T.C.
İNÖNÜ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI VERİ İŞLEME PLATFORM TASARIMI (TweetCASP)

YÜKSEK LİSANS

Tuğba Beril Doğuç

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğretim Üyesi Ahmet Arif AYDIN

HAZİRAN 2022

T.C
İNÖNÜ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GERÇEK ZAMANLI VERİ İŞLEME PLATFORM TASARIMI (TweetCASP)

YÜKSEK LİSANS

Tuğba Beril Doğuç

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğretim Üyesi Ahmet Arif AYDIN

HAZİRAN 2022

TEŞEKKÜR VE ÖNSÖZ

Bu tez çalışmasının her aşamasında yardım, öneri, bilgi, tecrübe ve desteklerini esirgemeden beni her konuda yönlendiren danışman hocam Sayın Dr. Öğretim Üyesi Ahmet Arif AYDIN'a, desteğini esirgemeyen rahmetli dedem Tarı BEKİL'e, çalışmalarımda ayrıca tüm hayatım boyunca olduğu gibi bu çalışmalarım süresince de benden her türlü desteklerini esirgemeyen aileme, teşekkür ederim.

ONUR SÖZÜ

Yüksek lisans tezi olarak sunduğum “Gerçek Zamanlı Veri İşleme Platform Tasarımı” başlıklı bu çalışmanın bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın tarafımdan yazıldığına ve yararlandığım bütün kaynakların hem metin içinde hem de kaynakçada yöntemine uygun biçimde gösterilenlerden oluştuğunu belirtir, bunu onurumla doğrularım.

Tuğba Beril Doğuç

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

SEMBOLLER VE KISALTMALAR

SQL	**:** Structured Query Language (Yapılandırılmış Sorgu Dili)
RDBMS	**:** Relational Database Management Systems
NoSQL	**:** Not Only SQL (SQL'den daha fazlası)
IoT	**:** Internet Of Things (Nesnelerin İnterneti)
CAP	**:** Consistency – Availability – Partition Tolerance(Tutarlılık - Erişilebilirlik - Bölümleme Toleransı)
IDCAP	**:** Incremental Data Collection and Analytics Platform(Artımlı Veri Toplama ve Analitik Platformu)
MR	**:** MapReduce (Haritalama ve İndirgeme)
CQL	**:** Cassandra Query Language
API	**:** Application Programming Interface (Uygulama Programlama Arayüzü)
JSON	**:** Java Script Object Notation (Java Script Nesne Gösterimi)
AR	**:** Augmented Reality (Artırılmış Gerçeklik)
VR	**:** Virtual Reality (Sanal Gerçeklik)
STOMP	**:** Streaming Text Oriented Messaging Protocol(Akan Metin Yönelimli Mesajlaşma Protokolü)
AMQP	**:** Advanced Messaging Queuing Protocol(Gelişmiş Mesajlaşma Kuyruk Protokolü)
KNN	**:** K-Nearest Neighbors (En Yakın Komşu Algoritması)

ÖZET

Yüksek Lisans Tezi

GERÇEK ZAMANLI VERİ İŞLEME PLATFORM TASARIMI
(TweetCASP)

TUĞBA BERİL DOĞUÇ

İnönü Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

36+VIII sayfa

2022

Danışman: Dr. Öğr. Üyesi Ahmet Arif AYDIN

Gelişen teknoloji ile birlikte sürekli gelişip değişen internet, yazılım ve donanım teknolojilerinin olduğu bir çağda yaşamaktayız. Toplum, bu gelişmelere hızlıca adapte olup gelişen teknolojinin bir parçası olmakta ve Büyük Veri'yi beslemektedir. Mobil uygulamalar, web sayfaları, e-mail, IoT ve daha birçok alanda üretilen ve kullanılan veriler büyük verinin oluşmasına katkı sağlamaktadır. Çeşitli kaynaklardan elde edilen farklı hız, boyut ve formattaki verilerin kaybedilmeden toplanması, istenilen formlara dönüştürülmesi, ihtiyaç anında erişilebilir olması, uygun veritabanı teknolojileri ve veri modelleri yardımıyla ölçeklenebilir bir biçimde depolanması ve gerektiğinde analiz edilip faydalı çıkarımların elde edilebilmesi ve günümüzde çok çeşitli ihtiyaçlara cevap verilebilmesi için önem arz etmektedir. Bu tez çalışmasında, Twitter üzerinden gerçek zamanlı olarak istenilen anahtar kelimelere göre veri toplanmasını, toplanan verinin gerçek zamanlı olarak işlenmesini ve ölçeklenebilir bir biçimde verinin NoSQL veri depolama teknolojilerinden Apache Cassandra kullanılarak depolanmasını ve elde edilen verinin işlenmesi için gerçek zamanlı veri işleme platformu(TweetCASP) geliştirilmiştir. Bu platform yardımıyla Twitter'da anlık oluşturulan tweetleri belirlenen kelimelere göre toplayıp, elde edilen tweetler içerisinde farklı kelimelere göre analiz yapılabilmektedir. Böylelikle geliştirilen platform farklı amaçlar için veri toplanması ve analizine imkan sağlamaktadır. Ayrıca, geliştirilen platform gerçek zamanlı veri işlemede kullanılabilecek teknolojilerin bir temsilini sunmaktadır.

Anahtar Kelimeler: Büyük Veri, Akan Veri, Twitter, Veri Analitiği, Veritabanı, CAP, NoSQL

ABSTRACT

Master Thesis

REAL TIME DATA ANALYTICS PLATFORM DESIGN
(TweetCASP)

TUĞBA BERİL DOĞUÇ

Inonu University
Graduate School of Nature and Applied Sciences
Department of Computer Engineering

36+VIII pages

2022

Supervisor: Assist. Prof. Ahmet Arif AYDIN

We live in an age where internet, software and hardware technologies are constantly developing and changing with the developing technology. The society quickly adapts to these developments and becomes a part of the developing technology and feeds Big Data. The data produced and used in mobile applications, web pages, e-mail, IoT and many other fields contribute to the formation of big data. In order to collect data in different speeds, sizes and formats obtained from various sources without losing, converting them into desired forms, making them accessible when needed, storing them in a scalable manner with the help of appropriate database technologies and data models, analyzing when necessary and obtaining useful inferences, and meeting a wide variety of needs today. In this thesis, a real-time data processing platform(TweetCASP) has been developed to collect data on Twitter in real time according to the desired keywords, to process the collected data in real time and to store the data in a scalable manner using Apache Cassandra, one of the NoSQL data storage technologies, and to process the obtained data. With the help of this platform, tweets created instantly on Twitter can be collected according to the determined words, and analysis can be made according to different words in the tweets obtained. Thus, the developed platform enables data collection and analysis for different purposes. In addition, the developed platform offers a representation of technologies that can be used in real-time data processing.

Keywords: Big Data, Streaming Data, Twitter, Data Analytics, Database, CAP, NoSQL

1. GİRİŞ

Günümüz dünyasında baş döndürücü bir hızla değişen ve gelişen teknoloji aynı zamanda toplumu birçok yönden etkilemektedir. Örneğin, dijital dönüşüm insanların gündelik ihtiyaçları, yaşam tarzları, alışkanlıkları, sosyal hayatlarının devamı için gereken işlemlerin gerçekleştirilmesini etkilemiştir. Bu teknolojik değişim toplumun hemen her kesiminde iş dünyasında, sağlık alanında, devlet kurumlarında ve diğer alanlarda veri merkezli dönüşüme sebep olmuştur. Dolayısıyla toplumsal hayatta hemen her alandaki işlemler gerçekleştirirken veri üretimi gerçekleştirilmektedir.

"Veri, ham (işlenmemiş) gerçek enformasyon parçacığına verilen addır." (Bocij, Greasley, & Hickie, 2015). Büyük Veri ise, 1979 yılında kurulan, dünya çapında yüzden fazla ülkede binlerce kuruluşun güvendiği nesnel bir danışmanlık firması olan Gartner'ın 2001 yılında yaptığı tanıma göre "*büyük veri, gelişmiş bilgi, karar alma ve süreç otomasyonu sağlayan düşük maliyetli, yenilikçi bilgi işlem biçimleri gerektiren yüksek hacimli, yüksek hızlı ve/veya çok çeşitli bilgi varlıklarıdır*". Günlük hayatımızın bir parçası haline gelen teknoloji sayesinde büyük veri, sürekli olarak ve çeşitli formlarda oluşturulmaktadır. Büyük veri teriminin, endüstri analisti Doug Laney'in büyük veriyi "3V" olarak tanımlamasıyla popülerliği artmıştır (Khan et al., 2014; Yaqoob et al., 2016). Burada bahsedilen 3V, "Volume, Velocity, Variety" olarak tanımlanmıştır.

- **Volume (Hacim)** **:** Büyük verinin devasa boyuttaki verilerden oluşmasını ifade etmektedir.
- **Velocity (Hız)** **:** Büyük verinin bir kaynaktan alınırken verinin elde edilme hızı (speed) olarak tanımlanmaktadır.
- **Variety (Çeşitlilik)** **:** Farklı kaynaklardan elde edilen verinin farlı format ve biçimlerde olmasını ifade etmektedir. Örneğin, Büyük veri metin belgelerinden e-postalara, ses dosyalarından videolara, bankacılık işlemlerinden firma anketlerine kadar her alanda veri tipini içermektedir.

Ayrıca yukarıda tanımlanan 3V' den başka birçok çalışmada ifade edilen tanımlamalar "Veracity" ve "Value" aşağıda açıklanmıştır:

- **Veracity (doğruluk) :** Çok çeşitli kaynaklardan elde edilen verinin güvenilir olmasını ifade etmektedir. Çok önemlidir çünkü gerçek olmayan bir kaynaktan üretilen yanlış veriler ile gerçekleştirilen analizlerin doğru olması beklenemez.
- **Value (değer)** : Veriden çeşitli analiz yöntemleri kullanılarak faydalı çıkarımların elde edilmesidir. Büyük verinin karakteristiklerinin (büyük verinin V' leri) ortaya çıkan bütün zorluklarının giderilmeye çalışılmasındaki temel amaç faydalı bilgilerin ortaya çıkarılabilmesidir.

Öte yandan, bu derece büyük boyutta, çeşitlilikte ve büyüme hızına sahip verinin doğru ve hızlı bir şekilde analiz edilmesi oldukça önemlidir. Büyük veriyi toplayıp depolayan ve analiz eden organizasyonların temel amacı belirli bir konu ile alakalı (potansiyel kullanıcılara ürün tanıtımı yapmak gibi) faydalı çıkarımlarda bulunabilmektir. Fakat, elde edilen çıkarımların doğru zamanda ve doğru yerde yapılabilmesi elzemdir. (Barlow, 2013) Bu sebeple, günümüzde akan veri analitiği yığın veri analitiğine göre daha fazla ilgi görmeye başlamıştır.

Akan veri, farklı kaynaklardan sürekli olarak düşük boyutlarda gelen verilerden oluşur. Akan verinin işlenmesi ve analiz edilmesi ise günümüzde popülerliği hızlı artan bir alandır. Gelişen teknoloji ve bu teknolojiye ayak uyduran insanların sayısının artmasıyla birlikte firmalar tarafından akan veri ve analizinin önemi artmış ve birçok farklı alanda kullanılmaya başlanmıştır. Örneğin, ürün fiyatlandırma, memnuniyet ve popülerlik analizi, yeni ürün tavsiyesi, IoT, acil durumlar ve afet yönetimi gibi alanlarda hem kamu kuruluşları hem özel sektör kurumları akan veri analitiğinden faydalanılmaktadır. Bu durum, akan verilerin sağlıklı bir şekilde alınıp analiz edilmesinin önemini anlayıp bu alanda yapılan çalışmaların sayısını artırmaktadır. Örneğin IDCAP (Aydin & Anderson, 2017), Apache Spark, Redis, RabbitMQ, ve Cassandra teknolojilerini kullanarak Twitter üzerinden güncel konular ile alakalı tweetlere anahtar kelimeler yardımıyla erişip akan tweet verileri üzerinde istenilen analizleri yapabilmek amacıyla tasarlanmış platformlardan biridir.

Bu tez çalışmasında kullanılan teknolojiler, akan veri analitiğini gerçekleştirmek için kullanılan teknolojiler ve bu teknolojiler ile uyumlu bir şekilde çalışabilen NoSQL

veritabanlarının, CAP teoremi baz alınarak araştırılıp incelenmesiyle seçilmiştir. Bu tez çalışması ile Twitter mikrobloging platformunun sağladığı Streaming API yardımıyla gerçek zamanlı olarak istenilen anahtar kelimelere göre tweet'lere erişim sağlandıktan sonra gerçek zamanlı veri analizi için geçici depolama teknolojilerinden RabbitMQ'ya sonrasında da kullanıcı isteğine göre gelecekteki yığın veri işleme taleplerini geröekleştirebilmek için populer NoSQL teknolojilerinden Apache Casandra kullanılmıştır. Böylelikle örnek bir gerçek zamanlı veri işleme platformu tasarlamıştır.

Bu tez çalışmasında, 2. Bölümde akan veri analitiğinde kullanılan teknolojilerden NoSQL veritabanlarına, bu veritabanlarının sınıflandırılmasında faydalanılan CAP teoremine, akan veri ve yığın verisinin anlatımına yer verilmiş, sistemin yazılım tarafı ve genel kavramlar açıklanmıştır. 3. Bölümde Literatür taramasına yer verilmiştir. 4. Bölümde akan veri işlemede kullanılan veri depolama teknolojileri anlatılmıştır 5. Bölümde sistemin çalıştığı platformun tasarımı ve detayları anlatılmıştır. 6. Bölümde sistemin değerlendirilmesine yer verilmiştir. 7. Bölümde ise sistemin sonuç kısmının açıklaması yapılmış ve ileride yazılım açısından neler eklenebileceğinden bahsedilmiştir.

2. GENEL KAVRAMLAR

Bu bölümde veri işleme, veri depolama ve veri analizi ile alakalı genel kavramlara yer verilmiştir. Bu bölümdeki kavramlar tez içerisinde Sırasıyla akan veri işleme (streaming data processing) ve akan veri işleme teknolojileri, yığın veri işleme (batch data processing) ve kullanılan teknolojiler, veri depolama kavramı kullanılan teknolojiler ve CAP teoremi açıklanmıştır.

2.1 Akan Veri işleme

Akan veri çeşitli kaynaklardan anlık olarak, küçük boyutlarda elde edilen verilerden oluşur. Akan veri, gelişip hızlanan teknojiyle birlikte doğru veriye hızlı bir şekilde ulaşıp filtreleme, arama ve analiz etme gibi ihtiyaçların artmasıyla ön plana çıkmıştır. İnsanların gelişen teknolojiye ayak uydurmasıyla birlikte kullanımı yaygınlaşan cep telefonu, bilgisayar, tablet, akıllı saatler, elektronik kartlıklar ve daha birçok farklı teknolojik üründen anlık olarak toplanan veriler, Twitter, Facebook, Instagram, Netflix, YouTube, Whatsapp, TikTok gibi platformlarda sürekli akış halinde olan ses, metin, resim ve video verileri akan verinin bir parçasını oluşturmaktadır. Akan veriyi doğru ve hızlı bir şekilde işleyebilmek, toplumun güncel ilgi alanlarını veya bir konu üzerindeki popüler düşüncelerin neler olduğunu öğrenip analiz edebilmek açısından oldukça önemlidir. Günümüzde kamu ve özel sektörde faaliyet gösteren çeşitli kurumlar, toplumda popüler olarak kullanılan ürünlerin çeşitlerini öğrenebilmek, piyasaya yeni sürülecek olan bir ürünün toplum tarafından nasıl karşılanacağı konusunda fikir edinebilmek, doğru yerde doğru reklamı yapabilmek, olası bir doğal afet durumunda kurtarma ekiplerinin çalışma yapacağı yerlerin hızlı bir şekilde belirlenebilmesi, salgın hastalıklarda hastalığın yayıldığı bölgelerin doğru tespit edilebilmesi gibi birçok farklı amaçla akan veriyi toplayıp, filtreleyip analiz etmektedir. Akan verinin işlenip analiz edilmesinde kullanılan teknolojilerin en popülerlerinden biri Apache Spark teknolojisidir.

2.1.1 Apache Spark

Apache Spark[1], açık kaynak kodlu veri analizi gerçekleştirmeye yarayan bir teknolojidir. 2009 yılında Matei Zaharia tarafından Berkeley Üniversitesi AMPLab

1

laboratuvarında geliştirilmeye başlanmıştır (Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010). Apache Hadoop'un MR tabanlı modelinden farklı olarak in-memory(bellek içi) özelliğe sahiptir. Yığın veri işlemede kullanılan Apache Hadoop teknolojisine göre 100 kat daha hızlıdır. Bu süre farkının asıl nedeni geröekleştirilen işlemlerde hard diski kullanan Apache Hadoop'tan farklı olarak Apache Spark'ın veri işlemede sistem hafızasını (RAM) kullanmasından ve az maliyetli veri dönüşümleri (shuffling) gerçekleştirmesinden dolayıdır (Sakr, 2017).

Apache Spark, bu çalışmada kullandığımız Python dili başta olmak üzere, Scala ve Java dilleri için API desteği sunmaktadır. Ayrıca Apache Spark, yine bu çalışmada kullandığımız NoSQL veritabanı olan Apache Cassandra[2] ile HDFS ve Amazon S3 gibi dağıtık sistemlerden veriyi okuyabilmektedir. Hadoop YARN ve Apache Mesos gibi kaynak yönetim ürünlerini de desteklemesinin yanında kendi kaynak yöneticisine de sahiptir. Spark SQL adında kendine özgü sorgulama dili vardır. Spark SQL ile birlikte DataFrames(veri çerçevesi) adı verilen yeni bir veri saklama yöntemi geliştirilmiştir. (Databricks, n.d.) DataFrame'lerde sorgulama, diğer SQL dillerindeki sorgulama yapısına benzer çalışmaktadır. Apache Spark, kendi sorgulama dili dışında MySQL, Oracle ve MSSQL gibi ilişkisel veritabanlarını da desteklemektedir. Apache Spark ile akan veri üzerinde işlem yapabilmek için Apache Spark Streaming teknolojisi geliştirilmiştir.

2.1.2 Apache Spark Streaming

Apache Spark ile yapılan tüm işlemlerin akan veri işlemlerine de uyarlanabilmesi için Apache Spark Streaming geliştirilmiştir. Apache Spark Streaming, akan veri işlemenin en önemli gereksinimlerinden olan bellek kullanımı ve süre sorunlarına çözüm sunmaktadır (Damji, Wenig, Das, & Lee, 2020). Spark, oldukça geniş bir geliştirici topluluğu ve büyük bir kullanıcı grubuna sahip olmasından dolayı Apache'nin diğer teknolojileri olan Storm ve Samza gibi arızalara karşı dirençli hale getirilmiştir. Ayrıca kullanılacak kaynakların dinamik olarak ölçeklenmesini destekler. Apache'nin diğer teknolojilerinden Storm, Samza

[2] Apache Spark, https://spark.apache.org/

Apache Cassandra, https://cassandra.apache.org/_/index.html

ve Trident'in Spark ile karşılaştırılması EK'te verilmiştir. (Wingerath, Gessert, Friedrich, & Ritter, 2016)

- **Storm**[3] **:** Ücretsiz ve açık kaynaklı dağıtılmış gerçek zamanlı bir hesaplama sistemidir. Apache Hadoop'un yığın verisi için yaptığı işlemi akan veri için yapar.

- **Samza**[4] **:** LinkedIn tarafından geliştirilmiş açık kaynak kodlu bit teknolojidir. Hem akan veri hem de yığın veri işlemek için tek bir kod yazılmasının yeterli olmasını sağlar.

- **Trident**[5] **:** Apache Storm ile birlikte gerçek zamanlı veri analizi için kullanılır. Yüksek aktarım hızı ve düşük gecikme sağlar.

2.2 Yığın veri işleme

Akan veri işleme teknolojisi yaygınlaşmaya başlamadan önce Yığın Verisinden oluşan veriler üzerinde analiz işlemleri yaygın olarak kullanılmaktaydı. Yığın Veri, rastgele yapılar ve büyük miktarlardaki verilerden oluşan bir veri yapısıdır. Yığın verisini işleyebilmek için kullanılan en önemli yöntemlerden biri MapReduce (MR) yöntemidir (Dean & Ghemawat, 2008). MR, üzerinde işlem yapılacak büyük miktardaki veriyi küçük işlenebilir verilere dağıtma(map) ve işlem gerçekleştirildikten sonra sonuçların birleştirilmesi(reduce) prensibine dayanmaktadır. Yığın veri işlemede MR teknolojisiyle kullanılan diller arasında Jaql, Hive ve Pig bulunmaktadır. (Neves & Bernardino, 2015)

- **Jaql**[6] **:** IBM tarafından geliştirilmiştir. JSON veri yapısı kullanır. JSON haricinde CSV, TSV, XML gibi farklı veri kaynaklarını da destekler.

[3] https://storm.apache.org

[4] https://samza.apache.org

[5] https://storm.apache.org/releases/current/Trident-tutorial.html

[6] https://code.google.com/archive/p/jaql/

- **HiveQL**[7] **:** Facebook tarafından geliştirilmiştir. Complex(karmaşık) veri yapısı kullanır. Apache Hadoop ile kolay entegre olması sorgulama ve analiz etmeyi kolaylaştırır.
- **Pig**[8] **:** Yahoo! Tarafından geliştirilmiştir. Karmaşık, Nested(iç içe) veri yapılarından kullanır. Pig Latin adlı dili kullanır. Programlama kolaylığı, optimizasyon seçenekleri ve genişletilebilirlik sunar.

Yığın veri işleme, her ne kadar oldukça popüler bir alan olsa da, afet yönetimi gibi milisaniyeler kadar kısa zamanda ve anlık sonuç beklenen durumlar için çok uygun olmamaktadır. (Ranjitha, 2015)

Akan ve Yığın Veri işlemenin hangisinin kullanılacağının doğru seçilmesi veri kaybının önlenmesi, işlem süresinin uzamaması ve doğru analiz sonuçlarını elde edebilmek açısından oldukça önemlidir. Çalışma yapılacak verinin yapısını doğru bir şekilde anlayıp bu veri için hangi teknolojilerin kullanılacağının doğru seçilmesi, çalışma sonuçlarının doğru ve güvenilir olabilmesini sağlar. Bu bağlamda, akan veri ve yığın verisi arasındaki farkları iyi anlamak gerekmektedir. Bu bağlamda, akan ve yığın veri karşılaştırmalara bir örnek EK'te verilmiştir. (Doguc & Aydin, 2019)

2.3 Veri Depolama Konsepti ve Teknoljileri

Akan veri işlemede veriler farklı kaynaklardan anlık olarak ve hızlı bir şekilde toplanmaktadır. Toplanan veriler sürekli olarak yerini anlık olarak değişen yeni verilere bırakmaktadır. Akan veri işleme sürecinde anlık toplanan verilerin, oldukça kısa süreler içerisinde işlenirken bir yandan veri bütünlüğünün korunması, yeni veriler ile karışmaması hem mevcut işlenen verilerde hem de yeni toplanan verilerde kayıp olmaması elzemdir. Verilerin istenilen şekilde korunabilmesi için uygun depolama teknolojilerinin seçilebilmesi çok önemlidir. Verileri depolama yapılarından dolayı yığın verilerinde genellikle SQL veritabanları kullanılırken, akan verilerde NoSQL veritabanları kullanılmaktadır. Kullanılan

[7] https://hive.apache.org

[8] https://pig.apache.org

veriler için doğru veritabanı seçerken kullanılan teoremler içerisinde en popülerlerinden biri CAP Teoremi'dir.

2.3.1 CAP Teoremi

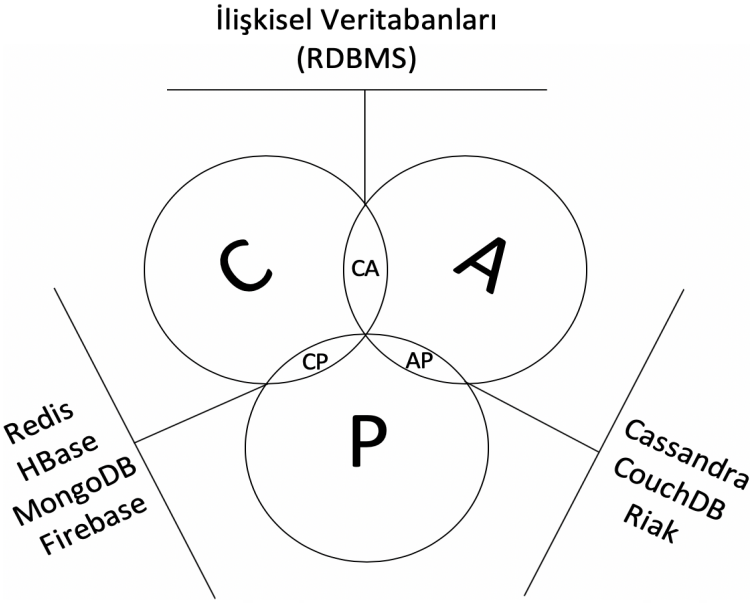
CAP Teoremi, şu anda Google'da Altyapı Başkan Yardımcısı olan Eric BREWER tarafından ortaya atılan, başlarda kanıtlanmış bir gerçekten ziyade sezgisel bir bir bulgu olmasına rağmen oldukça büyük bir etkiye sahip olduğu için birçok araştırmacının ilgisini çekmeyi başarmış ve bu araştırmaların sonucunda, ortaya atılmasından 2 yıl sonra resmi olarak kanıtlanmış bir teoremdir. BREWER 2012 yılında bir makale (Brewer, 2012) yayınlayarak CAP teoreminin en belirgin düzeltmelerini yapmıştır. CAP teoreminin temelinin üç temel ilke oluşturmaktadır:

1. **Consistency (Tutarlılık) :** Verinin dağıtıldığı bütün düğümlerde aynı zaman dilimi içerisinde aynı verinin olmasıdır. Veride tutarlılığın sağlanmasıdır.

2. **Availability (Erişilebilirlik)** : İstenilen veriye erişimi ifade etmektedir. Birden fazla düğümden oluşan dağıtık veritabanındaki bazı düğümlerin kullanımda olmaması durumunda bile, isteklerin cevapsız kalmamasına imkan sağlamaktadır.

3. **Partition Tolerance (Bölümleme Töleransı) :** Verinin düğümler arasında düzgün bir şekilde dağıtılması sayesinde, düğümlerden bazıları zarar görse bile sistemin çalışmaya devam edebilmesidir.

CAP teoremine göre bir sistem, yukarıda bahsedilen 3 ilkeden yalnızca 2 tanesini aynı anda sağlayabilmektedir. Dolayısıyla, bir veritabanı sistemi aynı anda CA, AP veya CP olabilir. NoSQL Veritabanı teknolojilerinin CAP teoremi baz alınarak sınıflandırması Şekil 1'de verilmiştir. (Doguc & Aydin, 2019)



Şekil 1. CAP ile Veritabanlarının Sınıflandırılması

2.3.2 İlişkisel Veritabanı Yönetim Sistemleri (RDBMS)

(Codd, 1970)İlişkisel veritabanı terimi 1970 yılında Edgar Frank Codd tarafından önerilmiştir. Organizasyonunda ilişkisel veri modelini kullanır. Bu bölümde ilişkisel veritabanlarına örnekler verilmiş ve ilişkisel veritabanlarının kullandığı SQL dili anlatılmıştır.

- **PostgreSQL[9] :** Postgres olarak da bilinen, 30 yılı aşkın aktif geliştirmesine sahip, açık kaynak kodlu bir veritabanıdır. Ayrıca, PostgreSQL oldukça genişletilebilir bir yapıya sahiptir. Örneğin, veritabanının yeniden derlenmesine gerek kalmadan farklı dillerde kodlar yazılmasına olanak tanır. (The PostgreSQL Global Development Group, 2021)

- **MySQL [10] :** İlk olarak MySQL AB adlı şirket tarafından 1994'te geliştirilmeye başlanmış, günümüzde ise ORACLE'a ait olan veritabanıdır. Ücretsiz platformları

9 https://www.postgresql.org

10 https://www.mysql.com

mevcut olmakla birlikte ücretli lisans seçenekleri de bulunmaktadır. (Axmark & Widenius, 2021)

- **IBM DB2**[11] **:** IBM tarafından geliştirilmiş hem şirket içinde hem de bulut ortamında kullanılabilen bir veritabanıdır. Yapay zekâ desteğine sahip olmasından dolayı iş yükünü hafifletici özelliğe sahiptir. Bu özelliğinden dolayı düşük maliyet de sağlamaktadır. (Mihaylov et al., 2021)

2.3.3 SQL (Structured Query Language)

Verilerin doğru kullanımı ve depolanması depolanması amacıyla veritabanları üzerinde çalışmalar artıkça, bu veritabanlarında kullanılacak bir sorgu diline ihtiyaç duyulmuştur. Bu ihtiyaç doğrultusunda matematiksel söz dizimine sahip SQUARE adı verilen bir sorgulama dili geliştirilmiştir. Bu dilin kullanıcı kitlesi genişlemeye başladıkça daha kolay ve anlaşılır olabilmesi için matematiksel söz dizimi yerine İngilizce'ye benzer bir söz dizimine sahip olan SEQUEL dili geliştirilmiştir. Zaman içerisinde SEQUEL kelimesi İngilizde telaffuzuna paralel olarak SQL adını almıştır. Yukarıda belirtilen veritabanları haricinde, Microsoft SQL Server ve Oracle gibi veritabanları SQL veritabanlarına örnektir. SQL veritabanları ACID ilkeleri üzerine çalışmaktadır. (A & Surendran, 2012)

Atomicity (Bütünlük) : Yapılacak işlemler veritabanı tarafından bir bütün olarak ele alınmalıdır. İşlemlerden herhangi birinin kesintiye uğraması durumunda bütün işlemler geçerliliğini yitirir.

Consistency (Tutarlılık) : Yapılan işlemler, örneğin bankacılık işlemlerindeki para çekme-yatırma senaryosu, birbiriyle tutarlı olmalıdır.

Isolation (Yalıtım) : Bir veri üzerinde aynı anda meydana gelen değişim isteklerinin birbirinden etkilenmemesi için işlemler seri sırada yapılır. Bir işleme başlanmak için diğerinin bitmesi beklenir.

11 https://www.ibm.com/analytics/db2

Durability (Dayanıklılık) : Veri üzerinde yapılan işlemler esnasında bir hata veya veri kaybı meydana gelirse, sistemin kendisini daha önceki geçerli duruma getirebilmesidir.

2.3.4 NoSQL Veri Depolama Sistemleri

NoSQL Veritabanları, mevcut SQL veritabanlarının akan veri işlemedeki yetersizliği ve oluşturduğu sorunlardan dolayı, veriyi birden fazla düğüm (node) içerisinde istenilen sayıda kopyasının oluşturulup dağıtılması ve birden fazla kopyası bulunabilen veriye hızlı bir biçimde erişimi gerçekleştirebilen BigTable ve DynamoDB'den ilham alınarak ortaya çıkarılmıştır. NoSQL Veritabanları 4 kategoride incelenmektedir. Bunlar: döküman tabanlı (document store), anahtar-değer(key-value), kolon tabanlı (column store), ve grafik(graph oriented) tabanlılardır. NoSQL veritabanları BASE paradigması tabanlıdır ve SQL veritabanlarındaki ACID kısıtlamarını içermez. (Lourenço, Cabral, Carreiro, Vieira, & Bernardino, 2015) BASE paradigması 3 temel maddeden oluşmaktadır:

- **Basically Available**: Sürekli erişilebilme
- **Soft State:** Verinin tutarlılığının öncelikli olarak zorunlu olmaması
- **Eventual Consistency**: Nihai olarak verinin tutarlılığının korunması

Akan veri işlemede kullanılan populer NoSQL veri depolama teknolojilerinden Cassandra, MongoDB, Couchbase, DynamoDB ve Firebase' in karşılaştırılması EK'te verilmiştir. (Doguc & Aydin, 2019)

3. LİTERATÜR TARAMASI

Bu bölümde gerçekleştirilen tez çalıması ile ilgili olan yayınlara yer verilmiştir.

Jambi ve Anderson yaptıkları çalışmada (Jambi & Anderson, 2017), çeşitli sebeplerden dolayı, akan veri analizi gerçekleştiren araçlara duyulan talepten bahsedilmiştir. Yaptıkları çalışmada veriler üzerinde yapılan sorguların kolaylaştırılması ve bu sorgulara esneklik kazandırılmasına odaklanılmıştır. Yazarlar, sorgulara esneklik kazandırılması amacıyla geliştirdikleri "EPIC REAL TIME" platformunun bir prototip uygulamasını anlatmışlardır. Geliştirilen bu prototipin performans, kullanılabilirlik, ölçeklenebilirlik ve güvenilirlik gibi alanlarda ve gerçek zamanlı toplu veriler üzerinde gösterdiği performans anlatılmıştır.

Aydin ve Anderson tarafından geliştirilen IDCAP (Aydin & Anderson, 2017), akan veri toplama ve analitiğinin, yazılım sistemleri için öneminden ve bu alandaki çalışmaların günümüz şartlarındaki gereklilğinden bahsedilmiştir. Akan veri analizini etkili ve doğru bir şekilde yapabilmek için kullanılması gerekli olan yazılım teknolojileri anlatılmıştır. Bu teknolojiler içerisinden IDCAP Platformu detaylı bir şekilde açıklanmıştır. Çalışmada akan veri analitiği için geliştirilen bir başka platform olan EPIC platformundan bahsedilmiş, IDCAP'ın EPIC'ten farkları anlatılmıştır. Her ne kadar çalışma içerisinde Twitter verilerine odaklanılmış olsa da IDCAP'ın her türlü alanda kullanılabileceği belirtilmiştir. IDCAP platformunun özellikleri şu şekilde listelenmiştir: sağlam, güvenilir, hata töleranslı, tam zamanlı kullanılabilen ve kullanıcı dostu arayüze sahip bir sistem olduğunun vurgusu yapılmıştır.

Marungo'nun yapmış olduğu çalışmada (Marungo, 2018), CAP teoremi, ACID ve BASE ilkeleri detaylı bir şekilede anlatılmıştır. NoSQL veritabanları ve karakteristik özellikleri ele alınmıştır. NoSQL veritabanlarının en popülerlerinden olan Apache Cassandra ve MongoDB ayrıntılı bir şekilde anlatılmış ve farklı durumlarda farklı veri tipleri üzerindeki performansları, hata töleransları CAP teoremi ve ACID-BASE ilkeleri baz alınarak incelenip kıyaslanmıştır.

Han ve arkadaşlarının yapmış olduğu çalışmada (Han, John, & Zhan, 2018), büyük veri sistemlerinin hızlı bir şekilde gelişmesinden dolayı bu sistemlerin kıyaslanması için yapılan çalışmalardan bahsedilmiştir. Adı geçen çalışmada, büyük veri sistemleri ile alakalı bazı temel kavramlar açıklanmış ve bu sistemlerin farklı açılardan kıyaslamaları yapılmıştır. Çalışma aynı zamanda büyük veri sistemlerinin güvenilirlik, kullanılabilirlik ve objektif

kıyaslamanın yapılabilmesi için dikkat edilmesi gereken hususlar hakkında tartışmalar sunmaktadır.

Medical IoT (Iot, n.d.) çalışmasında da gelişen teknoloji ile birlikte akan veri analitiğinin geleneksel veri analitiğine göre popülerleşmesinden bahsedilmektedir. Gelişen 5G teknolojisinin endüstiriyel, medikal, IoT, AR, VR, oyunlar ve daha birçok alanda sağlayacağı gerçek zamanlı verilerin, geleneksel veri analitiği yöntemlerine getireceği büyük yük anlatılmış ve akan veri işleyebilmenin öneminden bahsedilmiştir. Akan veri işlemenin faydaları şu şekilde listelenmiştir: düşük, öngörülebilir gecikme, daha basit geliştirme ve test süreci, daha düşük donanım ihtiyacı, hata dayanıklılığı ve bahsedilen diğer faydaların bir sonucu olarak daha düşük maliyet.

Taiwo ve arkadaşlarının yaptığı çalışmada (Kolajo, Daramola, & Adebiyi, 2019), son zamanlarda artan uygulamaların büyük veriyi hızlı bir şekilde beslediğinden ve bu sebeple büyük veri akışlarının her alanda bulunur hale gelmesinden bahsedilmiştir. Çalışmada, IEEE, ACM, SpringerLink ve Elsevier gibi kuruluşlarda yayınlanan çalışmaları indekleyen 3 ana veritabanı olan Scopus, ScienceDirect ve EBSCO'nun veri kaynakları baz alınmış. Çalışmada büyük verinin gerçek zamanlı analiz edilmesine yönelik önemli çalışmalara rağmen, bu çalışmaların ön işleme aşamasına çok fazla dikkat edilmediği belirtilmiştir. Araştırmaların verilerin sürekli büyüyen boyutu ve karmaşıklığıyla başa çıkabilmek için ölçeklenebilir framework(çerçeve) ve algoritmalar üzerine yoğunlaşması gerektiği önerilmiştir.

Gehring ve arkadaşlarının yaptığı çalışmada (Gehring, Charfuelan, & Markl, 2019), veri işleme sistemlerinin sayısındaki artışa değinilmiştir. Adı geçen çalışmada, bu sistemler çeşitli alanlarda değerlendirilmiş ve en doğru olanın seçilmesi konusunda kıyaslamalar yapılmıştır. Gehring ve arkadaşları tarafından Apache'nin Storm, Trident, Spark, Flink ve Samza teknolojileri detaylı bir şekilde incelenmiş, benzer ve farklı yönlerini anlatılmıştır. Bizim çalışmamızdan farklı olarak, Twitter verileri yerine borsa verileri ile analiz gerçekleştirilmiştir.

Aswaty ve arkadaşlarının çalışmasında (Aswathy, Prabha, Gopal, Pullarkatt, & Ramesh, 2022), Dünya'da meydana gelen afetler ve bu afetlerin yol açtığı kayıplardan yola çıka çıkarak doğru ve hızlı veri analitiğinin önemini anlatmaktadır. Adı geçen çalışmada sel, heyelan ve yağmur gibi doğal afetler ile ilgili Twitter'dan toplanan odak tweetlerin analiz edilip, gereksiz verilerin ayıklanarak analiz edilmesi gösterilmiştir. Toplanan veriler üzerinde analiz işlemi gerçekleştirilirken medya ve konum bilgisi gibi sosyal medya gizlilik

politikalarının ihlal edilmediği belirtilmiştir. Ayrıca bu çalışma, bizim çalışmamızdan farklı olarak Hint dillerindeki tweetleri toplayıp bu verileri İngilizce'ye çevirmektedir.

4. AKAN VERİ ANALİZİNDE KULLANILAN VERİ DEPOLAMA TEKNOLOJİLERİ

Bu bölümde akan veri analizinde kullanılan populer veri depolama teknolojileri ve teknik özellikleri anlatılacaktır. Bu bölümde sırasıyla Riak, CouchDB, Redis, HBase, MongoDB ve FireBase anlatılmıştır. Ayrıca Apache Cassandra teknolojisi akan veri işlemede kullanılmaktadır. Apache Cassandra 5.1.1 de anlatıldığı için bu bölüme tekrar eklenmemiştir.

4.1 Riak[12]

2009 yılında Basho Teknolojileri tarafından Erlang dili ile yazılan, açık kaynak kodlu bir anahtar-değer(key-value) veritabanı teknolojisidir. CAP teoremindeki kullanılabilirlik ve bölümleme töleransı ilkelerinin özelliklerine sahiptir. Riak S2 başta olmak üzere, Redis ve Apache Spark gibi teknolojilerle entegre olarak veri depolamayı optimize eder. (Doguc & Aydin, 2019; Rmis & Topcu, 2020)

4.2 CouchDB[13]

2005 yılında ortaya çıkmasına rağmen 2008 yılında Apache tarafından geliştirilmeye devam edilmiştir. Riak'a benzer şekilde CAP teoreminin AP ilkelerini karşılar ancak; Riak'tan farklı olarak belge(document) tabanlı bir veritabanı teknolojisidir. MapReduce ile birlikte çalışarak verilerin geri alımında kullanılabilir. Ayrıca geliştirici dostu bir sorgulama diline sahiptir. GrubHub, Cleverspeck ve IBM gibi şirketler tarafından kullanılmaktadır. (Doguc & Aydin, 2019; Razoqi, 2021)

4.3 Redis[14]

2009 yılında Salvatore SANFLIPPO tarafından ANSI C dilinde geliştirilmiştir. CAP teoreminin tutarlılık ve bölümleme töleransı olan CP ilkelerininin özelliklerine sahiptir.

12 https://riak.com

13 https://couchdb.apache.org

14 https://redis.io

Bellek içi(in-memory) çalıştığı için esneklik ve hız sağlar. Her ne kadar bellek içi özelliğe sahip olsa da istenildiği zaman disk üzerine de vei kaydederek çalışabilir. Stackoverflow, Twitter, Medium ve Alibaba şirketler tarafından kullanılmaktadır. (Aydin & Anderson, 2017; Doguc & Aydin, 2019)

4.4 HBase[15]

2007 yılında, Powerset firması tarafından Java dili ile yazılmıştır. Kolon tabanlı bir veritabanı teknolojisidir. Cap teoremindeki tutarlılık ve bölümleme töleransına karşılık gelen CP ilkelerinin özelliklerine sahiptir. Çapraz platformları destekleyen dağıtık bir yapısı vardır. (Mukherjee, Mondal, Chaki, & Khatua, 2019) çalışmasında HBase'in anahtar deposunun sağladığı zaman avantajına değinilmiştir. Alibaba, Pinterest, Xiaomi ve Yahoo! Tarafından kullanılmaktadır. (Doguc & Aydin, 2019)

4.5 MongoDB[16]

2007 yılında platform as a service (servis platformu) olarak geliştirilmiş, belge tabanlı bir veritabanıdır. Python, C++, JavaScript ve Go ile yazılmıştır. CAP teoreminin tutarlılık ve bölümleme töleransı (CP) ilkelerini karşılar. Zengin sorgulama dili, kolay kullanım, çoklu depolama motoru desteği ve yüksek performans sağladığı avantajlar arasındadır. MIT, Uber ve Foursquare gibi firmalar tarafından kullanılmaktadır. (Cattell, 2011; Chauhan, 2017; Doguc & Aydin, 2019; Eken, Kaya, Sayar, & Kavak, 2014; Lourenço et al., 2015)

4.6 Firebase[17]

Sahipleri James TAPLIN ve Andrew LEE olan, 2011 yılında kurulmuş Envolve şirketine ait bir teknoloji olmasına rağmen, daha sonra Google tarafından satın alınmış bir veritabanı teknolojisidir. CAP teoreminin tutarlılık ve bölümleme töleransı olan CP ilkelerinin özelliklerini taşımaktadır. Gerçek zamanlı verileri depolarken aynı zamanda bu

15 https://hbase.apache.org

16 https://www.mongodb.com

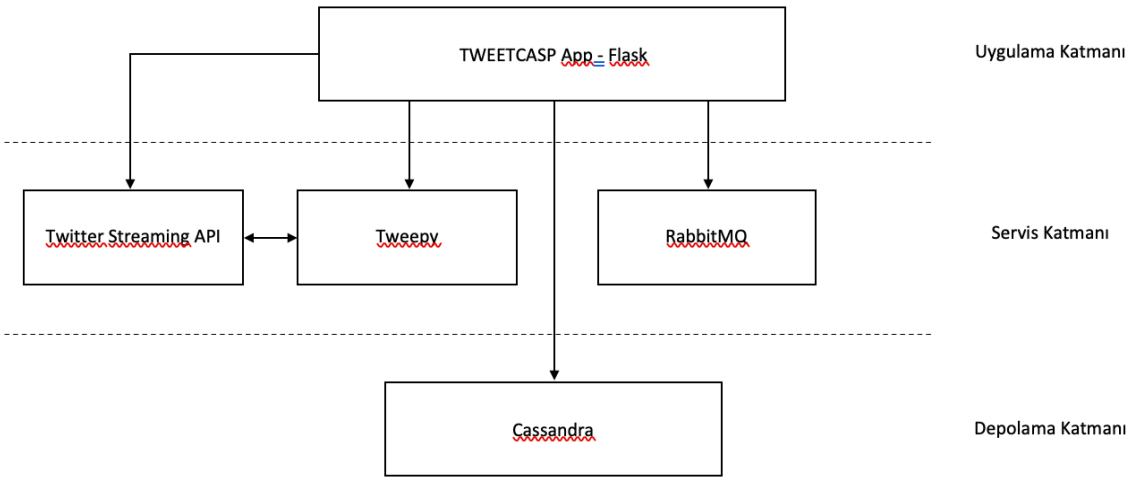
17 https://firebase.google.com

verilerin senkronizasyonunu sağlar. Admob, Google Bulut Platformu ve Google Ads. Gibi Google'ın diğer teknolojileri ile bütünleşmiş çalışarak geniş bir kullanım alanı sağlar. Trendyol, Twitch, Square gibi firmalar tarafından kullanılmaktadır. (Doguc & Aydin, 2019)

5. GERÇEK ZAMANLI VERİ ANALİZİ PLATFROM TASARIMI

Bu bölümde gerçek zamanlı veri analizi gerçekleştirmek için geliştirilen platformun mimarisi, kullanılan teknolojiler ve geliştirilen web uygulamasından bahsedilecektir.

5.1 Sistem Mimarisi ve Kullanılan Teknolojiler



Şekil 2. Sistem Mimarisi

5.1.1 Cassandra

Cassandra, açık kaynak kodlu dağıtık bir veritabanıdır. CAP teoreminin AP (Erişilebilirlik ve Bölümleme Töleransı) ilkelerini karşılar. Java dilinde yazılan Casandra, Facebook tarafından 2008 yılında geliştirilmeye başlanmıştır. Amazon'un Dynamo ve Google'ın BigTable veritabanlarına benzerlik gösterir. CQL adı verilen kendi sorgulama dili vardır. Tek bir hata noktası prensibi (No Single Point of Failure), verinin kaybolmadan saklanabilmesi, ölçeklendirilebilirlik, yüksek okuma-yazma hızı ve yüksek veri sıkıştırma oranı sayesinde avantaj sağlar. Günümüzde Apple, Netflix, CERN, eBay, GitHub, Instagram gibi 1500'den fazla şirket tarafından kulanılmaktadır. (Doguc & Aydin, 2019)

Bu tez çalışmasında Cassandra kalıcı veritabanı (permanent database) olarak kullanılmıştır. Twitter'dan stream edilen tweetler RabbitMQ'ya geçici olarak kaydedilip, tekrar alındıktan sonra tweet verisi içerisinden istenilen parametrelere ait veriler Cassandra'ya kaydedilmiştir. Böylece verilerin kaybolmaması amaçlanmıştır. Bu çalışmada kullanılan Cassandra tablosu EK'te verilmiştir.

```
[Beril-MacBook-Pro:dse beril$ bin/dse cassandra
Server will start with the original workload configuration. Edit the dse script
to change that (the script that printed this message).
Search: on
Graph: on
Spark: on
Tomcat: Logging to /Users/beril/dse/logs/tomcat
```

Şekil 3. Terminal Cassandra Komutu

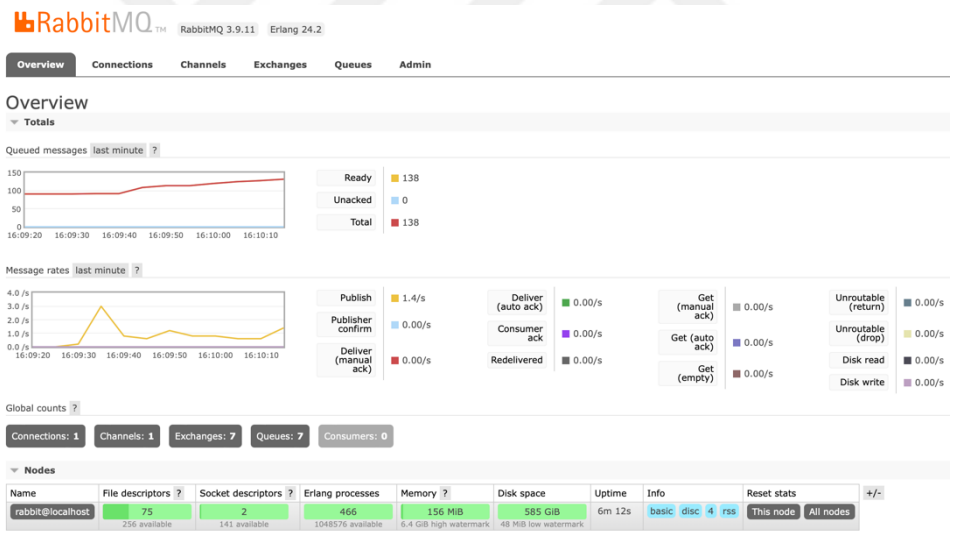
Şekil 4. Cassandra Tablo Örneği

5.1.2 RabbitMQ

RabbitMQ[18], farklı uygulamaların ortak bir protokol aracılığıyla veri paylaşmasına izin vermek veya işleri dağıtılmış işçiler tarafından işlenmek üzere basitçe sıraya koymak için kullanılabilen açık kaynaklı bir mesaj komisyoncusu ve kuyruk sunucusudur. STOMP ve AMQP başta olmak üzere birçok mesajlaşma protokolünü destekler. (Rostanski, Grochla, & Seman, 2014) Bu tez çalışmasında verileri işleme hızı, Cassandra ile rahat çalışıp entegre olabilmesi ve sağladığı web arayüzünden dolayı RabbitMQ kullanılmıştır.

```
[Beril-MacBook-Pro:~ beril$ brew services start rabbitmq
==> Successfully started `rabbitmq` (label: homebrew.mxcl.rabbitmq)
[Beril-MacBook-Pro:~ beril$ brew services stop rabbitmq
Stopping `rabbitmq`... (might take a while)
==> Successfully stopped `rabbitmq` (label: homebrew.mxcl.rabbitmq)
```

Şekil 5. Terminal RabbitMQ Komutları



Şekil 6. RabbitMQ Web Arayüzü

5.1.3 Tweepy

Tweepy[19], Twitter'ın Streaming API'larına erişim sağlamak için kullanılan bir Python kütüphanesidir. Kullanımı ve sistem kurulumu kolay, kullanıcı dostu bir teknolojidir.

18 https://www.rabbitmq.com

19

Tweepy'ı kullacak olanlara oldukça kullanışlı ve anlaşılır bir dökümantasyon sunmaktadır. Twitter API'sine kolay bağlanabilmesi, dökümantasyonuna erişimin rahat olması, Cassandra ve RabbitMQ'nun ortak desteklediği python sürümünde çalışabilmesinden dolayı bu tez çalışmasında Tweepy kullanılmıştır.

5.1.4 Flask

Flask[20], web uygulamarı geliştirmeye yarayan bir Python modülüdür. Küçük ve genişletilmesi kolay bir çekirdeğe sahiptir. URL yönlendirme, şablon motoru gibi birçok özelliğe sahiptir. Pek fazla araç ve kitaplık içermemesinden dolayı daha taşınabilir bir yapıya sahiptir. KNN uygulamaları Flask'ın kullanıldığı alanlara örnektir. (Anggoro & Aziz, 2021) Bu tez çalışmasında, web arayüzü oluştururken sağladığı rahatlık ve kolay kullanımdan dolayı Flask kullanılmıştır. Flask ile kullanıcıların Twitter'dan çekmek istedikleri veri ile ilgili kelime belirtebilecekleri ve veri çekme işlemini başlatıp, belirttikleri diğer kelime ile istedikleri zaman yüzdelik analiz elde edebilecekleri bir platform oluşturulmuştur.

https://www.tweepy.org

20 https://flask.palletsprojects.com/en/2.1.x/

6. GELİŞTİRİLEN PLATFROMUN KULLANILMASI

Bu bölümde geliştirilen platformun kullanılması ile alakalı kısımlar anlatılacaktır.

6.1 Twitter API Bağlantısı

Geliştirilen bu platformda öncelikle Tweepy ile Twitter API bağlantısı gerçekleştirilmiştir. Twitter tarafından bize sağlanan Consumer Key(API Key), Consumer Secret(API Secret) ve Access Token bilgilerini kullanarak Tweepy ile bir yetkilendirme sağladık. Böylelikle istediğimiz anda twetleri çekmeye hale geldik.

```
def main(text, isStart, searchText):
    ckey = "***"
    csecret = "***"
    atoken = "***"
    asecret = "***"

    auth = OAuthHandler(ckey, csecret)
    auth.set_access_token(atoken, asecret)

    twitterStream = Stream(auth, listener())

    if isStart:
        startStreaming(twitterStream, text)
```

Şekil 7. Twitter Stream API Bağlantısı

6.2 Twitter-RabbitMQ Bağlantısı

Çalışmanın bir sonraki aşamasında, Twitter'dan anlık çekilen tweet verilerinin geçici veritabanımız olan RabbitMQ'ya hızlı bir şekilde yazılmasını sağladık. Bu işlem esnasında hız kaybetmemek için herhangi bir analiz işlemi gerçekleştirmedik, alınan tweet verilerini

değişiklik veya bekleme olmadan RabbitMQ'ya gönderdik. Bu sayede vakit kaybı ve tweetlerin API servisten çekilmesi esnasında aksaklik olmamasını amaçladık.

```
def sendJSONToRabbitMQ(queueName, sendingJson):
    print(queueName)
    print(sendingJson)
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()
    channel.queue_declare(queue=queueName)
    channel.basic_publish(exchange='',
                          routing_key=queueName,
                          body=json.dumps(sendingJson))
    return queueName

queueName2 = None
jsonArray = []
allJsons = []
class listener(StreamListener):
    def on_data(self, data):
        global queueName2
        global jsonArray

        myData = json.loads(data)

        print(myData["text"])
        if myData["geo"] is None:
            print(myData["geo"])

        jsonArray.append(myData)
        allJsons.append(myData)

        if len(jsonArray) == 10:
            queueName2 = sendJSONToRabbitMQ('TestQue', jsonArray)
            jsonArray = []

        print ("queueName", queueName2)
        return(True)

    def on_error(self, status):
        print(status)


def startStreaming(twitterStream, text):
    twitterStream.filter(track=[text], is_async=True)
```

Şekil 8. RabbitMQ'ya Veri Yazma İşlemi

6.3 RabbitMQ-Cassandra Bağlantısı

Bu aşamada RabbitMQ'ya daha önceden yazdığımız verilerin okunup Cassandra'ya yazılması işlemini gerçekleştirdik. Verilerin Cassandra'ya yazılması esnasında daha önce olduğu gibi verilerin mevcut haliyle yazma işlemi gerçekleştirmedik, elde edilen veriler

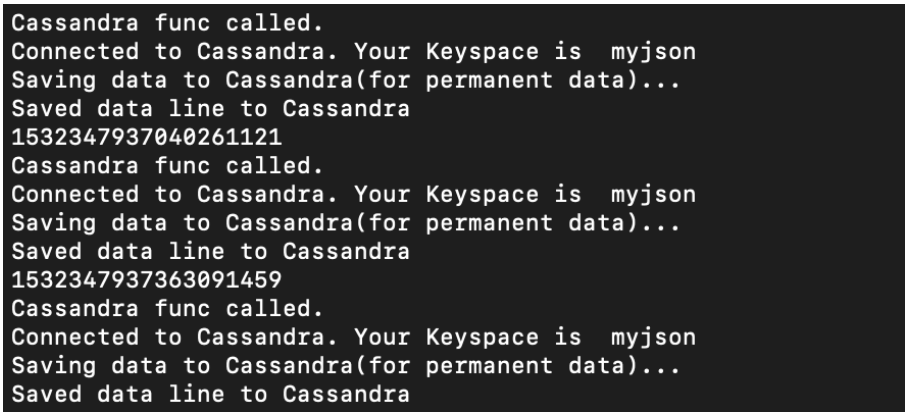
üzerinde analiz yaparak veri içerisinden sadece id, tarih, hesap bilgisi, tweet içeriği ve lokasyon bilgisini Cassandra'ya kaydettik. Bu sayede veri boyutunu düşürmeyi amaçladık. Cassandra veritabanının bir örneği EK'te verilmiştir.

```
def connectCassandra(id, date, account, tweet, location):
    print("Cassandra func called.")
    cluster = Cluster()
    session = cluster.connect('myjson')

    print("Connected to Cassandra. Your Keyspace is ", session.keyspace)
    print("Saving data to Cassandra(for permanent data)...")

    session.execute(
        """
        INSERT INTO tweetdatas (id, date, account, tweet, location)
        VALUES (%s, %s, %s, %s, %s)
        """,
        (id, date, account, tweet, location)
    )

    print("Saved data line to Cassandra")

def main():
    connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
    channel = connection.channel()

    queueName = 'TestQue'

    channel.queue_declare(queue=queueName)

    def callback(ch, method, properties, body):
        print(" [x] Received %r" % body)
        a = json.loads(body)
        for j in a:
            print(j["id_str"])
            connectCassandra(j["id_str"], j["created_at"], j["user"]["name"], j["text"], "No Location")

    channel.basic_consume(queue=queueName, on_message_callback=callback, auto_ack=True)

    print(' [*] Waiting for messages. To exit press CTRL+C')
    channel.start_consuming()
```

Şekil 9. Cassandra Bağlantı ve Kayıt Kodları

```
[Beril-MacBook-Pro:streamingTwitter beril$ python3 RabbitMQToCassandra.py
 [*] Waiting for messages. To exit press CTRL+C
```

Şekil 10. RabbitMQ'dan Cassandra'ya Veri Aktarımı



Şekil 11. Cassandra Veri Kaydedilmesi

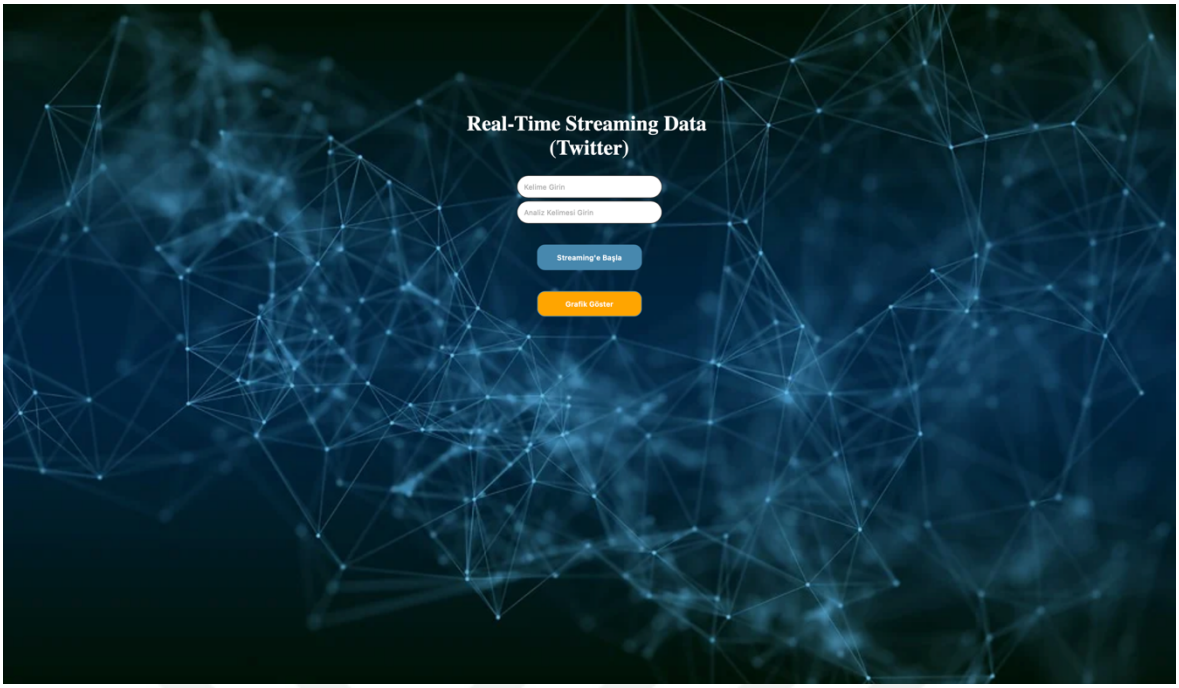
6.4 WEB Arayüzü

Çalışmanın bu kısmında Flask ile bir web arayüzü oluşturuldu. Bu arayüze sistemin çalışma prensibine uygun olarak kullanıcıların anlık tweetler içerisinde bulmak istedikleri kelimeyi girebilecekleri bir input alanı, analiz etmek için kelime belirtebilecekleri 2. bir input alanı, girilen kelimelere göre sistemin Twitter'dan verileri çekmeye başlamasını sağlayan ve çekilen verileri içerisinde gerekli analizi gerçekleştiren 2 adet button eklenmiştir. Bu sayede

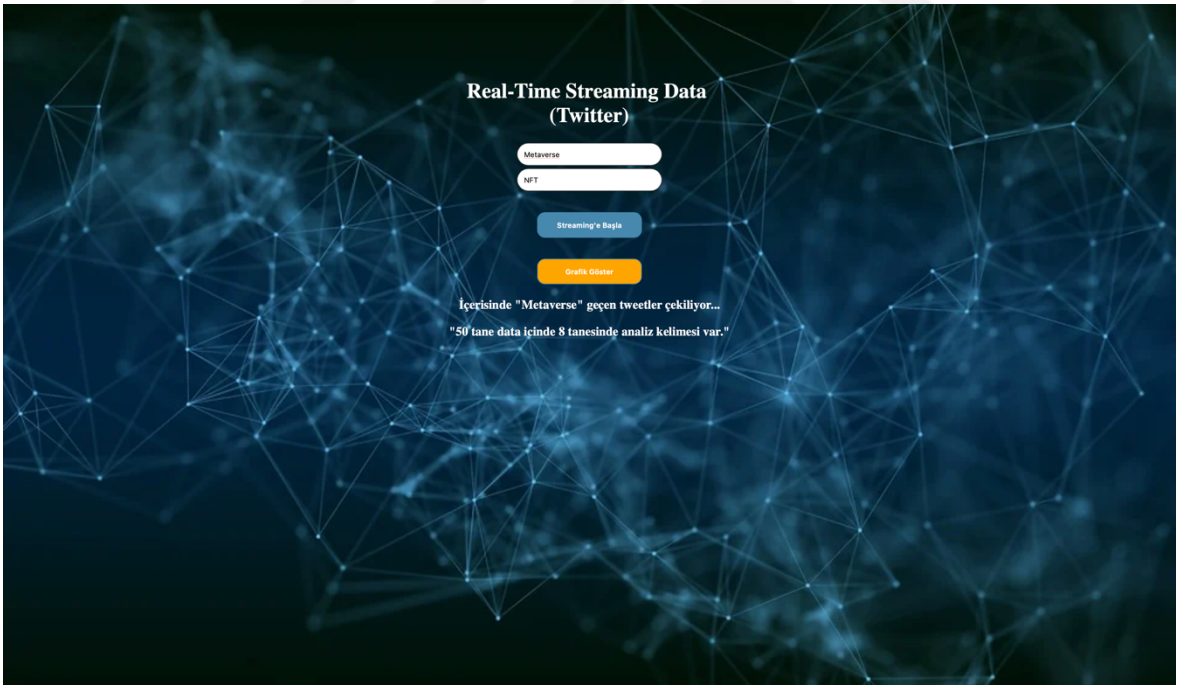
sistemi kullanacak olan kullanıcılara kolaylık sağlamak ve hızlı olduğu kadar kullanıcı dostu da olan bir platform tasarlamak istedik.

```python
@app.route("/")
@app.route("/home")
def home():
    return render_template("index.html")

@app.route("/result", methods=['POST', 'GET'])
def result():
    global text
    global text1
    output = request.form.to_dict()

    if "start" in output:
        print("start keyi geldi")
        text = output["text"]
        text1 = output["text1"]
        main(text, True, text1)
    else:
        print("analiz keyi geldi")
        text1 = analyzeSearchText(allJsons, text1)

    return render_template("index.html", text=text, text1=text1)


if __name__ == '__main__':
    app.run(debug=True, port=5010)
```

Şekil 12. Flask Kodları



Şekil 13. Platform Tasarımı



Şekil 14. Platformun Çalışma Örneği

7. SONUÇ VE ÖNERİLER

Bu bölümde gerçek zamanlı veri işleme için Twitter verisinden, NoSQL veritabanı olan Cassandra'dan, in-memory çalışan veritabanı RabbitMQ'dan, Twitter ile bağlantı sağlamak için kullandığımız Tweepy kütüphanesinden ve web arayüzü oluşturmak için kullandığımız Flask kütüphanesinden faydanalarak oluşturduğumuz TweetCASP platformu ile ilgili sonuçlardan, bu süreçte karşılaştığımız sorunlardan ve bu sorunlara sunduğumuz önerilerden ve bu sistemin gelecekte nasıl daha fazla geliştirilebileceğine dair fikirlerden bahsedeceğiz.

Gelişen teknoloji ile büyük veri hayatımızın her alanına dahil olmuştur. Gün içerisinde sürekli kullanılan akıllı telefonlar ve saatler, sosyal medya platformları, çeşitli cihazlardan üretilen sesli ve görüntülü medyalar büyük verinin hızlı bir şekilde büyümesini sağlamaktadır. Bu çalışmada, sürekli olarak büyüyen bu verilerin gelişme ve büyüme hızını çeşitli alanlarda yapılacak çalışmalar ve geliştirmeler için bir avantaj olarak kullanmayı amaçladık. Yaptığımız bu çalışmada Tweepy kütüphanesi ile Twitter'ın Streaming API'lerine bağlandık ve belirlediğimiz kelimeleri içeren tweet verilerini gerçek zamanlı olarak çekmeye başladık. Çekilen bu verileri in-memory olarak çalışmasının verdiği hız avantajından dolayı RabbitMQ kullanarak kaydettik. Ancak; her ne kadar RabbitMQ hız avantajı sağlasa da verileri bu şekilde hızlı işlemek veri kaybına sebep olabilirdi. Bu nedenden dolayı veri kaybını önlemek için RabbitMQ'ya kaydettiğimiz verileri belirli aşamalarda kalıcı veritabanımız olan Cassandra'ya taşıdık. Böylece veri kaybını önlemeyi amaçladık. Verileri Cassandra'ya taşırken veriler içerisinden sadece "id, date, account, tweet, location" parametlerine ait olanları alarak veri boyutunu küçültmüş olduk.

Çalışma esnasında güncel python versiyonu 3.9'du. Projeyi önce python 3.9 versiyonunda çalıştırmaya çalıştık ancak RabbitMQ python 3.9'u destelemiyordu. Bu yüzden versiyonu 3.7'ye düşürüp tekrar denedik ama Cassandra da python 3.7'yi desteklemiyordu. Çalışan stabil python versiyonuyla ilgili denemelerimizin sonucunda projemizde kullandığımız bütün teknolojilerin ortak desteklediği versiyon olan python 2.7 versiyonu ile projeyi çalıştırmayı başardık.

Gelecekte bu alanda çalışmayı düşünen araştırmacılara tavsiyemiz python 2.7 versiyonu ile çalışmalarıdır. Bu sayede Cassandra, RabbitMQ ve python kütüphaneleri ile rahat bağlantı yaparak çalışmalarına devam edebilirler.

Çalışmamızın sonraki aşamalarında oluşturduğumuz web arayüzünü daha kullanıcı dostu yapmayı, tweet verisi içerisinden sadece "id, date, account, tweet, location" parametrelerine ait verileri değil daha farklı verileri de çekerek daha kapsamlı bir analiz yapabilmeyi ve gelişen python sürümleriyle birlikte kullandığımız teknolojileri ve bu teknolojilerin çalışan versiyonlarını güncellemeyi amaçlıyoruz. Bu çalışmamız sayesinde olası bir doğal afet veya farklı bir acil durumda, akademik araştırma amacıyla ya da son yıllarda tüm dünyada geniş bir yankı oluşturan covid gibi virüslerin yayılımı, aşıları gibi haberleri doğru ve hızlı bir şekilde toplayıp analiz ederek küresel ve ulusal çapta herkesin bundan faydalanmasını hedefliyoruz.

KAYNAKLAR

A, K. K., & Surendran, S. (2012). BigTable , Dynamo & Cassandra – A Review. *International Journal of Electronics and Computer Science Engineering*, *2*, 133–141.

Anggoro, D. A., & Aziz, N. C. (2021). Implementation of K-Nearest Neighbors Algorithm for Predicting Heart Disease Using Python Flask. *Iraqi Journal of Science*, *62*(9), 3196–3219. https://doi.org/10.24996/ijs.2021.62.9.33

Aswathy, A., Prabha, R., Gopal, L. S., Pullarkatt, D., & Ramesh, M. V. (2022). An efficient twitter data collection and analytics framework for effective disaster management. *2022 IEEE Delhi Section Conference, DELCON 2022*. https://doi.org/10.1109/DELCON54057.2022.9753627

Axmark, D., & Widenius, M. (2021). MySQL :: MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL?

Aydin, A., & Anderson, K. (2017). Batch to Real-Time: Incremental Data Collection & Analytics Platform. *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*, 5911–5920. https://doi.org/10.24251/hicss.2017.712

Barlow, M. (2013). *Real-Time Big Data Analytics: Emerging Architecture*. 25.

Bocij, P., Greasley, A., & Hickie, S. (2015). Business information systems: Technology, development and management for the e-business. In *Pearson*.

Brewer, E. (2012). CAP twelve years later: How the “rules” have changed. *Computer*, *45*(2), 23–29. https://doi.org/10.1109/MC.2012.37

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, *39*(4), 12. https://doi.org/10.1145/1978915.1978919

Chauhan, D. (2017). *Using the Advantages of NOSQL : A Case Study on MongoDB*. (February).

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks.

Communications of the ACM, *13*(6). https://doi.org/10.1145/362384.362685

Damji, J. S., Wenig, B., Das, T., & Lee, D. (2020). Learning Spark: Lightning-Fast Data Analytics. In *Learning Spark: Lightning-Fast Data Analytics*.

Databricks. (n.d.). *A Gentle Introduction to 2 Spark*.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1). https://doi.org/10.1145/1327452.1327492

Doguc, B., & Aydin, A. (2019). *Akan Veri İşlemede Popüler NoSQL Veritabanı Teknolojilerinin CAP Tabanlı İncelenmesi*.

Eken, S., Kaya, F., Sayar, A., & Kavak, A. (2014). *Doküman Tabanlı NoSQL Veritabanları : MongoDB ve CouchDB yatay ölçeklenebilirlik kar ş ıla ş tırması*. (October).

Gehring, M., Charfuelan, M., & Markl, V. (2019). A comparison of distributed stream processing systems for time series analysis. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI)*, *P-290*, 205–214. https://doi.org/10.18420/btw2019-ws-21

Han, R., John, L. K., & Zhan, J. (2018). Benchmarking Big Data Systems: A Review. *IEEE Transactions on Services Computing*, *11*(3), 580–597. https://doi.org/10.1109/TSC.2017.2730882

Iot, M. (n.d.). *The Evolution of Smart Stream Processing Architecture*.

Jambi, S., & Anderson, K. M. (2017). Engineering scalable distributed services for real-time big data analytics. *Proceedings - 3rd IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2017*, 131–140. https://doi.org/10.1109/BigDataService.2017.22

Khan, N., Yaqoob, I., Abaker, I., Hashem, T., Inayat, Z., Kamaleldin, W., … Gani, A. (2014). *Big Data : Survey , Technologies , Opportunities , and Challenges*. *2014*. https://doi.org/10.1155/2014/712826

Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: a systematic literature review. *Journal of Big Data*, Vol. 6. https://doi.org/10.1186/s40537-019-0210-7

Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data*, *2*(1), 1–26. https://doi.org/10.1186/s40537-015-0025-0

Marungo, F. (2018). A primer on NoSQL databases for enterprise architects: The cap theorem and transparent data access with MongoDB and Cassandra. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2018-Janua*, 4621–4630. https://doi.org/10.24251/hicss.2018.583

Mihaylov, A., Corvinelli, V., Godfrey, P., Mierzejewski, P., Szlichta, J., & Zuzarte, C. (2021). Scalable Learning to Troubleshoot Query Performance Problems. *International Conference on Information and Knowledge Management, Proceedings*. https://doi.org/10.1145/3459637.3481947

Mukherjee, A., Mondal, S., Chaki, N., & Khatua, S. (2019). Naive bayes and decision tree classifier for streaming data using hbase. *Advances in Intelligent Systems and Computing*, *897*, 105–116. https://doi.org/10.1007/978-981-13-3250-0_8

Neves, P. C., & Bernardino, J. (2015). Big Data in the cloud:A Survey. *Open Journal of Big Data (OJBD)*, *1*(2), 3--20. Retrieved from http://www.slideshare.net/AmazonWebServices/big-data-the-cloud-13953902

Ranjitha, P. (2015). *Streaming Analytics over Real-Time Big Data*. *15*(5).

Razoqi, S. (2021). Data Modeling and Design Implementation for CouchDB Database. *AL-Rafidain Journal of Computer Sciences and Mathematics*, *15*(1). https://doi.org/10.33899/csmj.2021.168252

Rmis, A. M., & Topcu, A. E. (2020). Evaluating riak key value cluster for big data. *Tehnicki Vjesnik*, *27*(1). https://doi.org/10.17559/TV-20180916120558

Rostanski, M., Grochla, K., & Seman, A. (2014). Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ. *2014 Federated Conference on Computer Science and Information Systems, FedCSIS 2014*, 879–884. https://doi.org/10.15439/2014F48

Sakr, S. (2017). Big Data Processing Stacks. *IT Professional*, *19*(1). https://doi.org/10.1109/MITP.2017.6

The PostgreSQL Global Development Group. (2021). What Is PostgreSQL? *Postgresql.Org*.

Wingerath, W., Gessert, F., Friedrich, S., & Ritter, N. (2016). Real-time stream processing for Big Data. *It - Information Technology*, *58*(4), 186–194. https://doi.org/10.1515/itit-2016-0002

Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B., & Vasilakos, A. V. (2016). Big data: From beginning to future. *International Journal of Information Management*, *36*(6), 1231–1247. https://doi.org/10.1016/j.ijinfomgt.2016.07.009

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010*.

Anggoro, D. A., & Aziz, N. C. (2021). Implementation of K-Nearest Neighbors Algorithm for Predicting Heart Disease Using Python Flask. *Iraqi Journal of Science*, *62*(9), 3196–3219. https://doi.org/10.24996/ijs.2021.62.9.33

Aswathy, A. (2022). *An efficient twitter data collection and analytics framework for effective disaster management*.

Axmark, D., & Widenius, M. (2021). MySQL : MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL?

Aydin, A., & Anderson, K. (2017). Batch to Real-Time: Incremental Data Collection & Analytics Platform. *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*, 5911–5920. https://doi.org/10.24251/hicss.2017.712

Barlow, M. (2013). *Real-Time Big Data Analytics: Emerging Architecture*. 25.

Bocij, P., Greasley, A., & Hickie, S. (2015). Business information systems: Technology, development, and management for the e-business. In *Pearson*.

Brewer, E. (2012). CAP twelve years later: How the "rules" have changed. *Computer*, *45*(2), 23–29. https://doi.org/10.1109/MC.2012.37

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, *39*(4), 12. https://doi.org/10.1145/1978915.1978919

Chauhan, D. (2017). *Using the Advantages of NOSQL: A Case Study on MongoDB*. (February).

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, *13*(6). https://doi.org/10.1145/362384.362685

Damji, J. S., Wenig, B., Das, T., & Lee, D. (2020). Learning Spark: Lightning-Fast Data Analytics. In *Learning Spark: Lightning-Fast Data Analytics*.

Databricks. (n.d.). *A Gentle Introduction to 2 Spark*.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1). https://doi.org/10.1145/1327452.1327492

Doguc, B., & Aydin, A. (2019). *Akan Veri İşlemede Popüler NoSQL Veritabanı Teknolojilerinin CAP Tabanlı İncelenmesi*.

Eken, S., Kaya, F., Sayar, A., & Kavak, A. (2014). *Doküman Tabanlı NoSQL Veritabanları : MongoDB ve CouchDB yatay ölçeklenebilirlik kar ş ıla ş tırması*. (October).

Gehring, M., Charfuelan, M., & Markl, V. (2019). A comparison of distributed stream processing systems for time series analysis. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI)*, *P-290*, 205–214. https://doi.org/10.18420/btw2019-ws-21

Han, R., John, L. K., & Zhan, J. (2018). Benchmarking Big Data Systems: A Review. *IEEE Transactions on Services Computing*, *11*(3), 580–597. https://doi.org/10.1109/TSC.2017.2730882

Iot, M. (n.d.). *The Evolution of Smart Stream Processing Architecture*.

Jambi, S., & Anderson, K. M. (2017). Engineering scalable distributed services for real-time big data analytics. *Proceedings - 3rd IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2017*, 131–140. https://doi.org/10.1109/BigDataService.2017.22

Khan, N., Yaqoob, I., Abaker, I., Hashem, T., Inayat, Z., Kamaleldin, W., … Gani, A. (2014). *Big Data: Survey , Technologies , Opportunities , and Challenges*. *2014*. https://doi.org/10.1155/2014/712826

Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: a systematic literature review. *Journal of Big Data*, Vol. 6. https://doi.org/10.1186/s40537-019-0210-7

Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the

right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data*, *2*(1), 1–26. https://doi.org/10.1186/s40537-015-0025-0

Marungo, F. (2018). A primer on NoSQL databases for enterprise architects: The cap theorem and transparent data access with MongoDB and Cassandra. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2018-Janua*, 4621–4630. https://doi.org/10.24251/hicss.2018.583

Mihaylov, A., Corvinelli, V., Godfrey, P., Mierzejewski, P., Szlichta, J., & Zuzarte, C. (2021). Scalable Learning to Troubleshoot Query Performance Problems. *International Conference on Information and Knowledge Management, Proceedings*. https://doi.org/10.1145/3459637.3481947

Mukherjee, A., Mondal, S., Chaki, N., & Khatua, S. (2019). Naive bayes and decision tree classifier for streaming data using hbase. *Advances in Intelligent Systems and Computing*, *897*, 105–116. https://doi.org/10.1007/978-981-13-3250-0_8

Neves, P. C., & Bernardino, J. (2015). Big Data in the cloud:A Survey. *Open Journal of Big Data (OJBD)*, *1*(2), 3--20. Retrieved from http://www.slideshare.net/AmazonWebServices/big-data-the-cloud-13953902

Ranjitha, P. (2015). *Streaming Analytics over Real-Time Big Data*. *15*(5).

Razoqi, S. (2021). Data Modeling and Design Implementation for CouchDB Database. *AL-Rafidain Journal of Computer Sciences and Mathematics*, *15*(1). https://doi.org/10.33899/csmj.2021.168252

Rmis, A. M., & Topcu, A. E. (2020). Evaluating riak key value cluster for big data. *Tehnicki Vjesnik*, *27*(1). https://doi.org/10.17559/TV-20180916120558

Rostanski, M., Grochla, K., & Seman, A. (2014). Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ. *2014 Federated Conference on Computer Science and Information Systems, FedCSIS 2014*, 879–884. https://doi.org/10.15439/2014F48

Sakr, S. (2017). Big Data Processing Stacks. *IT Professional*, *19*(1). https://doi.org/10.1109/MITP.2017.6

The PostgreSQL Global Development Group. (2021). What Is PostgreSQL? *Postgresql.Org*.

Wingerath, W., Gessert, F., Friedrich, S., & Ritter, N. (2016). Real-time stream processing

for Big Data. *It - Information Technology*, *58*(4), 186–194. https://doi.org/10.1515/itit-2016-0002

Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B., & Vasilakos, A. V. (2016). Big data: From beginning to future. *International Journal of Information Management*, *36*(6), 1231–1247. https://doi.org/10.1016/j.ijinfomgt.2016.07.009

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010*.

EKLER

EK A - NoSQL Veri Depolama Sistemlerinin Karşılaştırılması

	Cassandra	***CouchDB***	***HBase***	***MongoDB***	***Redis***	***Riak***	***Firebase***
İkincil veritabanı modelleri	Yok	Yok	Yok	Yok	Belge deposu Graf veritabanı Arama motoru Zaman serisi veritabanı	Yok	Yok
Kullanım popülerliği	Genel: 10 Kendi kategorisi: 1	Genel: 33 Kendi Kategorisi:5	Genel: 17 Kendi kategorisi:2	Genel: 5 Kendi kategorisi:1	Genel: 8 Kendi kategorisi:1	Genel: 49 Kendi kategorisi: 7	Genel:41 Kendi kategorisi: 7
Mevcut sürüm	3.11.4, Şubat 2019	2.3.1, Mart 2019	1.4.8, Ekim 2018	4.0.8, Mart 2019	5.0.4, Mart 2019	2.1.0, Nisan 2015	Platforma göre değişiyor.
Veri şeması	Yok						
XML desteği	Yok						
İkincil indeks	Kısıtlı	Var	Yok	Var	Var	Kısıtlı	Var
Sorgulama Dili	CQL	Yok	Yok	MongoDB bağlayıcı aracılığıyla sadece okuma SQL sorguları	Yok	Yok	Yok
API ve diğer erişim metotları	Özel protokol	RESTful HTTP/JSON API	Java API, RESTful http API	Özel protokol (JSON kullanarak)	Özel protokol	HTTP API Yerel Erlang Arayüzü	Android IOS JavaScript API RESTful http API
Desteklenen programlama dilleri	C#, C++, Erlang, Go, Java, JavaScript, Perl, PHP, Python, Ruby, Scala	C, C#, Erlang, Java, JavaScript, Lisp, Objective-C, Perl, PHP, Python, Ruby	C, C#, C++, Java, PHP, Python, Scala, Groovy	C, C#, C++, Delphi, Erlang, Go, Groovy, Java, JavaScript, Lisp, Perl, PHP, Python, Prolog, Ruby, Scala	C, C#, C++, Erlang, Go, Java, JavaScript (Node.js), Lisp, Matlab, Objective C, Perl, PHP, Prolog, Python, Ruby, Rust, Scala, Swift, Visual Basic	C, C#, C++, Erlang, Go, Groovy, Java, JavaScript, Lisp, Perl, PHP, Python, Ruby, Scala	Java, JavaScript, Objective-C
Bölümleme metodu	Sharding	Sharding	Sharding	Sharding	Sharding	Sharding	Yok
Çoğaltma metotları	Seçilebilir çoğaltma faktörü	Master – Master Master – Slave	Seçilebilir çoğaltma faktörü	Master – Slave	Master – Slave Multi-master çoğaltma	Seçilebilir çoğaltma faktörü	Yok
İşlem Konsepti (transaction)	Yok	Yok	Yok	Anlık görüntü yalıtımlı çoklu belge ACID işlemleri	İyimser kilitleme, komut bloklarının ve dosyalarının atomik yürütülmesi	Yok	Var
Eşzamanlılık (concurrency)	Var						
In-memory	Yok	Yok	Yok	Var (MongoDB 3.2 versiyonu ile)	Var	Yok	Yok

EK B – Apache Akan Veri İşleme Teknolojileri

	Storm	Trident	Samza	Spark
Sıkı düzeyde garanti	En azından bir kez	Tam olarak bir kez	En azından bir kez	Tam olarak bir kez
Ulaşılabilir gecikme	<< 100 ms	< 100 ms	< 100 ms	< 1 s
Durum yönetimi	Evet	Evet (küçük durumlar)	Evet	Evet
İşleme modeli	Birer birer	Mikro parti	Birer birer	Mikro parti
Karşı basınç mekanizması	Evet	Evet	Gerekli değil(arabelleğe alma)	Evet
Sıralı olma garantisi	Hayır	Yığınlar arası	Akış bölümleri içerisinde	Yığınlar arası
Esneklik	Evet	Evet	Hayır	Evet

EK C – Yığın ve Akan Veri İşleme

	Yığın (batch)	***Akan veri (streaming)***
Analiz Karakteristiği	Karmaşık ve detaylı analiz gerçekleştirilir	Basit yanıt fonksiyonları, toplama ve yuvarlama.
Veri İşleme kapsamı	Veri kümesindeki verilerin tamamı veya büyük çoğunluğu üzerinde işlem yapar.	Bir çalışma zamanı içerisinde toplanmış veya sadece en güncel veriler üzerinde işlem yapar.
İşlenen Veri Boyutu	Çok büyük miktarda depolanmış veriler işlenir	Belirli zaman aralıklarında ki az miktarda akan veri işlenir (Örn:10 saniyelik Twitter verisi)
İşlem Süresi	Dakikalar veya saatler	Saniye veya Milisaniye

ÖZGEÇMİŞ

Ad-Soyad : Tuğba Beril DOĞUÇ

ÖĞRENİM DURUMU:

- **Lisans :** 2014 - 2018, İnönü Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği
- **Yüksek Lisans :** 2018 - Halen, İnönü Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı

MESLEKİ DENEYİM:

- 2019 - 2020 T-Soft E-Ticaret Yazılımları, İstanbul şirketinde iOS Developer olarak çalıştı.
- 2020 – Halen Arneca Teknoloji, İstanbul şirketinde iOS Developer olarak çalışmaktadır.

ÖNCEKİ ÇALIŞMALAR

- (Doguc & Aydin, 2019)**,** Akan Veri İşlemede Popüler NoSQL Veritabanı Teknolojilerinin CAP Tabanlı İncelenmesi