ilk seferde başarılı bir şekilde işlenmemiş verileri yeniden yürütme özelliğiyle birlikte verilerin garantili işlenmesini sağlayabilir.

Apache Storm, özellikle yüksek veri hızı ile dağıtılmış makine öğrenimi, gerçek zamanlı analiz ve diğer birçok durum için kullanılmaktadır. Apache Storm, YARN üzerinde çalışır ve Hadoop ekosistemleri ile bütünleşiktir. Küçük partiler halinde bir dizi yerine, bir akış işleme motorudur. Apache Storm düşük gecikme süresine sahiptir ve tek bir varlık olarak alınması gereken verilere çok uygundur. Apache Storm, toplu işleme ve akış işleme arasındaki bir köprüdür ve Hadoop, üzerinde işlenecek şekilde tasarlanmamıştır. Apache Storm bir milyon yüz bayt msgs / sn / node işlemek için kullanılabilir ve güvenilirdir. Her bir veri biriminin (tuple) bir kez işleneceğini garanti eder. Mesajlar sadece hata olduğunda tekrarlanır[16].



Şekil 6. Apache Storm çalışma yaşam döngüsü[17]

3.5 İşlenen verilerin kaydedilmesi

RabbitMQ mesaj kuyruklarında bulunan verilerin anlık olarak gerçek zamanlı işlenmesi sonrasında, bu verilerin kaybolmaması için kaydedilmesi gerekmektedir. Çalışmamızda bu amaçla Elasticsearch veritabanı kullanılmıştır.

Elasticsearch, Apache Lucene altyapısı üzerine kurulmuş, Java dilinde yazılmış açık kaynak (open source) kodlu bir tam metin (full text) arama motoru ve veri analiz aracıdır. Veri saklama biçimi ilişkisel değil doküman merkezlidir . [18]

Elasticsearch üzerinde tutulan veriler JSON (JavaScript Object Notation) formatında saklanır ve aramalar bu JSON dokümanları içerisinde yapılır. JSON dokümanlar Elasticsearch veritabanında kaydedilmeden önce hangi kelimenin hangi dokümanda geçtiği indekslenir. Elasticsearch veritabanında aramalar bu oluşturulan indeks listeleri üzerinden yapılır. Bir kelime aranmak istendiğinde doğrudan bu indeks listesine bakılır ve anında cevap verilir. Performanslı çalışmasını bu indeks yapısına borçludur.

Elasticsearch'ün sağladığı avantajlar aşağıdaki gibidir.

- Cluster yapısı çok basittir.
- Rakiplerine göre çok fazla kaynak tüketmez.
- Kendi içinde yüksek erişilebilirlik (high availability) sunar.
- İndeksleme mantığı olduğu için hızlı arama yapar.
- Doküman ve indeksleme kavramı çok kullanılır.
- Dokümanları JSON formatında indekslemektedir.

Aşağıda Şekil 7'de Elasticsearch veritabanında verilerin saklanma şekli görülmektedir.

Term	Documents
and	<6>
big	<2> <3>
dark	<6>
did	<4>
gown	<2>
had	<3>
house	<2> <3>
in	<1> <2> <3> <5> <6>
keep	<1> <3> <5>
keeper	<1> <4> <5>
keeps	<1> <5> <6>
light	<6>
never	<4>
night	<1> <4> <5>
old	<1> <2> <3> <4>
sleep	<4>
sleeps	<6>
the	<1> <2> <3> <4> <5> <6>
town	<1> <3>
where	<4>

Şekil 7. Elasticsearch veritabanında terimlerin indekslenme yapısı [19]

3.6 İşlenen verilerin son kullanıcıya iletilmesi

İşlenen verilen Elasticsearch veritabanına kaydedildikten sonra bu verilerin son kullanıcıya iletilmesi gerekmektedir. Bu işlemin son derece hızlı ve güvenlikli şekilde gerçekleştirilmesi oldukça önemlidir. Bu amaçla çalışmamızda REST API kullanılmıştır. Elasticsearch veritabanındaki veriler ilgili API kullanılarak son kullanıcıya sunulmuştur.

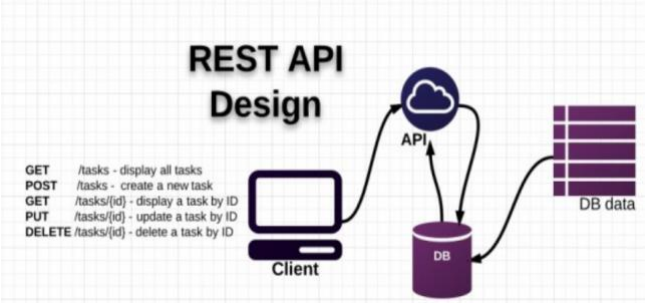
REST(Representational State Transfer) ifadesi, ilk kez Roy Fielding tarafından yazılan doktora tezinde geçmektedir. REST mimarisi dağıtık sistemlerin haberleşmesini sağlamak amacıyla tasarlanmıştır. REST, istemci-sunucu mantığıyla çalışır. İstemci üzerinden yapılan istekler sunucuya HTTP (Hyper Text Transfer Protocol) protokolü üzerinden geçer.

İletilen veri XML (Extensible Markup Language) ya da JSON formatında olabilir. REST durumsuz (stateless) çalışır. Yani yapılan isteğin durumunu mesaj içeriğinde tutmaz. Ayrıca REST isteklerinde mesajın içeriğinde ekstra başlık bilgileri gönderilmez. REST mimarisi ile haberleşen servislere ise RESTful servis denir.

- Rest API özellikleri aşağıdaki gibidir.

- REST API, istemci-sunucu iletişimiyle ilgili bir mimaridir.
- REST API, SOAP (Simple Object Access Protocol), RPC (Remote Procedure Call) gibi kompleks mimariler yerine HTTP protokolü üzerinden işler.
- REST API, SOAP, RPC'nin aksine basit ve hızlıdır.
- REST API'nin SOAP gibi katı standartları yoktur.
- SOAP üzerinde güvenlik sağlamak katı kurallar sebebiyle daha kolay ve hızlı bir şekilde sağlanabilirken, REST için karmaşık bir hale gelebilmektedir.
- REST API, SOAP gibi proxy kullanmaya ve WSDL'e (Web Services Description Language) gereksinim duymaz.
- SOAP, güvenlik protokollerini bünyesinde barındırır ve state bilgisini talep ve yanıtlarda saklar.

Aşağıda Şekil 8'de REST API çalışma yaşam döngüsü verilmiştir.



Şekil 8. REST API çalışma yaşam döngüsü [21]

4 Sonuçlar

Bu çalışmada, IoT cihazlarının oluşturduğu verilerin gerçek zamanlı olarak son kullanıcıya iletilmesi amacıyla bir model önerilmiştir. Çalışmamızda kullandığımız tüm teknolojiler günümüzün en popüler ürünleridir. Bu yönüyle çalışmamız diğer çalışmalardan ayrılmaktadır. Yaptığımız deneylerde bu ürünleri doğrudan kullanamadığımız durumlar meydana gelmiştir. Bu nedenle API'ler üzerinde özel çözümler üretilmiştir. İzlenimlerimize göre RabbitMQ mesaj kuyruğu anlık olarak bir milyon veriyi kuyruklara dağıtabilmektedir. Ancak daha fazla mesaj içeren durumlar için kurumsal paketleri satın almak gerekmektedir. Bir milyondan fazla mesajı kuyruklara dağıttığımızda, sistemin performansına negatif yönde etki ettiği görülmüştür. Bunun yanısıra Apache Storm tasarımı hatalı yapıldığında, sistemin isteklere cevap veremez duruma gelebildiği gözlemlenmiştir. Elasticseach veritabanında saklanan veriler yapısal olarak tutulmadığı için hız bakımından oldukça iyi olduğu görülmüştür. Elasticsearch devre dışı bırakılıp yerine yapısal bir veritabanı (MS SQL) koyduğumuzda sonuçların oldukça kötü olduğu görülmüştür.

Literatüre bundan sonraki süreçlerde katkı sağlayacağı düşünülen aşağıdaki çalışmalar ele alınabilir.

- IoT cihazlarından elde edilen verilerin ağırlıklandırılması amacıyla Helmholtz prensibi kullanılabilir.
- RabbitMQ haberleşme sistemi yerine, MQTT asenkron haberleşme sistemi kullanılabilir.
- Verilerin gerçek zamanlı işlenmesini sağlamak amacıyla kullanılan Apache Storm yerine Apache Hadoop kullanılabilir.
- İşlenen verilerin kaybolmaması amacıyla kaydedildiği Elasticsearch veritabanı yerine Redis veritabanı kullanılabilir.

İşlenen verileri son kullanıcıya iletmek için kullanılan REST API yerine SOAP servisleri kullanılabilir.

Kaynaklar

[1] Mediaclick."IoTnedir?".https://www.mediaclick.com.tr/blog/iot-nedir (14.11.2020).
[2] Prnewswire."NewsReleases".https://www.prnewswire.com/newsreleases.html (14.11.2020).
[3] Beyaz,"IoTnedir?".https://www.beyaz.net/tr/arge/makaleler/iot_nedir.html(14.11.2020).
[4] Christoph, P. M. (2009). Security and privacy challenges in the internet of things. Proc WowKiVS, 2009-1.
[5] K. R. Sollins, "IoT Big Data Security and Privacy Versus Innovation," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1628-1635, April 2019. doi: 10.1109/JIOT.2019.2898113
[6] U. Ahsan and A. Bais, "A Review on Big Data Analysis and Internet of Things," 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Brasilia, 2016, pp. 325-330. doi: 10.1109/MASS.2016.048
[7] V. Moreno-Cano, F. Terroso-Saenz and A. F. Skarmeta-Gómez, "Big data for IoT services in smart cities," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 418-423. doi: 10.1109/WF-IoT.2015.7389091
[8] M. R. Bashir and A. Q. Gill, "Towards an IoT Big Data Analytics Framework: Smart Buildings Systems,"

2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, NSW, 2016, pp. 1325-1332. doi: 10.1109/HPCC-SmartCity-DSS.2016.0188

[9] M. Mohammadi, A. Al-Fuqaha, S. Sorour and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," in IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 2923-2960, Fourthquarter 2018. doi: 10.1109/COMST.2018.2844341

[10] Acar, D.B,Yılmaz, A. K, Küçükbay, S. R. (2018). IoT Verileri İçin Gerçek Zamanlı ve Ölçeklenebilir Büyük Veri Mimarisi: Karşılaşılan Problemler ve Geliştirilen Çözümler.UYMS'18,2018-1.

[11] Tanasa, D., Trousse, B., 2004. Advanced data preprocessing for intersites web usage mining, IEEE Intelligent Systems, 19(2), 59–65.

[12] Chitraa, V., Dr. Davamani, A. S., 2010. A Survey on Preprocessing Methods for Web Usage Data. International Journal of Computer Science and Information Security, 7(3), 78-83.

[13] Caner Tosun. "RabbitMQ Nedir? Windows Üzerinde Kurulumu".http://www.canertosuner.com/post/rabbitmq-nedir-windows-uzerinde-kurulumu (16.07. 2020).

[14] Medium.Kafka vs. RabbitMQ" https://medium.com/kariyertech/kafka-vsrabbitmq-abe52d5eee34 (16. 07.2020).

[15] Oguzhan İnan. "Hadoop Ekosistemi ve Kullanılan Araçlar". https://oguzhaninan.gitlab.io/Hadoop-Ekosistemi-ve-Kullanilan-Araclar/#what-is-storm (14.11.2020).

[16] Womaneng. "Kafka-Flink-Storm-Platformları". https://womaneng.com/kafkaflinkstormplatformlari/ (16. 07.2020).

[17] Büyükveri. "Büyük Veri Ekosistemi". http://www.buyukveri.co/buyuk-veri-ekosistemi/ (14.11.2020).

[18] Devnot. "Bir Bakışta ElasticSearch". http://devnot.com/2017/bir-bakistaelasticsearch/ (14.11.2020).

[19] Mehmet Ayhan. "Elasticsearch nedir?". http://mehmetayhan.com.tr/yazi/elasticsearch nedir (14.11.2020).

[20] Burcu Altınok." REST API Nedir?". https://burcualtinok.com.tr/blog/rest-api-nedir/ (14.11.2020).

[21] Arif Gökçe. "REST API Nedir? Nasıl Kullanılır?". http://arifgokce.net/post/2017/08/07/REST-API-Nedir-Nasıl-Kullanılır- (14.11.2020).