T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

HADOOP / MAPREDUCE TEKNOLOJİSİ KULLANILARAK HIZLI TÜKETİM SEKTÖRÜNDE BÜYÜK VERİ ANALİZİ

Serdar ÇETİNKAYA

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Yrd. Doç. Dr. Fatih KELEŞ

Haziran 2016

İSTANBUL

Bu çalışma 30.06 2016 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği programında Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Jürisi:

İmza
Yrd. Doç Dr. Fatih KELEŞ (Danışman)
İstanbul Üniversitesi
Mühendislik Fakultesi

Prof. Dr. Nimet SERTBAŞ
İstanbul Üniversitesi
Mühendislik Fakultesi

Doç. Dr. Atakan KURT
İstanbul Üniversitesi
Mühendislik Fakultesi

Yrd. Doç. Dr. Selçuk SEVGEN
İstanbul Üniversitesi
Mühendislik Fakultesi

Yrd. Doç. Dr. Gökhan Bilgin
Yıldız Teknik Üniversitesi
Mühendislik Fakultesi

ÖNSÖZ

Çalışmalarım süresince değerli zamanını ayırarak beni yönlendiren tez danışmanım Sayın Yrd. Doç. Dr. Fatih KELEŞ'e, verilerini kullanarak analiz yapmama izin vermeleri ve desteklerinden ötürü Hedef Gıda A.Ş'ne, manevi destek ve anlayışları için aileme, Kağan, Cansu ve Meerim'e teşekkürlerimi borç bilirim.

Haziran, 2016 Serdar ÇETİNKAYA

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar	Açıklama
HDFS	**:** Hadoop Distributed File System
YARN	**:** Yet Another Resource Negotiator
CRUD	**:** Create, Read, Update, Delete
PSA	**:** Pazar Sepeti Analizi
HDP	**:** Hortonworks Data Platform
RDBMS	**:** Relational Database Management System
ETL	**:** Extract Transform Load
RAID	**:** Redundant Array of Inexpensive Disks
RAM	**:** Random Access Memory
CPU	**:** Central Processing Unit – İşlemci
SSH	**:** Secure Shell
NTP	**:** Network Time Protocol
ASF	**:** Apache Software Foundation
FMCG	**:** Fast Moving Consumer Goods
CERN	**:** Conseil Européen pour la Recherche Nucléaire

ÖZET

YÜKSEK LİSANS TEZİ

HADOOP / MAPREDUCE TEKNOLOJİSİ KULLANILARAK HIZLI TÜKETİM SEKTÖRÜNDE BÜYÜK VERİ ANALİZİ

Serdar ÇETİNKAYA

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Yrd. Doç. Dr. Fatih KELEŞ

Hayatımızın her alanına giren internet ile birlikte hızlı bir şekilde ürettiğimiz veriler eğer kullanılmaz ise devasa veri yığını olmaktan öteye geçemez. Eğer üretilen bu verileri kullanarak bilgiye dönüştürebiliyor isek o zaman toplumu etkileyen bir veri elde edebiliriz. Bu çalışmanın amacı geleneksel yollar ile analizi mümkün olmayan veya çok fazla zaman alan analizlerin nasıl daha kısıtlı kaynaklar ile yapılabileceğini tartışmaktır. Büyük veri, analiz edildiğinde insanların yaşam ve anlayış biçimini değiştirecek veriler olarak kabul edilmiştir.

Büyük verinin analizi için Hortonworks'un Hadoop platformunda MapReduce ve HDFS ile pazar sepeti analizi ve k-means ile kümeleme algoritmaları üzerinde çalışılmıştır. Bu çalışmalar neticesinde müşteri kanalları miktara ve karlılığa göre yeniden oluşturulmuştur. Ayrıca pazar sepeti analiz'i, MapReduce tekniği ile yapılarak ürün birliktelikleri bulunmuştur. Bu analiz ile ürünlerin birliktelikleri belirlenerek aşağıdaki maddelerin sağlanması amaçlanmıştır.

- Müşterilere ürün önerme,
- Müşterilerin alışkanlıkları hakkında fikir edinme,
- Raf dizilimi,
- Kampanya modellerinin belirlenmesi
- Müşteri memnuniyeti,

Hadoop üzerinde MapReduce tekniği ile paralel olarak yapılan analizlerin veri miktarına bağlı olarak belirli bir düğüm sayısına kadar kazanım sağladığı gözlemlenmiştir. Veri miktarı arttırıldıkça en optimum sürede analiz etmek için kullanılacak düğüm sayısını arttırmak gerekmektedir. Ayrıca veri miktarına bağlı olarak belirli bir düğüm sonrasında düğüm sayısını arttırmanın kazanım sağlamadığı aksine daha uzun zamanda analiz edildiği gözlemlenmiştir.

Bu çalışma ile paralel işlemenin geleneksel yöntemlere oranla daha verimli olduğu gözlemlenmiştir. Aynı zamanda Hadoop platformunun kullanılması ile daha etkin sonuçlar alındığı belirlenmiştir.

Haziran 2016, 80 sayfa.

Anahtar kelimeler: Büyük Veri, Hadoop, MapReduce, Pazar Sepeti Analizi, K-means Algoritması

SUMMARY

M.Sc. THESIS

BIG DATA ANALYSIS IN FAST MOOVING CONSUMER SECTOR BY USING HADOOP / MAPREDUCE TECHNOLOGY

Serdar ÇETİNKAYA

İstanbul University

Institute of Graduate Studies in Science and Engineering

Department of Computer Engineering

Supervisor : Asst. Prof. Dr. Fatih KELEŞ

If the data we produce quickly with the internet that has been into every aspect of our lives will not be used, it will be no more than a huge pile of data. If we are able to convert the information generated by using this data, then we can obtain a data that affects the community. The purpose of this study is to discuss how can we carry out these analyses which cannot be done with limited sources and takes too much time. In this study "Big Data" is considered as a data which can change people's lives and formats of their mentality.

For Big Data analysis, Market Basket Analysis and Clustering Algorithm with MapReduce and HDFS have been studied. As a result of these studies customer channels have been re-created according to the amount and profitability. Besides, product association have been found by Market Basket Analysis which had been performed by MapReduce tecnique. With this analysis we aim to provide:

- Product proposition to customers by determining product association,
- To obtain an idea about the habits of the customers,
- Shelf layout,
- To determine campaign models,
- Customer satisfaction.

It has been observed that analyses which are performed on Hadoop in parallel processing with MapReduce technique can obtain gains up to certain number of nodes. However,

after certain number of nodes no gains have been observed depending on the data structure and data amount. The amount of data in the most optimum time will be used to analyze the need to increase the number of nodes.

June 2016, 80 pages.

Keywords: Big Data, Hadoop, MapReduce, Market Basket Analysis, K-means Algorithm

1. GİRİŞ

İnsanlar hayatlarını bilgi ile ileriye taşır ve anlamlandırırlar. Bunların yanında bilgisayar bilimlerinde günden güne gelişen teknolojiler ve yenilikler sayesinde bilgi artışı baş döndürücü şekilde hızlanmıştır. Bilgi artışına sosyal medyanın yaygınlaşması, mobil cihaz dağılımının artması ile hayatın her aşamasında kullanılan algılayıcılar de katkı sağlamaktadır. Ayrıca bilgiye erişmenin ve bilgiyi kullanmanın çok kolay ve hızlı olduğu bir dünyada yaşıyoruz. Bu sebepten 2000'li yıllar sonrası yaşayan insanlar bilgi toplumu diye adlandırılıyor. Artık herkesin cebinde bir akıllı telefon, herkesin evinde bir bilgisayar ve tüm şirketlerin ofislerinde bilgi teknolojileri yönetimini yapan birimler bulunmaktadır. Hatta Amerikan Federal Ticaret Komisyonunun gündelik hayatta kullandığımız cihazların internete bağlanarak veri alış verişinde bulunması yeteneği[1] olarak tanımladığı nesnelerin interneti de dâhil sürekli bir bilgi üretme çağında yaşıyoruz.

Büyük veri ile ilgili Forbes dergisinde yayınlanan aşağıdaki bilgileri incelendiğinde tam ne anlama geldiği daha iyi anlaşılacaktır [1].

- Son iki yılda üretilen veriler insanlığın o güne kadar ürettiği toplam veri boyutundan büyüktür.
- Veri her zamankinden daha hızlı büyüyor, 2020 yılında saniyede her insan için 1,7 MB veri üretiliyor olacak.
- Bugün 4,4 ZB (4,4 Trilyon GB) olan veri boyutu 2020'de 44 ZB olacak.
- Her saniye yeni veri üretilmektedir. Örneğin Google üzerinde her saniyede 40,000 arama sorgusu çalıştırılmaktadır.
- Ağustos 2015 itibari ile 1 Milyar insan Facebook kullanıyor.
- Facebook kullanıcıları her dakikada 31,25 milyon mesaj gönderiyorlar.

Yukarıda sıralanan maddelerde de görüleceği üzere verinin artması ile büyük verinin hacmi ve artma hızı, hemen hemen yaşamın her aşamasında insanları etkilemektedir.

[1] https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf

Günümüzde sağlık, hizmet, üretim, spor ve yenileşme (inovasyon) kadar birçok sektörde yaygın olarak kullanıldığı görülmektedir. İşletmeler büyük veri değişiminin farkında olarak, yatırım yapmaya başlamışlardır. Yapılan bazı çalışmalarda bunu destekler niteliktedir. 2012 yılında 600 adet uluslararası firma üzerinde yapılan bir araştırma referans alındığında bu farkındalık görülmektedir. Bu araştırmaya göre bu firmaların %75 kendilerini veri eğilimli olarak nitelemiştir. Ayrıca büyük veri on üzerinden puanlandığında dokuz puanla üretim, işgücü ve sermayeden sonra dördüncü faktör olarak sınıflandırmıştır [2] . Bunların yanında büyük boyutta ki veri yığınları arasında çözümleme yapılabilecek bir ortam ile karar vermek ve sonuca ulaşmak çok karmaşık bir hal almıştır. Bu durumda büyük verinin hacmi ne olursa olsun, bu büyük veri ile ne yapılabileceğinin incelenmesi ve örüntülerin bulunması gerekmektedir.

Veri miktarlarının her gün bir önceki güne oranla hızlı artması yüzünden hayatımıza giren büyük veriyi sadece boyutsal olarak ele almak yanlış olacaktır. Doğru yaklaşım verilerin doğru sınıflandırılmasıyla elde edilecek sonuçlar ile iş yapma şekillerini etkileyecek bilgi üretmek sayesinde olacaktır. Sistemlerin ürettiği veriler bir başlarına pek anlamlı değillerdir. Veriler belli bir amaç doğrultusunda çözümlendiğinde değerli olurlar.

Sadece ham veri ya da veri kopyaları ile geçmiş verileri üzerinden gidilerek sonuç elde etmek tek başına anlam ifade etmemekle birlikte pek başarılı bir yöntem olarak da görülmemektedir. Bu sebepten büyük boyutlara sahip verileri çözümleyen yöntemler ve teknikleri kullanmak fazlaca değer kazanmaktadır. Büyük boyutlarda ki verilerin anlaşılır hale getirilerek bilgiye dönüştürülmesi için veri madenciliği yöntemleri kullanılmalıdır. Veri madenciliği büyük veri yığınları arasında bulunan eğilim, örüntü ve benzerlikleri bulma ve faydalı olabilecek bilgiye erişme işlemidir [3].

Veri madenciliği sayesinde işletmeler müşteri davranış ve alışkanlıklarının yanında eğilim ölçmek, müşterilerin hangi konulara yöneldikleri hakkında fikir edinmek ve müşterilerin farklı durumlarda nasıl bir tepki vereceğini tahmin edebilmektedirler. Bu sayede sahaya uyguladıkları özendirme (promosyon) ve kampanyaların sahada nasıl algılandığı ve geri dönüşünü görebiliyorlar. Geniş veri yığınları içerisinde, faydalı olabilecek beklenmeyen ilişkileri keşfetme, anlaşılır ve kullanılabilir bir yapıya dönüştürmeye yönelik geliştirilen yöntemlerde veri madenciliği yöntemleridir. Bu yapılar karar verme aşamasında oldukça yapıcı faydalar sağlamaktadır. Sonuç olarak amaç

çözümleme yaparak bilgiye ulaşmak ve bu sayede kazanım elde etmektir. Veri madenciliği, bir süreç olarak ele alınmalıdır. Veri yığınları arasında, yöntemler ile kazı yapılarak bilgiyi ortaya çıkarmaya yaramaktadır. Bunun yanında bilgi keşfi aşamasında bulunan örüntüyü ayrıştırıp filtrelemek ve diğer aşamada kullanılabilir yapıya getirmek bu sürecin bir parçasıdır.

Böyle büyük verilerin olduğu bir dünyada işletmelerin veri madenciliği ile sektörlerinde rekabetçi olabilmesi ve varlıklarını koruyabilmesi adına müşterilerine en uygun ürünü en uygun şekilde sunmaları gerekmektedir. Bu sayede müşteri memnuniyetini sağlayarak aynı zamanda ticari faaliyetlerini daha karlı sunabilme imkânı bulacaklardır. İşletmelerde tüketici alışkanlıklarının araştırılması için büyük verinin çözümlenmesi günden güne önem kazanmaktadır. Büyük veri sayesinde, satış ve pazarlamada kullanılmak üzere önceki zamanlara oranla günümüzde daha çok yöntem ve araç olmasına rağmen işletmelerin sadece %12'sinin bu katma değerli imkândan faydalandığını araştırmalardan görüyoruz. Büyük veri, birçok firma ve pazarlama ekipleri tarafından nasıl kullanacakları konusunda bilgi sahibi olmadıkları sihirli bir yapı veya kelime gibi algılanmaktadır.

İşletmelerin pazarlama stratejisi üç temel madde üzerine kuruludur:

- Kapsamlarına yeni müşteri katmak
- Müşterilerin aidiyetlerini pekiştirmek
- Daha fazla ürün ve hizmeti satarak daha fazla kar elde etmek

Bu hedefleri ulaşmak için bazı sorulara cevap bulmak gerekmektedir. Sorulara cevap vermek için satış veya pazarlama bölümleri, müşterilerinin satın almayı nasıl bir yol izleyerek gerçekleştirdiğini çözümlemeli ve buna göre hareket etmelidirler. İşletmelerde sorulacak temel sorular aşağıda ki gibidir:

- Kime?
- Nerede?
- Hangi Kanal (Ticari şartlar)’da satış yapılmalı?
- Hangi kanala önem verilmeli?

Satın alım için müşterileri izledikleri yolları, bu müşterilerin gruplandırılmasının, müşteri eğilimlerinin ve aynı zamanda nelere ilgi duyabileceklerinin önceden belirlenmesi ve

satış elemanları tarafından rotasındaki ziyareti sırasında biliniyor olması satış aşamasında oldukça etkili bir sonuç doğuracaktır. Bu sayede satış ve pazarlama için yeni bir fırsat yaratılmış olacak ve kazanım elde edilmiş olacaktır.

Bu çalışma giriş ve sonuçla birlikte beş bölümden oluşmaktadır. Birinci bölüm giriş bölümü olmakla birlikte, bu bölümde büyük veri hakkında kısaca bilgi verilmiştir. Ayrıca çalışmanın yapılacağı sektör ve işletmeler için büyük verinin önemi vurgulanmıştır.

İkinci bölümde Büyük veri kavramı ele alınarak incelenmiştir. Büyük veriyi oluşturan bileşenler yani büyük verinin özellikleri ele alınmış, büyük verinin temel özellikleri yanında farklı tanımları olan hacmi, çeşitliliği ve hızının yanından diğer özelliklerine ve tanımlarında da değinilmiştir. Ayrıca büyük verinin saklanması ve işlenmesi için kullanılan araçlara değinilerek, bu araçlar ve yöntemler tartışılmıştır. Bu çalışmada kullanılan Hadoop ve özellikleri incelenerek paralel işleme tekniklerinin Hadoop üzerinde uygulanması incelenmiştir. Paralel işleme için kullanılan MapReduce yöntemi ve analiz için kullanılacak algoritmalar tartışılmıştır.

Üçüncü bölümde ise kullanılan yöntem ve araçlar anlatılmıştır. Hadoop teknolojinin sağlayıcılarına ve bu çalışmada kullanılan Hortonworks'un platformu olan Hortonworks Data Platform'un neden seçildiği bilgisine yer verilmiştir.

Dördüncü bölümde ise Hızlı Tüketim Sektöründe (FMCG) üretilen veriler üzerinde ilk olarak popüler kümeleme algoritmalarından k-means kullanılarak müşteri karlılıkları ve ürün karlılıkları incelenmiştir. Daha sonrasında ise Pazar Sepeti Analizi (PSA) algoritmaları kullanılarak Hadoop üzerinde paralel işleme ile büyük veri analizlerinde elde edilebilecek kazanımlar incelenmiştir.

Son bölüm, sonuç ve değerlendirme kısmıdır. Bu bölümde çalışmada kullanılan teknik ve ele alınan yöntemlerle yapılan çalışmanın özellikleri değerlendirilmiştir. Elde edilen sonuçlar yorumlanmıştır. Yapılan çalışma ile büyük verilerde Hadoop üzerinde paralel işleme ve MapReduce ile paralel işlemede elde edilen kazanımlar yorumlanmıştır. Hadoop üzerinde MapReduce kullanılarak sıradan makinalar ile veri çözümlemeleri yapılmış olup elde edilen kazanımlar değerlendirilmiştir.

2. GENEL KISIMLAR

2.1. BÜYÜK VERİ NEDİR

Büyük veri, mevcut geleneksel yöntem ve araçlarla çözümleme yapılamayacak kadar büyük olan, yönetilmesi anlamında geleneksel sistemlerin yetersiz kaldığı büyüklükteki veri boyutlarını tarif etmektedir[4]. Büyük veri çözümleme için ihtiyaç duyulan sistemlere olan alakanın artmış olmasının birçok sebebi vardır. Mevcut veriler kullanılarak yeni müşteri ve bilgi keşfedilmesi, araştırma yapıldığında neticeye ulaşmak için uzun sürelerin nasıl kısaltılabileceği, toplumsal olarak yapılacak işlem ve hizmetlerin araştırılması, sağlık sektöründe hastalıkları önlenebilmesi için erken teşhis sistemlerinin geliştirilmesi, trafik yoğunluğunun ve kazaların azaltılması gibi farklı dallarda faydaları görülmeye başlamıştır. Tablo 2.1'de veri boyutları sunularak, bu büyüklükteki verileri yönetmek (saklama, arama, paylaşma, analiz, görselleştirme) ne derece maliyetli ve zor olduğu bir nebze olsun anlatılmaya çalışılmıştır. Büyük verinin boyutu hakkında verilebilecek ilginç ve güncel bir örnek CERN(Avrupa Nükleer Araştırma Merkezi)'de yapılan deneylerde, Hadron Çarpıştırıcısı saniyede 40 terabyte veri üretmektedir [5].

Tablo 2.1: Veri büyüklükleri.

Birim	**Boyut**
Bit	1 ya da 0
Byte	8 bits
Kilobyte	1000 ya da 2^{10} bytes
Megabyte	1000 KB; 2^{20} bytes
Gigabyte	1000 MB; 2^{30} bytes
Terabyte	1000 GB; 2^{40} bytes

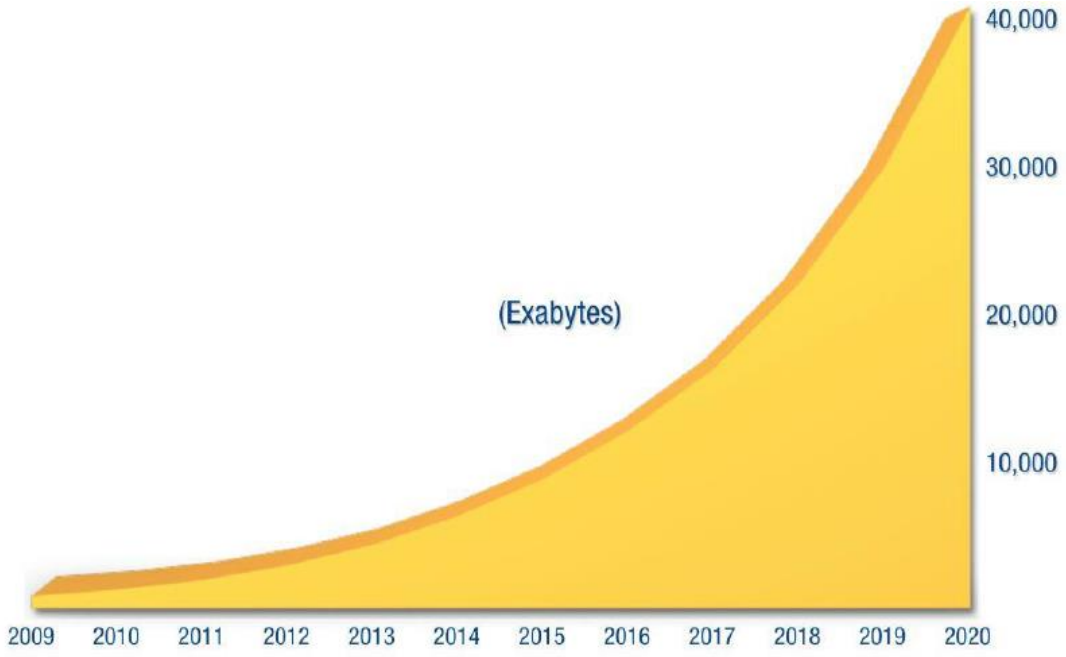
Tablo 2.1 (Devamı)

Petabyte	1000 GB; 2^{50} bytes
Exabyte	1000 PB; 2^{60} bytes
Zettabyte	1000 EB; 2^{70} bytes
Yottabyte	1000 ZB; 2^{80} bytes

Günümüzde ki şirketlerin birçoğu öz değerlerinin yükselmesi ve rekabete uyum için büyük veri yatırımı yapmaktadırlar. Bu firmalar IBM, Oracle, Google, HP, Hortonworks ve başka şirketlerdir. 15 yıla yakın bir zamandır büyük boyutta ki (terabyte düzeyinde) veriler için şirketler veri ambarı kullanmaktaydılar. Fakat günümüzde veriler çeşitli formatlarda oluşmakla birlikte sahadan toparlanıp, depolanmaktadır. Bu verilerden belirli bir yapıya sahip olanlarına yapısal veri, belirli bir yapıda olmayanlarına ise yapısal olmayan veri denilmektedir. Veri ambarları büyük veri yığınları için günümüzde tam bir çözüm sunmamaktadır. Çünkü bu veri yığınını işlemede çok fazla kaynak ve çok fazla zaman gerektiği görülmüştür. Tam bu noktada ortaya çıkan büyük veri teknolojisi veri yığınları üzerinden bilgiye ulaşmak için farklı kaynakların aynı anda paralel olarak işletilmesini ve bu işlem sırasında da veri doğruluğu, hata toleransı gibi durumları kullanıcı yerine kullanılan sistemin yapmasını sağlamaktadır. Bu haliyle büyük veri teknolojisinin kullanılması kaçınılmaz bir hal almaktadır.

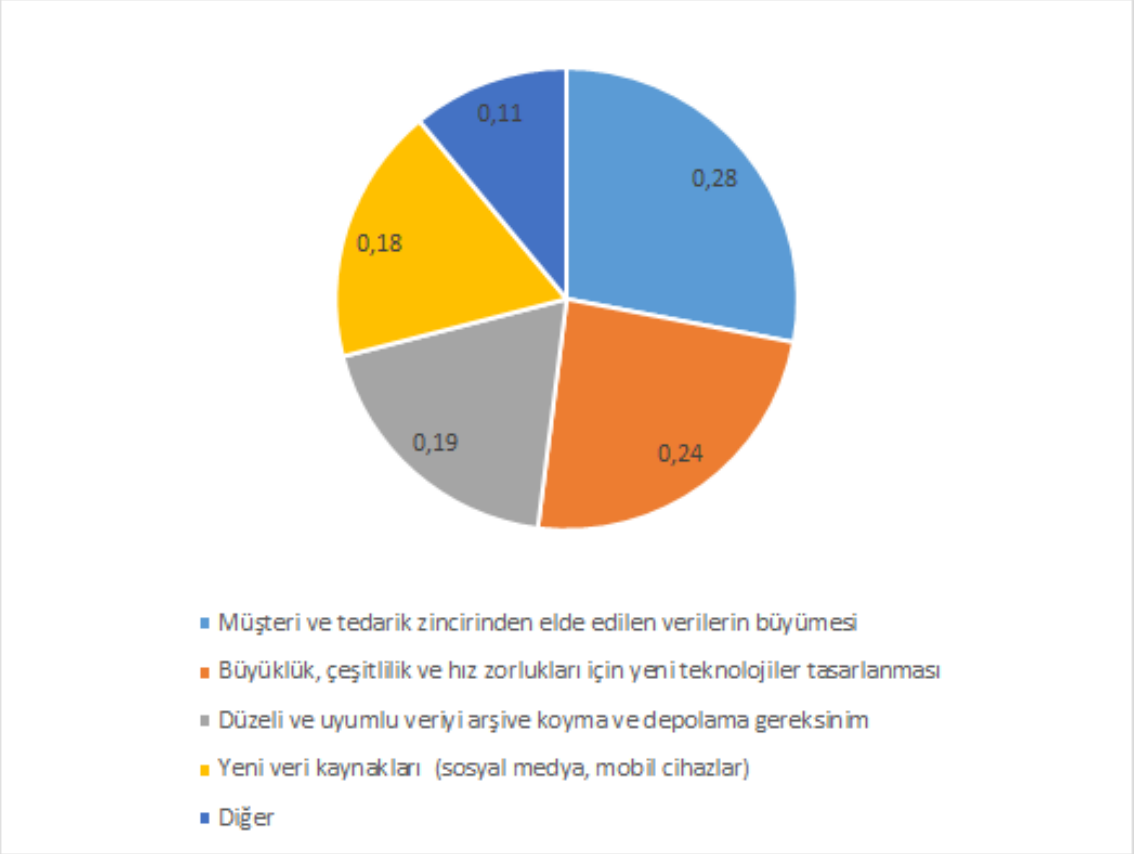
IDC tarafından yapılan "dijital universe" çalışması ile 2009 ve 2020 yılları arasında sayısal veri boyutlarının 44 misli artacağı düşünülmektedir. Bu araştırmada ayrıca veri boyutunun her yıl için 35 ZB büyüdüğü belirtilmiştir[6]. Şekil 2.1'de 2010 – 2020 arasında veri artış oranları verilmiştir. Bu araştırmada ilgi uyandıran bir diğer konu verinin en fazla oluşacağı alanlar olarak algılayıcılar, akıllı telefonlar, tabletler olacağı belirtilmektedir. Bu veri ve veri sağlayıcı cihazların artması ile sağlık, bilim araştırma ve

finans alanlarında katma değer sağlanacağı düşülmekte ve bu alanlarda yapılacak yeniliklerle toplumda belirli kazanımlar sağlanacağı belirtilmektedir.



Şekil 2.1 : IDC Araştırmasında verilerin artış hızını gösterir eğri.

Büyük veri teknolojik gelişmelerle birlikte çok hızlı ve geniş yelpazeli bir şekilde geliştiği için farklı disiplinlerde farklı anlamlarla ifade edilebilmektedir. Disiplinler arası bir kavram olması nedeniyle hem farklı disiplinlerdeki araştırmacıların hem de büyük veri ile ilgilenen kurumların büyük veriye yükledikleri anlam ve bakışı farklılıklar göstermektedir. Örneğin SAP şirketinin 2012 yılında 154 şirket üst düzey yöneticisi ile yapmış olduğu araştırmaya göre büyük verinin kavram olarak algılanması ve kullanımı değişiklik göstermektedir. Yapılan araştırmanın sonuçları Şekil 2.2'de verilmiştir. Görüldüğü üzere bazı şirket yöneticileri verinin boyutu ile ilgilenmekte ve öyle algılamaktadır, bazı şirket yöneticileri ise ne yapılabileceği ile ilgilenmektedir[7]. Bu araştırmadan ulaşılan sonuçlara bakıldığında büyük veri kavramı için tam bir tanım ve algı söz konusu olmadığı görülmüştür.



Şekil 2.2: SAP Şirketinin araştırmasına göre "Büyük veri algısı".

Büyük veri üzerinde çalışmalar yapan Schönberger ve Cukier [8], büyük veri kavramının bir tanımı olmadığını belirtmekte ve tanımlamaların çeşitlilik gösterdiğine atıf yapmaktadırlar. Yukarıda belirtilen araştırma ve Schönberger ve Cukier [8] tarafından yapılan çalışmalar ışığında büyük veriyi kavramı için yapılan çalışmaları taramak ve genel kanıya varmak gerekmektedir.

Rubinstein[9] büyük veriyi işlevselliği ve uygulanabilirliği olarak incelemiştir. İşletmeler ve devlet için büyük verinin farklı veri yığınlarının içerisinde gizli kalmış bilgilere istatistik ve veri madenciliği ile ulaşmak olarak tanımlamıştır.

Yine farklı bir tanımlama ise Beyer ve Laney [10], tarafından yapılmıştır. Bu tanımlamada büyük verinin boyutları ele alınmıştır. Bu tanıma bakıldığında büyük veri, kurum ve kuruluş içinde karar verme yapısını hızlandırmak adına veriyi süreçlere

dönüştürülmesinin ve maliyeti arttırıcı yüksek hacim, yüksek hız, büyük çeşitliliğe sahip bilgi kümeleridir.

Dumbill [11] ise farklı kavramsal bir tarzda ele almıştır. Şu açıklamayı getirerek kavramı tanımlamaya çalışmıştır. Büyük veri, geleneksel veri tabanı sistemleri ile analiz edilemeyecek boyutlara ulaşmış veridir. Bu veri boyut olarak çok büyüktür ve çok hızlı hareket eder veya veri tabanında kaynak yeterli değildir. Bu veri yığınından bilgi ve kazanım sağlamak, onu işlemek için başka yöntem bulunması gerekmektedir.

Bu tez çalışmasında büyük veri kavramı yukarıda değinilen tanımlar ışığında aşağıdaki gibi tanımlanmıştır. Büyük veri, geleneksel veritabanı ve yazılım araçları ile depolanıp yönetilemeyen ve analiz edilemeyen veya bunları yapma kapasitesini aşan büyüklükteki verileri temsil etmekle birlikte analiz edildiğinde toplumun yaşam ve anlayış biçimini değiştirecek boyuttan bağımsız değer katan veriler olarak kabul edilmektedir.

2.2. BÜYÜK VERİNİN ÖZELLİKLERİ

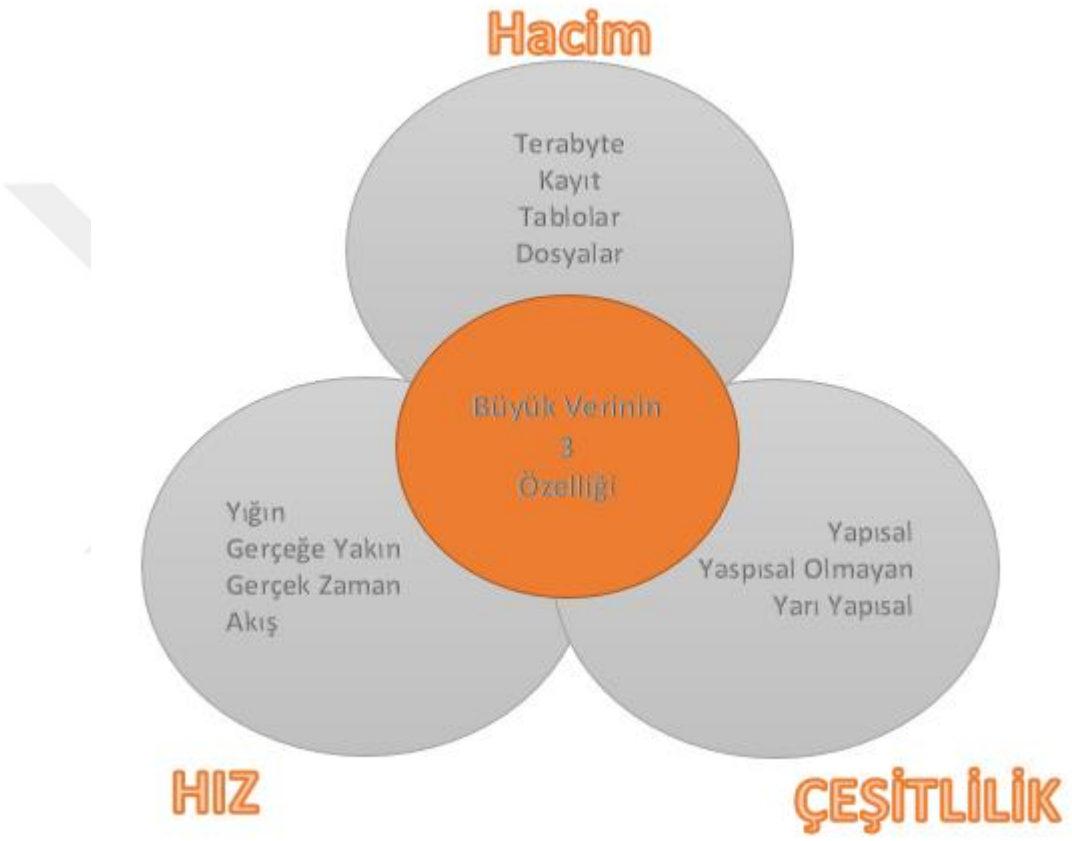
Büyük verinin tam bir tanımı olmaması özelliklerini de etkilemektedir. Bazı tanımlarda veri hacmi (volume), veri hızı (velocity) ve veri çeşitliği (variety) olarak 3 özellik belirtilmektedir. Fakat farklı tanımlarda farklı özellikleri de belirtilmektedir. En çok kabul edilen ve hemen hemen her tanımda belirtilen 3V (volume, velocity, variety)'dir. Farklı tanımlamalarda ise doğruluk (verification) ve değer (value) de özellik olarak ele alınmıştır ve 5V olarak belirtilmektedir. Diğer yandan kimi kaynaklarda ise Büyük veri boyutları hacim, hız ve çeşitliliğe ek olarak sadece doğruluk veya sadece değer eklenerek 4V olarak da yayıncalarda yer almaktadır.

Büyük verinin özellikleri için kabul edilen genel bir tanım yoktur. Fakat kaynaklarda büyük veri bileşenleri, elemanları, özellikleri gibi farlı isimlerle ifade edilmektedir. Şekil 2.3'de görüldüğü gibi veri hacmi verinin miktarını, veri hızı; algılayıcılar, mobil gibi kaynaklardan hızlı bir veri akışını, veri çeşitliliği ise veri kaynak ve yapı farklılığını ifade etmektedir.

2.2.1. Büyüklük (Volume)

Özellik olarak büyük veri denilince akla ilk gelen özelliktir. Temelde verinin boyutunu belirtmek için kullanılmaktadır. Büyük verinin miktarının terabyte ve petabyte boyutlarına ulaşabildiği araştırmalardan görülebilmektedir. Mesela Facebook'ta

depolanan yaklaşık 260 milyar fotoğrafın 20 petabyte olduğu tahmin edilmektedir[12]. 2013 yılında yapılan başka bir araştırmaya göre ise dünyada 41.821 civarında havaalanı olduğu saptanmıştır[13]. Bunların her bir uçuş ve müşteri, güvenlik bilgilerinin saklanması standart ve geleneksel yöntemler ile pek mümkün gözükmemektedir. Dolayısı ile büyük verinin en temel özelliği olan büyüklük veri hacmini ifade etmektedir.



Şekil 2.3: Büyük verinin özellikleri.

2.2.2. Çeşitlilik (Variety)

Veri yığınları genellikle hep aynı veri türü ve yapısında olmamaktadır. Çeşitlilik özelliği, veri tiplerini ve kaynağını belirtmektedir. Veri kaynakları çok farklı olabilmektedir. Bunlar mobil telefonlar, algılayıcılar veya sosyal medya gibidir. Kaynak olarak farklı mecralardan gelen bu veriler form olarak da değişkenlik göstermektedir. Veri formları yapısal veriler, yarı yapısal veriler ve yapısal olmayan veriler olarak üçe ayrılmaktadır. İlişkisel veri tabanında bulunan veya belirli formlarda tutulan verileri yapısal verilere

örnek verebiliriz. Kullanıcı geri bildirimi, e-mail veya log dosyalarını yarı yapısal verilere örnek olarak verilebilir. Yapısal olmayan verilere görüntü, ses veya metin dosyaları örnek verilebilir. Günümüzde verileri yapılarına göre sınıflayacak olursak; araştırmalar verilerin %95'nin yapısal olmadığı, geri kalan %5'nin yapısal veri olduğu ortaya çıkmaktadır [14].

Özellikle sistemler için yapısal bir formatta olan veriler daha kolay işlenebilmektedir. Özellikle yapısal olmayan verilerin işlenmesi zorlu süreçler sonrası yapılabilmektedir.

2.2.3. Hız (Velocity)

Sürekli yeni teknolojik mecraların açılması ve hızlı gelişim sayesinde veri üretilme hızı da artmıştır. Bu veri hızının artması algılayıcılar gibi sürekli veri üreten kaynakların oluşması veri yığınlarının çok dinamik ve değişken olmasına sebep olmaktadır. Bu da veri yığınının çözümlenmesini ve işlenmesini zorlaştırmaktadır. Örneğin; modern bir aracın 100'ü aşkın algılayıcısı olduğu ve kullanım anında bu algılayıcıların veri üretmeye devam ederken, sistem bu veriler ışığında bir sonraki adıma karar verilmelidir. Bu karar sayesinde belki insan hayatı söz konusu olacağından hız çok büyük önem arz etmektedir. Google'ın insansız araç projesinde sadece normal araçların algılayıcıları dışında çevresel etkileri anlayıp yorumlaması gereken sistemler ve insansız hava sistemleri gibi insan hayatını kolaylaştıran ve tehlikeye sokabilecek sistemlerin hayatımıza girmesi ile veri işleme hızı özellikle önemli duruma gelmiştir.

2.2.4. Değer (Value)

Oracle tarafından büyük verinin diğer özelliklerinin yanında dördüncü özellik olarak değer ele alınmaktadır. Büyük verinin en önemli özelliklerinden birisi olarak ortaya çıkmaktadır. Nitekim toplanan verinin doğruluğuna karşılık gelmektedir[15]. Büyük verinin işlenmesi sonucunda ortaya çıkan değer, tüm veri yığınını anlamlı hale getirmektedir. Diğer tüm büyük veri özellikleri veri yığınındaki değeri ortaya çıkarmaya hizmet etmektedir.

2.3. HADOOP EKOSİSTEMİ

2.3.1. Hadoop

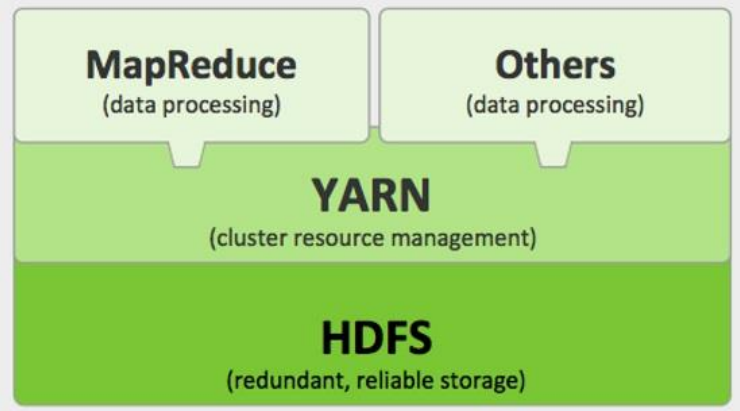
Hadoop, sıradan ve hatta kişisel bilgisayarlar boyutunda ki sunuculardan oluşan kümeler (cluster) ile büyük verileri analiz etmek için geliştirilmiş bir sistemdir. Üzerinde çalışan

uygulamaları ve Hadoop Distributed File System (HDFS) olarak adlandırılan dağıtık dosya sistemi ile MapReduce tekniğini bir araya getirmektedir. Java ile geliştirilmiş açık kaynaklı bir kütüphanedir. Hadoop, HDFS ve MapReduce bileşenlerinden oluşan bir yazılımdır[2]. Apache Hadoop yazılım kütüphanesi kendine özel programlama yapılarını kullanır ve bilgisayar kümeleri arasında dağıtılan verileri dağıtık işlemeye izin verir. Bu yapı üzerinde veri barındırabilen ve veri işleme yapabilen bir yerel sunucudan yüzlerce sunucuya kadar arttırılabilir şekilde tasarlanmıştır. Yüksek maliyetli donanımlar yerine daha düşük seviyede ama sayı olarak fazla bilgisayarları aynı kümede yazılım aracılığı ile tutarak işlem yapma modeli benimsenmiştir. HDFS ve MapReduce olarak iki temel kısımdan oluşur. Hadoop teknolojisiyle geliştirilen projelere yardımcı olmak için hataları otomatik olarak yakalayabilecek şekilde gelişmiştir.

Hadoop büyük veri dosyalarının paralel şekilde işlenmesi için kullanılan MapReduce yöntemi açık kaynak kodlu geliştirilmiştir. Apache firmasının en çok desteklediği projelerden biri olan bu teknoloji java programlama dili kullanılarak geliştirilmiştir. Bilgisayar kümeleri üzerinde çalışan dağıtık programlar için alt yapı desteği sunmaktadır. Hadoop'un sunduğu diğer önemli bölümden biri de Hadoop Dağıtık Dosya Sistemi'dir (HDFS). Bu özellikten ilerleyen bölümlerde bahsedilmiştir.

Temel olarak Şekil 2.4'te görüldüğü gibi aşağıdaki modülleri içermektedir. İlerleyen bölümlerde detaylıca değinilecektir.

[2] http://devveri.com/hadoop-nedir



Şekil 2.4: Hadoop altyapısı.

- **Hadoop Common:** Diğer parçaları bir arada çalıştırmaya yarayan alt yapıdır.
- **Hadoop Distributed File System (HDFS):** Uygulama verilerine yüksek veri ile erişim sağlayan dağıtık dosya sistemidir.
- **Hadoop YARN:** İş planlaması (Job Scheduling) ve küme (Cluster) planlama için bir yazılım altyapısıdır (framework).
- **Hadoop MapReduce:** Büyük veri setlerini paralel işlemek için Hadoop YARN tabanlı bir sistemdir.

2.3.1.1 Hadoop Mimarisi

Hadoop mimarisi geliştirilme amacına uygun olarak verileri, sistem üzerinde güvenli, etkin ve ölçeklenebilir olarak tutmakta ve işlenmektedir. Verileri farklı düğümler üzerinde istenildiği kadar kopyasını dağıtarak tutmaktadır. Verileri dağıtık olarak bilgisayar düğümleri üzerinde tutmanın, veri kayıplarına sebebiyet verebileceği ihtimalini düşünerek yedek kopyası ile çalışmaktadır. Herhangi bir sebepten ötürü bir düğüme erişim kaybolur ise başka bir düğüm üzerinde olan sağlam bir kopyadan çalışma yapılır ve tekrar kopyalama yapar. Böylece verilerin doğru ve güvenli bir formda sistem üzerinde olması sağlanmış olur.

Hadoop paralel işleme mantığında çalışır. Ölçeklenebilir verileri çözümleme ve işlemeye elverişli bir sistemdir. Hadoop mimarisi temelde dosya sistemi barındırmaktadır. HDFS, Hadoop kümesinde verileri düğümler üzerinde kaydetmektedir. Dışarıdan bir istemci tarafından HDFS, bilinen diğer dosya sistemleri gibi yekpare olarak algılanmaktadır.

Yani dosya oluşturulması, silinmesi gibi diğer dosya sistemleri tarafından gerçekleştirilen işlemler yapılabilir. Mimari olarak kendi içerisinde HDFS özel düğümlere ayrılır. Bu düğümler;

- **NameNode:** HDFS içinde üst veri (metadata) bilgisinin barındırıldığı düğümdür. Hangi dosyanın hangi parçalarının ve kopyalarının hangi düğümler üzerinde olduğu bilgisi tutulmaktadır. Genellikle Master Node olarak adlandırılır.
- **DataNode:** HDFS'i oluşturan düğümler içinde verilerin tutulduğu düğümlerdir.

2.3.1.2 Hadoop Özellikleri

Hadoop ekosisteminin sunduğu kolaylıklar ve esneklikler barındırdığı özellikler sayesindedir. Bu özellikler aşağıdaki gibidir.

Veri İşleme Esnekliği: İşletmelerin son yıllarda karşılaştığı en büyük zorluklardan biri yapısal olmayan verilerin işlenmesi olarak öne çıkmıştır. Çünkü işletmelerin çoğunluğunda yapısal veriler tüm verilerin %20'sini oluşturmaktadır. Hadoop yapılandırılmış veya yapılandırılmamış, biçimlendirilmiş veya biçimlendirilmemiş verileri hatta başka bir veri türü olup olmadığına bakmaksızın verileri yönetir. Hadoop'un bu özelliği sayesinde yapılandırılmamış veriler, karar verme sürecinde yararlı olabilmektedir.

Kolay Ölçeklenebilme: Hadoop'un en değerli özelliklerindendir. Pahalı olmayan sunucular üzerinde paralel olarak çok büyük verileri dağıtması sayesinde kolay ölçeklenebilen bir platform olduğunu göstermektedir. İlişkisel veritabanı sistemleri (RDBMS)'nin aksine Hadoop terabyte'larca verileri işleyebilecek binlerce düğüm ile genişletilebilir.

Hataya Dayanıklı: Hadoop'ta veriler HDFS üzerinde iki farklı yerde tutulmaktadır. Bu iki farklı sunucunun çökmesi veya sistemden ayrılması durumunda en azından üçüncü bir yerde bir kopyası mevcuttur. Bu çalışma prensibi sayesinde hatalara dayanıklılığını yüksek seviyelere çıkarmaktadır. Ayrıca çoğaltma (replication) özelliği ayarlanabilmektedir ve bu güvenli bir depolama sistemi olduğu konusunda bir delildir.

Hızlı Veri İşleme: Geleneksel sistemlerde ETL (Extract-Transform-Load; "çıkart-dönüştür-yükle") ve toplu veri işleme adımları saatler, günler hatta haftalar almaktadır.

Bu işlemler için Hadoop üzerinde paralel işleme ve HDFS sayesinde 10 kata varan hızda daha hızlı veri işleyebilmektedir.

Hadoop Ekosisteminin Sağlamlığı: Hadoop'un çok geniş bir ekosistemi vardır. Öyle ki küçük veya çok büyük işletmelerin ihtiyaçlarını karşılayabilmektedir. Çok farklı analiz ve ihtiyaçları karşılayacak paket uygulamaları bünyesinde barındırmaktadır. MapReduce, Hive, Hbase, Zookeeper, HCatalog Apache Pig vb birçok yeni araçlar ve teknolojileri gibi projeler ile birlikte geliyor.

2.3.1.3 YARN (Yet Another Resource Negotiator)

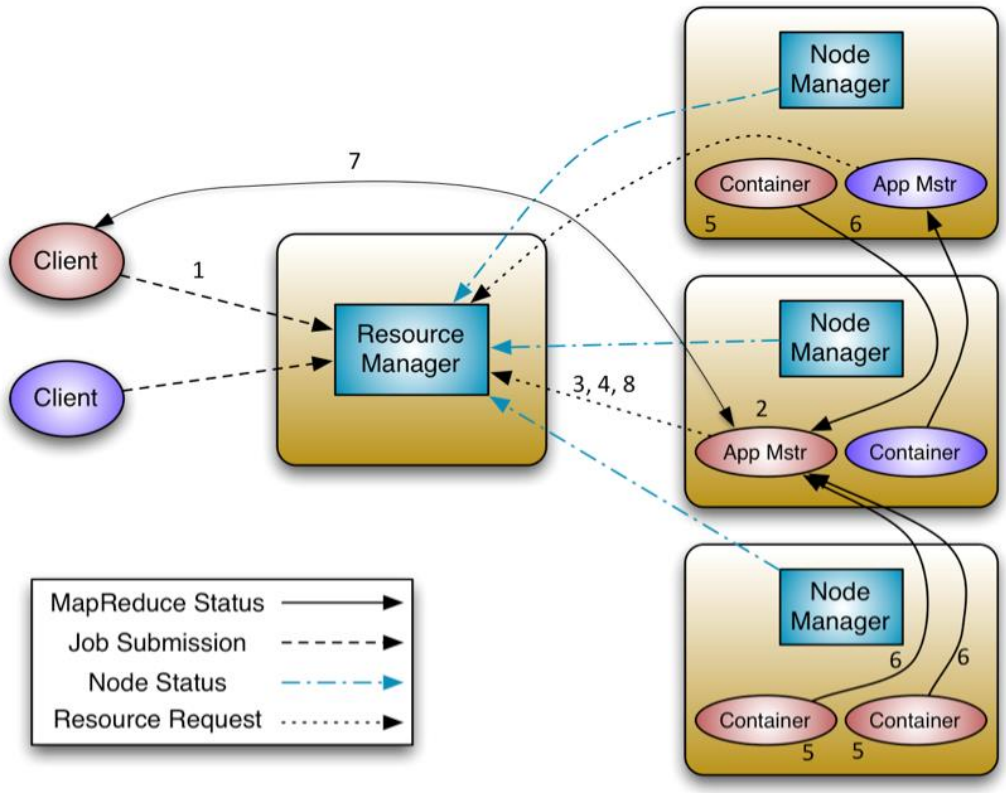
Hadoop tartışmasız olarak büyük veri işlemede en çok benimsenen alt yapı haline gelmiştir. Bunun temel nedeni ise bazı örnekleri kendine has özelliklerle çözme konusunda başarılı çalışmalardır. Büyük veri işlemlerinin önünde duran sorunlardan bir tanesi kaynak paylaşımıdır. Örneğin MapReduce 4000 ölçeklenebilir düğüm seviyesine ulaşmıştır[16]. Bu büyüklükteki kümelerde koordinasyon ve kaynak paylaşımı çok önemlidir. Ayrıca büyük verilerin işlenmesinde olmazsa olmaz olan en önemli özelliklerden bir tanesi ise çoklu hesaplama katmanları arasında kaynak paylaşımıdır. Bu konu o kadar önemli yer kaplamaktadır ki bilgisayar mühendisleri ve bilim adamları bu yapıyı geliştirmek için halen çalışmalara devam etmektedirler. Hadoop bu kaynak paylaşımını çözmek için YARN (Yet Another Resource Negotiator)'ı kullanmaktadır. Hadoop'un ilk versiyonlarında YARN bulunmamaktaydı ve 2.0 sürümü ile Hadoop ailesine eklenmiştir. YARN'ın eklenmesi Hadoop'un yeteneklerini geliştirmiştir[3]. YARN sadece MapReduce ile sınırlı değildir. YARN'ın temel mantığı JobTracker'ın işlevlerini iki parçaya bölmektir. Bu iki parça kaynak yönetimi ve iş planlaması/izlemesidir. Bu fikir çerçevesinde genel bir ResourceManager (RM) ve her uygulama için bir ApplicationMaster (AM) bulunmaktadır. Bir uygulama diğer birçok işlerin toplamından oluşmaktadır. Bunların yanında düğümlerde çalışan NodeManager hizmeti bulunmaktadır. Bu hizmet düğümler üzerinde çalışan uygulamalardan, düğümün kendi içinde kaynak yönetiminden sorumludur. Ayrıca NodeManager düğümlerde ki son durumları ResourceManager'a iletmekle yükümlüdür. Bir uygulama başlatıldığında, o uygulamaya özel bir ApplicationMaster işlevi başlatılır. Bu süreç ResourceManager'ın

[3] http://hortonworks.com/hadoop/yarn/

üzerindeki kaynakları alıp NodeManager üzerinde işin parçacıklarını yapacak olan alanları çalıştırır. Şekil 2.5'de YARN süreç akışı görünmektedir.

Resource Manager: Tam anlamıyla bir zamanlama uygulamasıdır. Tüm kaynakları her zaman kullanıma hazır tutar. Çeşitli kullanımlar için kaynakları optimize eder. Adil kapasite ve zaman kaynak kullanımı sağlanması için farklı algoritmalar içermektedir.

ApplicationMaster: Framework'e özel kütüphaneler yardımıyla ResourceManager'dan ayrılan kaynakların yönetiminden sorumludur. Ayrıca NodeManager ile çalışarak uygulamaların yürütülmesi, izlenmesi ve onların kaynak tüketimini yönetir.



Şekil 2.5: YARN süreç akışı.

Bir uygulamanın çalışma mantığı Şekil 2.5'de sayılarla belirtilen adımlar şeklindedir ve aşağıda bu adımlar sıralanmıştır[4].

1. Bir istemci tarafından herhangi bir uygulama çağırıldığında kendine gerekli olan bütün verileri çağırır. Bu işlemi ApplicationMaster'ın kendisi yapar.
2. ResourceManager, ApplicationMaster'ı çalıştıran ve sürdüren özel alanların birbirleriyle olan haberleşme ve ilişkilerinden sorumludur.
3. ApplicationMaster ResourceManager tarafından başlangıçta kayıt altına alınır. Bu kayıt işlemi istemci uygulamanın kendine ait ApplicationMaster'ına ResourceManager ile iletişim kanalı kurarak detaylarını sorgulamasını sağlıyor.
4. Normal operasyonlar sırasında ApplicationManager uygun kaynak alanlar ve kaynak istek protokolü (resource – request protocol) arasında ilişki kuruyor.
5. Başarılı alan tahsisi durumunda, ApplicationMaster NodeManager'a ait alan başlatma özelliklerini kullanarak alan tahsisini sağlar.
6. Uygulama kodlarının yürütülmesi sırasında uygulama alanlara gerekli bilgileri (süreç, durum v. b.) ApplicationManager'a uygulamaya özel protokolü kullanarak sunar.
7. Uygulamanın çalıştırılması alanlar içerisinde olmaktadır. Yanı sıra uygulamaya özel protokol ile ApplicationMaster'a gerekli bilgileri sağlar.
8. Uygulamanın çalışması tamamlandığında ve bütün gerekli işlemler bittiğinde ApplicationManager ResourceManager'dan kendi kaydını siler ve kapanır. Kendine tahsis edilen alanı serbest bırakır.

2.3.2. Hadoop Dağıtık Dosya Sistemi (HDFS)

Dosya sistemleri teknolojinin hızlı gelişimine ayak uyduramamaktadır. Sürekli depolama üniteleri ve dosya sistemlerindeki yavaş teknolojik gelişmeler diğer gelişmelere engel olmaktadır. Bu sorun sebebi ile geliştiriciler alternatif dosya sistemleri ve algoritmalar ile dosyalama sistemlerini geliştirmeye çalışmaktadır. Geliştirilen yöntemlerden biri de dağıtık dosya sistemidir.

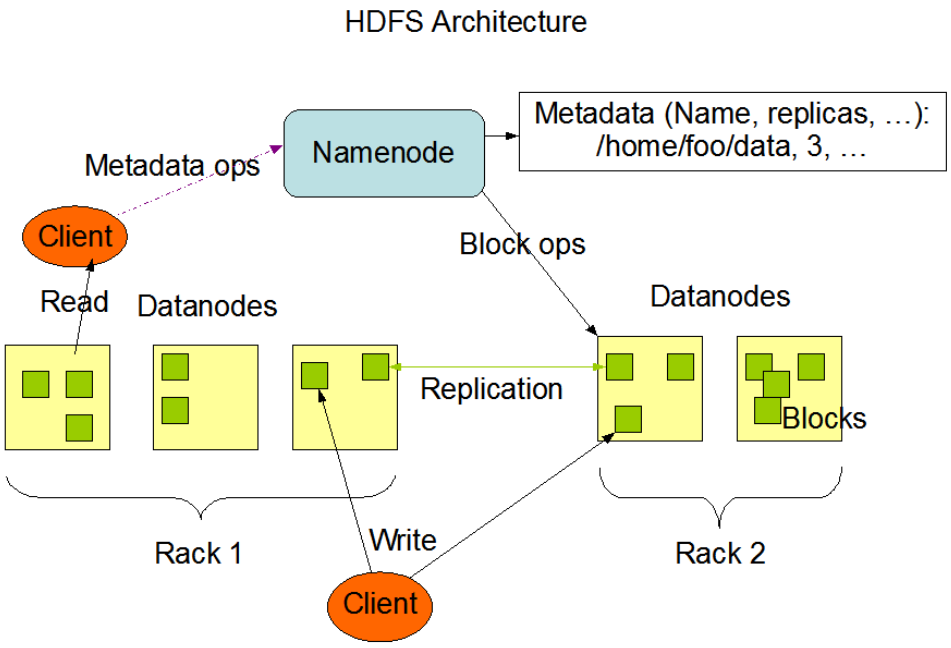
[4] http://hortonworks.com/blog/apache-hadoop-yarn-concepts-and-applications/

Bir dosya sistemi eğer birden fazla makine üzerinde verileri bir ağ üzerinden depolanması ve yönetilmesini sağlayabiliyorsa Dağıtık Dosya Sistemidir. Ağ üzerinde oluşabilecek sorunların karmaşıklığı sebebi ile dağıtık dosya sistemleri daha karmaşık ve zordur.

Hadoop dosya sistemlerinde haklı sebepler dolayısı ile dağıtık dosya sistemlerini geliştirmiştir. Hadoop yapısının temelinde dosya sistemi olarak HDFS kullanılmaktadır. HDFS, büyük boyuttaki dosyalar depolamak için sıradan sunucu kümelerini kullanmak üzere tasarlanmıştır[17].

HDFS sıradan bilgisayarların depolama alanlarını tek bir disk olarak kullanmaktadır. Dolayısı ile düşük maliyetli sunucular ile çalışma fırsatı sunmaktadır. Bu sayede veri boyutunun ne olduğu çok fazla anlam ifade etmemeye başlamıştır. Büyük boyutlu dosyalar bloklara (varsayılan 64MB) ayrılarak farklı sunucu üzerine (varsayılan 3 kopya) dağıtık olarak tutulurlar. Farklı kopyalar hata toleransı için tutulmaktadır. RAID yapısında yedekleme yapılır. Bu sayede veri kaybı önlenmiş olur. HDFS üzerinde çok büyük boyuttaki veriler için okuma işlemi yapılabilir ve paralel olarak bu işlem gerçekleşir. Fakat rastgele erişim (random access) mümkün değildir sadece sıralı okuma söz konusudur.

HDFS genellikle düşük maliyetli donanım üzerinde konuşlanmış olduğundan, sunucu arızalarının olması yaygındır. Dosya sistemi işlem düğümleri arasındaki hızlı veri transfer kolaylaştırmak ve bir düğüm başarısız olursa çalıştırmaya devam etmek Hadoop sistemleri tarafından sağlanmış ve son derece hataya dayanıklı olacak şekilde tasarlanmıştır. HDFS mimarisi Şekil 2.6’da gösterilmektedir.



Şekil 2.6: HDFS mimarisi.

2.3.2.1 NameNode

HDFS üzerindeki dosya ve dizinlerin hiyerarşisinin tutulduğu alandır. Dosya ve dizinler NameNode üzerinde inode'lar tarafından temsil edilir. inode kayıtları izinler, kayıt bilgileri, kayıt tarihleri ve disk kota bilgilerini barındırmaktadır. Dosya içerikleri büyük bloklara ayrılmışlardır. Genellikle bu blok boyutları 64 MB veya 128 MB olmakla beraber kullanıcı her bir dosyaya özel bu boyutu seçebilmektedir. Ayrıca dosyanın her bir bloğu bağımsız üç DataNode (Veri düğümü)'da tutulmaktadır. NameNode veri listelerini ve DataNode'lara eklenmiş blokların listelerini yönetir. Her bir küme (Cluster) için bir NameNode vardır. Binlerce düğümden oluşan kümelerde tüm istemcilere cevap verebilecek şekilde tasarlanıştır aynı zamanda tüm istemcilere cevap verebilecek durumdadır [18].

2.3.2.2 Görüntü (Image) ve Günlük Dosyası (Journal)

Sistemdeki tüm üst veri barındıran inode ve blokların listesi Görüntü (Image) olarak adlandırılır. NameNode tüm alandaki görüntüleri RAM'de tutar. Kalıcı görüntü kayıtları NameNode'un üzerindeki yerel dosya sisteminde Kontrol Noktası (CheckPoint) diye adlandırılan yerde tutar. HDFS üzerinde bir NameNode kaydının değişmesi gerektiğinde ilk önce yerel dosya sisteminde Günlük Dosya (Journal) çağırılır. İstemciler tarafından

başlatılan her istek günlük dosyası (Journal) üzerinde kaydedilir ve bildirime onay verilip istemciye gönderilmeden önce günlük dosya senkronize edilerek temizlenir. Kontrol noktası olarak tutulan dosya NameNode tarafından asla değiştirilmez. Yönetim tarafından istek yapıldığında yeniden başlatma aşamasında yeni bir kontrol noktası oluşturulur. Başlama aşamasında NameNode dosyasının görüntüsü kontrol noktasından yüklenir ve değişiklikler günlük dosyasından uygulanır. NameNode istemcilere hizmet etmeye başlamadan yeni kontrol noktası ve günlük dosyası temizlenerek yeniden depolamaya yazılır.

Geliştirilmiş dayanıklılık için kontrol noktası ve günlük dosyalarının gereksiz kopyaları yerel dosya sistemine bağımsız olarak kopyalanır. Öncelik olarak tek yığın üzerinde kaybı önler ikincil olarak hataya karşı tüm düğümleri korur. Eğer NameNode günlükleri yazmada herhangi bir hata algılar ise yığının bulunduğu dosyaları otomatik olarak sistemden çıkartır. Eğer hiçbir dosyalama dosyası sistemde olmaz ise NameNode otomatikman kapanır.

2.3.2.3 DataNode

DataNode üzerindeki her bir blok kopyası yerel dosya sisteminde iki dosya ile temsil edilir. Bu dosyalardan ilki verinin kendisini tutar. Bloğun ikinci dosyası ise veri ve zaman damgasının tümünün meta verisini tutar. Veri dosyasının boyutu bloğunun gerçek uzunluğuna eşittir ve geleneksel dosya sistemi gibi fazladan boşluk gerektirmez. Bir bloğun yarısı doluysa yerel dosya sisteminin dolu bir bloğunun yarısına ihtiyaç duyar. Başlatma sırasında DataNode her bir NameNode ile bağlantı kurar ve el sıkışır (Handshake). Bu el sıkışmanın nedeni DataNode sürümünü doğrulamak içindir. Eğer sürümler uyuşmuyor ise DataNode otomatik olarak kapanır. Dosya sistemi formatlandığında alan adı kimlik bilgisi atanır. Kimlik bilgisi kümenin tüm düğümlerinde depolanır. Bu sayede aynı alanların kümeye başka kimliklerle eklenmesi engellenerek dosya bütünlüğü korunur.

El sıkışma sonrasında DataNode, NameNode üzerinde kaydolur. DataNode kendi benzersiz kimliğini alır ve depolar. Bu kimlik bir daha hangi fiziksel ağ üzerinden sisteme dâhil olursa olsun değişmez. Belirli periyodlarda DataNode kendi üzerindeki blokların raporunu NameNode'a göndererek güncelliği sağlar. Bu rapor; blok kimliği, zaman damgası ve sunucu üzerindeki her bir kopyanın uzunluğundan oluşur. İlk blok raporu

Datanode'un kayıt olması ardından hemen gönderilir. Akabinde her saat başı blok raporları NameNode'a gönderilerek güncel kalması sağlanır. Normal çalışma esnasında DataNode'lar kalp atışı (Heartbeat) olarak adlandırılan sinyaller göndererek blok kopyalarının sunucu üzerinde var olduğunu bildirir. Bu sinyallerin aralığı normalde üç saniyedir. NameNode'a eğer 10 dakika içinde DataNode'dan kalp atışı gelmez ise ilgili DataNode servis dışına alınır. NameNode üzerinde ise servis dışına alınan DataNode'un yeni kopyasının diğer DataNode'larda oluşturulması için zamanlanır. Kalp atışı sinyalleri toplam depolama kapasiteleri, depolama kısmı ve devam eden veri transferi sayısını bildirir. Bu bilgiler NameNode''un blok tahsisi ve yük dengeleme kararı için kullanılır.

NameNode DataNode'lara direk olarak istek göndermez. Bunun yerine kalp atışlarına verilen yanıtları talimat olarak kullanır. Bu talimatlar blok replikasyonu, yerel blokların silinmesi, yeniden kayıt, acil blok raporu, düğümün kapatılması olabilir. Bu komutlar sistemin bütünlüğünün korunması için önemlidir ve bu sebepten büyük kümlerde sık kalp atışları olur ve bunlar büyük öneme sahiptir. Bu sebepten sistem tasarımında NameNode diğer işlevlerini aksatmadan binlerce kalp atışını işleyebilecek yapıda kurgulanmıştır.[5]

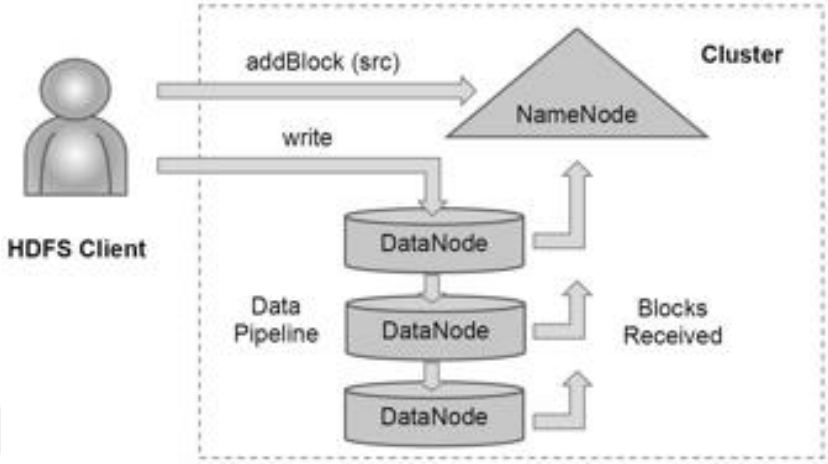
2.3.2.4 HDFS İstemci

Kullanıcı uygulamaları dosya sistemi için bir kütüphane sunan HDFS istemcisi kullanarak bağlanabilmektedir. Geleneksel dosya sistemleri gibi HDFS'te silme, yazma, okuma ve klasör işlemlerini desteklemektedir. İsim alanı içerisindeki referans bilgileri ile dosya yolları ve dosyalara erişim sağlanmaktadır. Kullanıcı uygulaması dosya sistemi üzerindeki üst veri ve yedeklerini bilmek zorunda değildir.

Bir uygulama bir dosyayı okuduğunda, HDFS istemcisi ilk olarak NameNode'a verileri barındıran DataNode üzerindeki blokların listesini sorar. Bu liste ağ topolojisinde uzaklığa göre sıralı bir şekildedir. Sonrasında istemci DataNode ile doğrudan iletişim kurarak gerekli blokların transferi için istekte bulunur. Bir istemci yazma işlemi yaptığında, ilk dosyanın ilk bloğunun kopyalanacağı Veri düğümünü seçmek için NameNode'a sorar. İstemci düğümler arasındaki hattı düzenler ve yazma işlemine başlar. İlk blok tamamlandığında sıradaki bloğun nereye yazılacağını sorar. Ve işlem bu şekilde

[5] http://www.aosabook.org/en/hdfs.html

devam eder. Her bir veri bloğunun farklı DataNode üzerinde tutulması muhtemeldir. İstemci, NameNode ve DataNodes arasındaki etkileşimler Şekil 2.7'te gösterilmiştir.



Şekil 2.7: İstemci, NameNode ve DataNodes arasındaki iletişim.

2.3.2.5 Kontrol Noktası Düğümü (Checkpoint Node)

HDFS içerisinde NameNode birincil rolü olan istemci isteklerine hizmet etmeye ek olarak alternatif bir Kontrol Noktası Düğümü ya da Yedek Düğümü (BackupNode) olarak adlandırılan rollerin atamasını yapar. Bu rol ataması düğümün başlangıcında belirtilir.

Kontrol noktası düğümü periyodik olarak var olan günlük dosyaları ve kontrol noktalarını toplar bu bilgilerden yeni kontrol noktası oluşturarak günlük dosyalarını boşaltır. Genellikle kontrol noktası düğümü NameNode'dan farklı bir ana bilgisayarda çalışır çünkü bellek iki görevinde ciddi bellek ihtiyacı vardır. Bu makine kontrol noktası ve günlük dosyalarını NameNode'dan indirerek kendi üzerinde birleştirir ve yeni kontrol noktasını geri yükler. Bu işlem dosya sisteminin üst verisini korur.

2.3.3. MapReduce

MapReduce dağıtık mimaride tutulan verilerin paralel olarak çalışıp işlenmesini ve analiz edilmesini sağlayan bir yazılım altyapısı (Framework)'dır[6]. Sistem 1960'lı yıllarda geliştirilen fonksiyonel programlamadaki map ve reduce fonksiyonlarından esinlenmiştir. Veriler işlenirken temel olarak bu iki fonksiyon kullanmakla beraber her bir aşamada ek olarak bazı fonksiyonlarda mevcuttur. Her ne kadar ilk ortaya çıkma zamanı tartışılırsa

[6] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

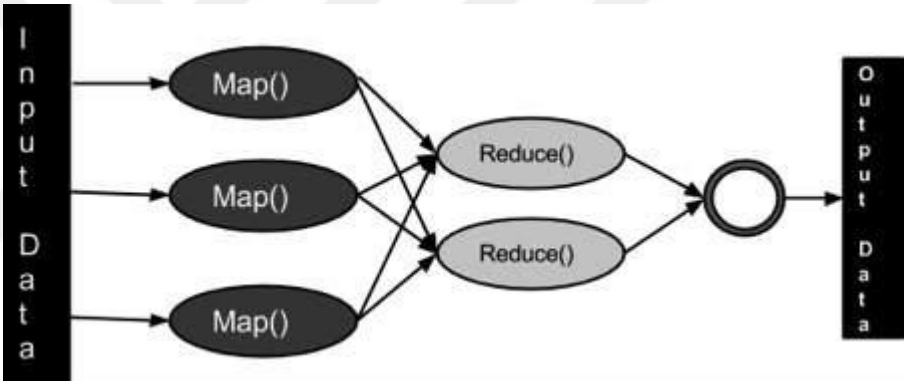
da 2004 yılında Jeffrey Dean ve Sanjay Ghemawat tarafından yayınlanan “MapReduce: Simplified Data Processing on Large Clusters” adlı makale ile tanınmaya başlanmıştır. Bu makalede Google’ın büyük verileri nasıl parçalara ayırdığı ve işlediği anlatılmaktadır[19]. Bu makaleden sonra Nutch isimli açık kaynak kodlu proje ile MapReduce uyarlama çalışmaları yapılmıştır. Zamanla yaygınlaşmış ve özellikle Yahoo’nun yatırımları ile gelişmiş ve zamanla ayrılarak Apache Fonu tarafından geliştirilmeye başlanmıştır. Hadoop’un açık kaynak kodlu olmasının avantajlarından biride geliştiricilerin sürekli sistemi geliştirerek farklı projeler ile MapReduce yöntemi üzerinde çalışma yapmalarıdır. Bu projelere örnek olarak Pig, Hive ve Hbase örnek verilebilir.

Google sunucularında kayıtlı verilerin paralel işlenmesi amacıyla geliştirilmiştir. Günlük olarak kaydedilen web sayfalarının kütüğe yazılması sonucu oluşan veri kümesi içerisinde çok kısa sürede sonuç üretecek şekilde arama yapmak için geliştirilmiştir. Binlerce paralel çalışan bilgisayarı kullanabilecek şekilde tasarlanmıştır. Bu noktada dağıtık sistemlerdeki verinin parçalanması, dağıtılması ve işlemin paralel işlemesi gibi problemler ortaya çıkmıştır. Bu problemleri aşabilmek için yük dengeleme (load balancing), verinin dağıtılması ve hata toleransı işlemleri için bir soyutlama modeli tasarladırlar. Geliştirdikleri MapReduce yöntemi bu nedenlerle kullanıcı tarafından kolayca kullanılabilir ve dağıtık sistem üzerinde kolayca işlem yapmaya izin veren bir ara yüze sahiptir. Basit ve güçlü olan bu sistem sayesinde kullanıcı sadece map ve reduce fonksiyonlarını yazarak paralel işlemesini istediği işi tanımlar, sistem bu iş için gerekli veri bölümleme, paralel işleme ve hata yönetimini kendi yönetir. MapReduce kullanarak paralel işlenen farklı uygulamalar mevcuttur. Bunlara örnek olarak web sayfası erişim frekansı tarama ve bulma, arama motorlarında kelime veya dosya arama, büyük boyutlu imgeler üzerinde işlem yapmak, Dağıtık olarak sıralama algoritmaları çalıştırmak ve Paralel görüntü işlemek gibi paralel hesaplama gerektiren bazı işlemleri verebiliriz.

MapReduce modeli daha sonra Apache yazılım topluluğu tarafından Hadoop adında açık kaynak kodlu bir proje olarak geliştirilmiştir. Hadoop ücretsiz bir sistem olduğundan çok geniş bir geliştirici topluluğu tarafından kullanılmaktadır. Hadoop kullanıcıları arasında Yahoo gibi büyük kurumlar da mevcuttur.

Hadoop, MapReduce mimarisi temelinde tasarlanmıştır. Hem MapReduce hem de HDFS dosya sisteminin kullanımı ile birlikte düğüm hataları otomatik olarak düzeltilir ve kullanıcıya yüksek güvenlikli bir sistem sunulur. MapReduce mimarisi sayesinde olası donanım problemlerinde kullanıcı desteği olmadan kurtarma işlemleri yapılabilir. Bilgisayar kümeleri üzerinde çalışan dağıtık programlar için alt yapı desteği sunmaktadır. Kullanıcı etkileşimi gerekmeksizin verilerin ilgili düğüme taşınmasını ve olası bir donanım sorununda kurtarma işlemlerini otomatik gerçekleştirmektedir.

MapReduce çok uzun süren sorgular veya analizlerin büyük ölçekli veriler üzerinde toplu olarak işletilmesi için ideal bir yöntemdir. Birçok bilgisayarın bir arada küme olarak çalışabilmesini sağlayan mimariye MapReduce mimarisi denir. MapReduce veri akışı Şekil 2.8'de gösterilmektedir[7].



Şekil 2.8: MapReduce veri akışı.

2.3.3.1 MapReduce Algoritması

Hadoop MapReduce işleri (Job) iki parçaya bölünmektedir, bu parçalar map ve reduce olarak adlandırılan verilerin ayrıştırılması görevi ve indirgeme görevidir. Her iki görevde dağıtık yapıdaki bilgisayar kümeleri üzerinde çalışacak şekilde tasarlanmıştır. Tüm görevler küçük veri alt kümeleri ile çalışır bu sayede HDFS üzerinde dağıtık olarak tutulan veriler üzerinde çalışma sağlanarak toplam iş yükü dağıtılır. Map görevleri genellikle verileri yüklemek, ayrıştırmak, dönüştürmek ve filtrelemek ile görevlidir. Her reduce görevi ise map görevinin çıktılarının alt kümelerinden birini işlemekle

[7] http://www.slideshare.net/martyhall/hadoop-tutorial-mapreduce-part-4-input-and-output

yükümlüdür. Daha sonra veriler reduce görevi tarafından gruplama, matematiksel işlemler yapılır.

Hadoop'ta map fonksiyonu tanımlayabilmek için Mapper ara yüz sınıfını işletmek gerekmektedir. Aşağıdaki map fonksiyonu Şekil 2.9'da görüldüğü gibi MapReduceBase sınıfından türeyen ve Mapper ara yüz sınıfını işleten bir sınıf yazılır. Burada dört tane parametre ile işletilen Mapper ara yüzü için sırasıyla girdi anahtar formatı, girdi değer formatı, çıktı anahtar formatı ve çıktı değer formatı şeklindedir. Örnekte bunlar "<LongWritable, Text, Text, IntWritable>" şeklinde tanımlanmıştır. Bu parametre sınıfları Hadoop içinde tanımlanmış sınıflardır ve org.apache.hadoop.io paketinden diğer formatlara da ulaşılabilir.

```
public class LineCountMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    public  static  String SPLITTER=";";
    @Override
    public void map(Object key, Text value, Context output) throws IOException,
            InterruptedException {

        String[] liste = value.toString().split(SPLITTER);
        for (int i=0;i<liste.length;i++) {
        output.write(new Text(liste[i]), one);
        }
    }
}
```

Şekil 2.9: Örnek Mapper sınıfı.

Reduce fonksiyonu, MapReduceBase sınıfından türeyen ve Reducer ara yüzünü işleten bir Reducer sınıfı içerisinde yer alır. Map işleyicinin oluşturduğu ortanca çıktı kayıtları gruplanarak reduce fonksiyonuna verilir. Reduce fonksiyonu gelen kayıtları en son işleme tabi tutarak OutputCollector sınıfına ait "collect" metodu ile çıktı olarak HDFS'ye kaydeder. Şekil 2.10'da sunulan örnek reduce fonksiyonu kodunda map fonksiyonundan alınan çıktı kayıtlarındaki kelime değerleri bulunarak çıktı dosyasına yazılmıştır. Bulunan her bir kelime için bir değeri "value" olacak ve kelime de "key" olacak şekilde elde edilen kayıt verileri OutputCollector sınıfı yardımıyla çıktı olarak yazılır. Normalde map fonksiyonu çıktısı ile oluşan ortanca çıktı (intermediate output) olup tüm görevin nihai çıktısı değildir. Reducer'lardan elde edilen çıktı ise direkt olarak HDFS'ye kaydedilir ve nihai çıktıdır. Hadoop çıktısı reduce görev sayısı ile doğru orantılıdır.

```
public class LineCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

	@Override
	public void reduce(Text key, Iterable<IntWritable> values, Context output)
			throws IOException, InterruptedException {
		int Count = 0;
		for(IntWritable value: values){
			Count++;
		}
		output.write(key, new IntWritable(Count));
	}
```

Şekil 2.10: Örnek Reducer sınıfı.

Algoritmayı bir örnek üzerinden anlatmak konuyu daha basit ve anlaşılır kılacaktır. Örneğin elimizde Tablo 2.2'de görüldüğü gibi müşterilere yapılan satış miktarları dosyası olduğunu varsayalım ve MapReduce modeli ile hangi müşteriye toplam kaç adet ürün satıldığını bulmak istiyoruz.

Tablo 2.2: Müşteri satış miktarlarını gösteren tablo.

Müşteri	Ürün	Miktar (Adet)
Müşteri1	Ürün1	5
Müşteri2	Ürün3	3
Müşteri4	Ürün4	2
Müşteri2	Ürün1	1
Müşteri3	Ürün4	1
Müşteri4	Ürün3	4
Müşteri1	Ürün2	2
Müşteri4	Ürün1	7

Tablo 2.2'deki verilerin HDFS üzerinde bir dosyada tutulduğunu her bir satırdaki verilerinde "," ile ayrıldığını varsayarsak, map fonksiyonuna sırası ile aşağıdaki değerler gelecektir.

Müşteri1,Ürün1,5
Müşteri2,Ürün3,3
Müşteri4,Ürün4,2
Müşteri2,Ürün1,1
Müşteri3,Ürün4,1
Müşteri4,Ürün3,4
Müşteri1,Ürün2,2
Müşteri4,Ürün1,7

Record Reader sırası ile her sırası gelen satırı okuyacaktır. Recor Reader fonsiyonu dosyadan okuduğu satırın offset (ilgili satırın dosyadaki yeri) değerini Anahtar ve satırdaki veriyi değer olarak Mapper fonksiyonuna gönderecektir. Mapper fonksiyonu ise geliştiricilerin belirleyeceği (Bu örnekte biz belirleyeceğiz) yapıda anahtar, değer eşleştirmesi yapmaktadır. Bu örnekte her bir müşteriye satılan miktarı bulmak istiyorsak anahtar alanı müşteri bilgisi olmalıdır ve değer alanıda satış miktarı olmalıdır. Bu örnekte combiner kullanılmayacaktır dolayısı ile map görevinin son adımında bölümleyici verileri sırası ile reducer fonksiyonuna aşağıdaki formatta gönderecektir.

Müşteri1,5
Müşteri2,3
Müşteri4,2
Müşteri2,1
Müşteri3,1
Müşteri4,4
Müşteri1,2
Müşteri4,7

Reduce görevine gelen veriler ilk olarak Shuffle ve Sort adımında anahtar değerine göre sıralanır ve aşağıdaki şekilde bir çıktı üretir.

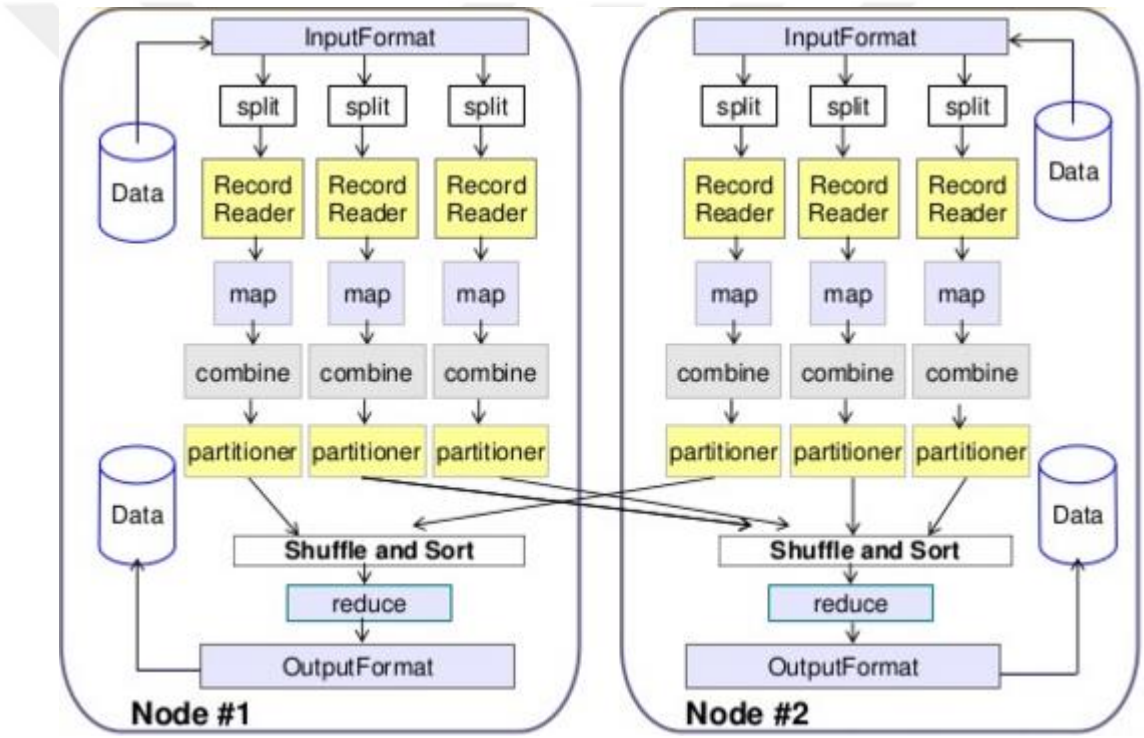
Müşteri1,2
Müşteri1,5
Müşteri2,1
Müşteri2,3
Müşteri3,1
Müşteri4,2
Müşteri4,4
Müşteri4,7

Reduce görevinin son adımında reducer aşamasında yukarıdaki gibi olan veriler kullanıcının belirleyeceği analitik işlemler (aggregation) yapılacaktır. Biz örnekte satış adetlerinin toplamı alınacağından, yapılacak işlem müşteri temelinde toplama yapmaktır. Reducer adımından sonra aşağıdaki çıktıya ulaşılır ve Output Format ile çıktı dosyası HDFS'e yazılır.

Müşteri1,7
Müşteri2,4
Müşteri3,1
Müşteri4,13

2.3.3.2 Map ve Reduce Görevleri

MapReduce temelde bir iş emiri ile temsil edilir. Bu iş emiri ise iki göreve ayrılır bunlardan birincisi map diğeri ise reduce görevleridir. Bu görevlerde kendi içlerinde map görevi 4 faza, reduce görevi ise 3 faza ayrılır. Şekil 2.11'de MapReduce fazları görülmektedir. Detaylı açıklamaya aşağıda yer verilmiştir.



Şekil 2.11: Map ve Reduce aşamaları.

Hadoop üzerinde map görevleri 4 faza ayrılır: record reader (kayıt okuyucu), mapper (eşleştirici), combiner (birleştirici) ve partitioner (bölümleyici). Map görevinin çıktısı olan Key (anahtar), Value (değer) olarak adlandırılan eşleştirmeler reducer (indirgeyici)'e gönderilir.

Record Reader: Girdi formatında gelen bölümlenmiş verileri kayıt haline dönüştürür. Recor reader'ın amacı gelen veriyi kayıtlara dönüştürmektir fakat kayıtları ayrıştırmaz. Gelen veriden elde ettiği gerekli kayıtları key, value halinde eşleştirerek mapper'a gönderir. Genellikle key özellik ve yer bilgisini içerir ve value ise veri yığınının kendisidir. Bu okunan veriler sırası ile mapper'a gönderilir.

Maper: Recor reader'dan gelen veri yığını ve verinin yer bilgilerini alarak eşleştiricide kullanıcı tarafından yazılan kodlar çalıştırılarak gelen veri üzerinde key/value eşleştirmeleri oluşturulur. Bu aşamada neyin anahtar neyin değer olacağı çok önemli değildir. Mapper kullanıcının tanımladığı kodlara göre eşleştirme yapar.

Combiner: Birleştirici yerel olarak çalışan bir indirgemedir. Kullanıcının belirlediği yapıdaki mapper'dan gelen veriler üzerinde yerel bir indirgeme yapar. Reduce ile arasındaki temel fark combiner çıktı olarak reducer'a girdi üretir fakat reducer çıktı olarak diske yazar. Birleştiricinin kullanıldığı birçok durumda veriler ağ üzerinde taşınmak durumunda kalır. Birçok geliştirici birleştiricileri göz ardı eder fakat genellikle birleştiriciler ciddi derecede performans kazancı sağlar.

Partitioner: Bölümleyici mapper veya eğer var ise combinerdan key/value eşleşmesini alır ve onları tekli parçalar halinde reducer'a gönderir. Genellikle bölümleyici nesneyi reduce görev sayısı kadar parçaya hash kodları ile ayırır. Bu parçaları reducer'lar üzerinde rastgele dağıtır ama aynı değere sahip olan farklı mapper'lardan gelen anahtarları aynı reducer'a gönderir. Bölümleyicilerin varsayılan özelliği farklı tanımlarla değiştirilebilir fakat değiştirmek çok nadiren gereken bir durumdur. Map görevi aşamasının son adımını oluşturan bu kısımda bölümleyiciler verileri yerel dosya sistemine yazar ve bunların ilgili reducer tarafından çekilmesini bekler.

Reduce görevi de 3 faza ayrılır: shuffle (karıştırma) ve sort (sıralama), reducer (indirgeyici) ve output format (çıktı formatı). Verilerin bulunduğu düğümler üzerinde map görevlerinin çalıştırılası isteğe bağlıdır. Fakat genellikle verilerin bulunduğu düğüm üzerinde işlem yapılır. Bu yöntem ile verilerin ağ üzerinde taşıması önlenerek işlemlerin daha hızlı yapılması sağlanır.

Shuffle ve Sort: Reduce görevi karıştırma ve sıralama adımı ile başlar. Bu adım map görevinin son adımını olan bölümleyicinin çıktısını alır ve reduce görevinin yürütüleceği

makinaya yani kendi üstüne indirir. Bu ayrışık veri parçaları anahtara göre sıralanır. Bu sıralamanın amacı aynı anahtarları kolay bir tekrarlamalı adımla reduce görevini yerine getirmektir. Bu adım özelleştirilemez ve alt yapı bu adımı otomatik yerine getirmektedir. Özelleştirilebilen kısım sadece anahtarların nasıl sıralanacağını belirten karşılaştırma (comparator) fonksiyonudur.

Reduce: İndirgeme fonksiyonu gruplanmış verileri alır ve her bir anahtar gurubu için bir defa çalışır. Bu fonksiyona anahtar ve bu anahtar ile alakalı tüm değer listesi tekrarlamalı olarak iletilir. Veriler bu adımda filtrelenebilir, birleştirilebilir veya aritmetik işlemler yapılabilir. İndirgeme işlemi bittiğinde eğer var ise key/value eşleşmeleri final adımı olan Output Format'a gönderilir.

Output Format: çıktı biçimi key/value eşleştirmesinin nihai halini alır ve bir dosyaya kayıt yazıcı yardımı ile yazar. Varsayılan olarak yeni satır karakteri ile key/value eşleşmelerini ayırır. Farklı çıktı formatları için kodlamada istenilen değişiklik yapılmasına izin verilmektedir fakat format ne olursa olsun sonuç olarak çıktı HDFS'e yazılacaktır.

3. MALZEME VE YÖNTEM

Bu çalışmada kullanılan sunucular bir fiziksel bir sunucu üzerinde 8 sanal makinadan oluşmaktadır. Bu sanallaştırma alt yapısında VMware vSphere5 Hypervisor kullanılmıştır. Fiziksel sunucu bilgileri aşağıdaki gibidir.

- Üretici: IBM
- Model: IBM System x3650
- CPU Core: 16 CPU x 2,493 GHz
- İşlemci Tipi: Intel (R) Xeon (R) CPU E5420 @ 2.50 GHz
- İşlemci Soket Sayısı: 4
- Her soketteki Core Sayısı: 4
- Ram: 24 GB
- Data Store: 2 TB

Yukarıda belirtilen özelliklere sahip bir sunucu üzerinde 1 Master ve 7 tane de slave sunucu çalıştırabilmek için sanallaştırma teknolojilerinden yararlanılmıştır. Sanallaştırma teknolojisi kullanılarak bir donanım ile sadece bir işletim sisteminin çalışması zorunluluğunun önüne geçilmiş olmaktadır. Sanallaştırma ile yazılım olan işletim sistemini ve uygulamaları fiziksel yapıdan ayrılmaktadır. Geleneksel yapıdan ziyade daha düşük maliyetli kümeler kurulmasına olan sağlamaktadır.

Sunucu sanallaştırması, günümüz sunucularının potansiyelini açığa çıkarıyor. Verimsiz kullanılan veya atıl kalan fiziksel kaynakların kullanımı için bu yöntem tercih edilmiştir. Oluşturulacak kümede karmaşıklığın önüne geçilmiş olmaktadır[8]. Bu kapsamda VMware vSphere, eksiksiz bir sunucu sanallaştırma platformu sunmaktadır.

Bu sanal ortamda yukarıda da bahsedildiği gibi 8 sanal sunucu oluşturulmuş ve kurulum için Hadoop sistemlerinden Hortonworks kullanılmıştır. Hortonworks HDP 2.3 ve neden tercih edildiği ilerleyen bölümlerde detaylıca ele alınacaktır.

Hadoop kümesi oluşturmak için kullanılan Hortonworks'un HDP platformu master – slave yapısını desteklediği için tamamen açık kaynak kodlu ve çalışma aşamasında en

[8] http://www.vmware.com/tr/virtualization

güncel ve kararlı sürümü olan HDP 2.3 için hazırlanmış Hortonworks Sandbox kullanılmıştır. Hortonworks Sandbox taşınabilir Apache Hadoop ortamıdır ve Hadoop ekosistemine ait onlarca özellikle gelir[9]. Kurulumu kolaydır ve birçok işlem için Ambari kullanılarak kullanıcı deneyimi kolaylaştırılmıştır. Master – Slave yapısında Master düğüm olarak Hortonworks Sandbox ile HDP 2.3 yüklenmiştir. Diğer 7 slave düğüm için Hortonworks Sandbox ile kullanılması ve aynı işletim sistemi ile daha performanslı çalıştığı için Linux CentOS 6.7 versiyonu tercih edilmiştir.

CentOS işletim sistemi Red Hat dağıtımına sahiptir. Red Hat Enterprise Linux (RHEL) kaynak kodlarını temel alan bir sistemdir. Bağımsız bir grup tarafından geliştirilmekteyken yakın bir zamanda geliştiricilerinin lideri RedHat ekibine katıldığını açıklamıştır. İşletim sistemi adını The Community ENTerprise Operating System kelimelerinden almaktadır. Linux tabanında geliştirilen işletim sistemidir[10]. Bireysel kullanımdan çok sunucu işlemleri için geliştirilmiştir.

Master düğümü için ayrılan kaynak bilgileri aşağıdaki gibidir.

- 8 Core CPU
- 10 GB RAM

Slave düğümler için ayrılan kaynak bilgileri her biri için aşağıdaki gibidir.

- 4 Core CPU
- 4 GB RAM

Yukarıda belirtilen özelliklerde sunucuların kurulum işlemlerinden sonra Hadoop kümesi oluşturma işlemlerine başlanmıştır. Hadoop kümesi kurulumu Ambari Server aracılığı ile yapılacaktır. Bu kurulum için öncelikle her slave sunucuda bir dizi işlem yapılması gerekmektedir. Bu işlemler farklı işletim sistemlerine göre küçük farklılıklar gösterebilmektedir. Bu yüzden bu çalışmada CentOS için anlatılacaktır.

Şifresiz Erişim Ayarları: Ambari Sunucusu otomatik olarak tüm kümedeki sunuculara ana sunucudan erişebilmek için Ambari Agent yüklemek zorundadır. Kurulum ve daha

[9] http://hortonworks.com/products/sandbox/
[10] https://www.centos.org/about/

sonrasında erişim için şifresiz SSH erişim kurulması gerekmektedir. Bu işlem için sırası ile aşağıdaki adımlar izlenmelidir. Bu adımlar aksi belirtilmedikçe Master düğümde komut satırında yapılmalıdır.

- Ambari Sunucuda (Master düğüm) yerel ve genel SSH anahtarı üretilmesi için Komut satırına "ssh-keygen" komutu yazılır. Bu dosya yolu /root/.ssh şeklindedir.
- Üretilen dosyalar "id_rsa" ve "id_rsa.pub" slave sunucularda aynı yola kaydedilir.
- Slave sunucularda yine aynı dosya yolunda bulunan veya oluşacak olan "authorized_key" dosyasına bu anahtar değerinin kaydedilmesi "cat id_rsa.pub >> authorized_keys" komutu yardımıyla yapılır.
- Dosya yoluna ve dosyaya erişim vermek amacı ile sırası ile "chmod 700 ~/.ssh" ve "chmod 600 ~/.ssh/authorized_keys" komutları çalıştırılır.
- Bu erişim yetkilendirmesinin geçerli olup olmadığı master sunucuda "ssh root@<remote.target.host>" komutunu çalıştırılması ve olumlu bir sonuç alınması ile tamamlanmış olur.

NTP'nin aktive edilmesi: Tüm düğümlerin birbirleri ile senkronize olabilmeleri için saat ayarları aynı olması gerekmektedir. Bunun içinde NTP'nin aktive edilmesi gerekmektedir. Bu işlem için önce "chkconfig --list ntpd" komutu ile her bir sunucuda NTP'nin açılışta otomatik çalışıp çalışmadığına bakılır. Ardından "service ntpd start" komutu ile hizmet çalıştırılır.

Host dosyasının düzenlenmesi: Her bir sunucuda /etc/hosts dosyasına tüm kümedeki düğümlerin hostname bilgisi girilmelidir. Ayrıca aşağıdaki satırlarında Hadoop ekosisteminde ve sunucuda çalışan hizmetlerin sağlıklı çalışması için dosyada bulunması gerekmektedir.

127.0.0.1 localhost.localdomain localhost

::1 localhost6.localdomain6 localhost6

Son olarak komut satırında "hostname <sunucu adı>" komutu çalıştırılarak sunucun adı atanır.

Ağ ayarlarının yapılması: /etc/sysconfig/network dosyasında aşağıdaki gibi değerler atanır.

NETWORKING=yes

HOSTNAME=<sunucu adı>

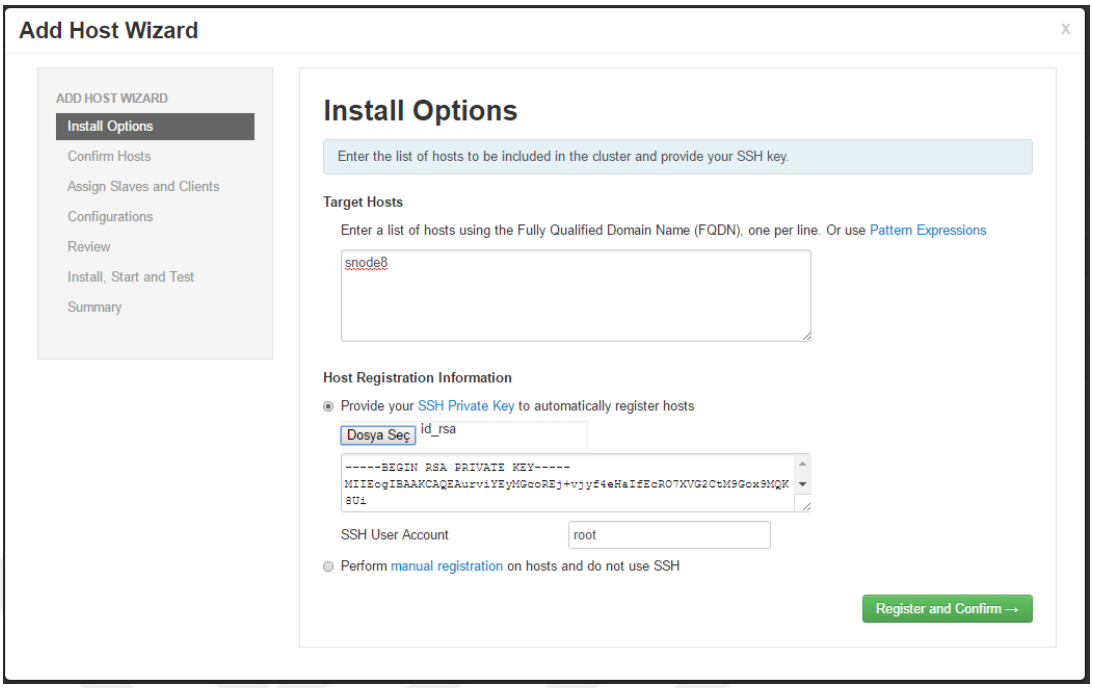
Firewall'ın kapatılması: Aşağıdaki komutlar ile firewall kapatılarak sunucuda portlar kullanıldığında herhangi bir engellemeyle karşılaşılmasının önüne geçilir.

chkconfig iptables off

/etc/init.d/iptables stop

SELinux (Zorunlu Erişim Denetimi)'un Kapatılması : /etc/selinux/config dosyasında SELINUX değerinin karşılığı "disabled" olarak atanır. Daha sonra /etc/yum/pluginconf.d/refresh-packagekit.conf dosyasında enabled değerinin karşılığı "0" olarak atanır.

Bu işlemler yapıldıktan sonra <master>:8080 adresinden Ambari Server arayüzüne erişilir. Bu arayüzden istenilen sunucular kümeye düğüm olarak dâhil edilebilir. Sunucu eklemek için Hosts sekmesinde Action butonuna basılarak Add Host butonu yardımı ile Şekil 3.1'deki ekrana ulaşılır.



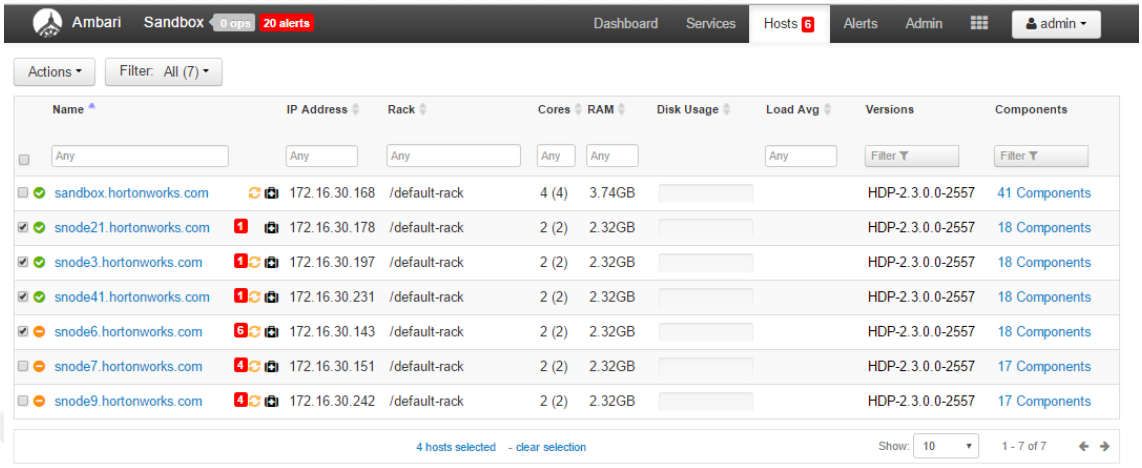
Şekil 3.1: Ambari server düğüm ekleme ekranı.

Bu ekranda öncelikle eklenecek sunucu adları belirtilir ve yukarıdaki adımlarda oluşturulmuş olan SSH erişim anahtar değeri girilerek bir dizi işlemden sonra "Assign Slaves And Clients" adımına gelinir. Bu adımda hangi eklenti ve uygulamaların kurulması gerektiği belirtilir. Biz bu çalışmada her bir slave sunucu için aşağıdaki bileşenleri seçmek tercih edilmiştir.

- DataNode
- Metrics Monitor
- NFSGateway
- NodeManager
- Clients

Bu tercihin başlıca nedenleri NameNode ve NFSGateway kurarak sorunsuz olarak HDFS sisteminden yararlanmak amaçlanmıştır. Aynı zamanda verileri düğümler üzerinde dağıtabilmek için bu tercihte bulunulmuştur. Düğümleri izleyebilmek ve ölçümlemek adına Metrics Monitor bileşeni yüklenmiştir. YARN ve dağıtık olarak görevlerin yürütülmesi için NodeManager yüklenmiştir. Son olarak da MapReduce veya diğer görevleri yürütebilmek için Clients yüklemesi yapılmıştır.

Bu adımlardan sonra eğer kurulu başarılı olur ise düğümler Şekil 3.2' de görüldüğü gibi oluşmuş olacaktır.



Şekil 3.2: Ambari sunucu listesi (Kümeyi oluşturan düğümler).

Küme kurulum işlemleri bittikten sonra artık Hadoop sistemi kurulum amacına hizmet etmeye hazırıdır. Bu çalışmanın amacı büyük verilerde Hadoop ve MapReduce teknolojinin incelenmesidir. Bu araştırma ve incelemeyi yaparken hızlı tüketim sektöründeki verileri temel alarak analiz yapıldı. Bu veriler genel olarak satış verilerini içermektedir. Bu veriler üzerinde birden fazla yöntem ile analizler yapılmıştır.

Çalışmada yapmış olduğumuz en temel analiz ise Pazar Sepeti Analizi (Market Basket Analysis)'dir. Bu analiz müşterilerin yapmış olduğu satın alım faturalarını inceleyerek, hangi ürünleri hangi ürünlerle tercih ettiği ve eğilimlerini gösterir. Müşterinin satın alma eğilimleri incelenip ortaya çıkarılarak kullanıma sunulması Pazar Sepeti Analizi olarak adlandırılır. Veri madenciliğinde en çok kullanılan yöntemlerden birisidir.

Müşterilerin almaya eğilimli oldukları ürünlerin belirlenmesinde önemli rol oynayan bir çözümleme yöntemidir. Bu sayede satışların arttırılması, rota verimliliğinin arttırılması ve aynı zamanda doğru müşteriye doğru ürün ile gitmeyi sağlamaktadır. Aynı zamanda raf dilimi, özendirme (promosyon), stok seviyesi kontrolü gibi konularda da fırsatlar sunmaktadır.

Pazar sepeti analizi yapılarak ürün birliktelikleri incelenmekte ve hangi ürün gruplarının daha çok bir arada satıldığı ölçümlenmektedir. Aynı zamanda bu analiz ile Hadoop sistemi ve paralel işleme üzerinde çalışmalar yapılmıştır. Bu çalışmalar ile Hadoop kümesi ile elde edilen zaman kazancı gözlemlenmiştir.

Bu çalışmada yapılan analizlerden ikincisi kümeleme algoritmalarıdır. Bu işlem için k-means (k -ortalama) algoritması seçilmiştir. Kümeleme analizi ile belirli bir veri kümesi içinde bulunan elemanları kendi içlerinde ki yakınlıklarına göre gruplama yöntemidir. Bu üst küme kendi içerisinde alt kümelere ayrıştırılır. Bu işleme kümeleme veya gruplama adı verilir[20]. Temel olarak uzaklık tanımda geçmesine rağmen aslında verilen özelliklere göre benzer elemanları gruplayarak benzerlik kümelerine ayrıştırma işlemidir. Yapılan kümeleme işlemi sonrasında aynı gruptaki elemanlar işlemin doğal sonucu olarak benzerlikleri fazla olanlardır. Farklı grupta bulunan elemanlar arasında ki benzerlik ise azdır.

Kümeleme algoritması olarak k-means seçilmiştir ve detaylı bilgi ilerleyen bölümde verilecektir. Bu algoritma ile satış miktarı ve karlılık değerlerine göre ürün ve müşterilerin sınıflandırılması amaçlanmaktadır.

3.1. HORTONWORKS DATA PLATFORM (HDP)

Hortonworks, önde gelen Apache Hadoop sağlayıcısıdır. Amaçları olarak kurumsal boyuttaki işletmelerin yeni mimarilerinin temelinde Hadoop'un kullanılmasının sağlanmasını belirtmektedirler[11]. Büyük veriler için çözüm olarak Hortonworks Data Platformu (HDP ™) konumlandırılmıştır. Hortonworks Data Platformu (HDP) ile bir veri kümesi üzerinde aynı anda toplu, interaktif ve gerçek zamanlı uygulamaları çalıştırmak için merkezi bir mimariye sağlayan bir kurumsal veri yönetimi platformudur. HDP Apache Hadoop üzerine inşa edilmiş ve YARN ile desteklenmiştir.

Hortonworks kurucuları Apache Software Foundation (ASF) yönetişim modeli altında işbirlikçi açık kaynak geliştirme yoluyla Hadoop gibi teknolojilerin iyi şekilde geliştirileceğine inanmaktadırlar. Dolayısı ile %100 açık kaynak kodlu bir platform sunmaktadırlar.

[11] http://hortonworks.com/about-us/quick-facts/

Tüm bu çalışma boyunca Hortonworks Hadoop alt yapısı kullanılmıştır. Açık kaynak kodlu olması, yerel sistemlerde çalışabilmesi sebebi ile Hortonworks tercih edilmiştir. Aslında Hadoop sektörün temel 3 hizmet sağlayıcı vardır. Bunlar Hortonworks, Cloudera ve MapR firmalarıdır. Bu firmalar şuan sektörün en öncü hatta tamamını elinde bulunduran firmalardır. Bu firmalara kısaca göz atmak ve neden Hortonworks tercih edildiğine bakılmasında yarar vardır.

Cloudera Hadoop kuruluşundan beri sektörde ilk kurulan organizasyondur. Hortonworks daha sonra gelmiştir. Cloudera ve Hortonworks yüzde 100 açık kaynak olmakla birlikte, MapR çoğu sürümleri tescilli ve ücretli bölümler ile birlikte gelir. Her biri farklı özelliklere sahiptir. Her bir organizasyonun artıları ve eksileri olmakla beraber birçok ortak özelliklere sahiptirler. Eğer Hadoop altyapısından ve teknolojisinden en iyi şekilde yararlanmak isteniyor ise bu ilk üç Hadoop organizasyonundan birinde, karşılaştırma yaparak karar kılınmalıdır.

Cloudera A.Ş. 2008 yılında Facebook, Google, Oracle ve Yahoo'dan büyük veri uzmanları tarafından kurulmuştur. Apache Hadoop tabanlı yazılım geliştirme ve dağıtım firması oldu. Sektörde hala kullanıcı sayısı olarak en çok istemciye sahip hizmet sağlayıcıdır. Temel özellikleri Apache Hadoop'a dayanmasına rağmen Cloudera Management Suite ile gerçek zamanlı düğümlerin sayısını yönetmek, görüntüleme, yükleme işlemini otomatikleştirmek ve kullanıcıların rahatlığını arttırmak için geliştirmiştir[12].

Hortonworks, 2011 yılında kurulmuş ve hızla Hadoop'un önde gelen hizmet sağlayıcılarından biri olarak ortaya çıkmıştır. Apache Hadoop'a dayalı büyük veri analizi, depolama ve yönetmek için açık kaynak platform sağlar. Hortonworks tam bir açık kaynak Apache Hadoop dağıtımı için ticari satıcıdır. Bu da demektir ki tamamen Hadoop'a odaklı işler geliştirilmektedir. Hortonworks HDP 2.x doğrudan ücretsiz olarak kendi web sitesinden indirilebilir ve kurulumu kolaydır. Hortonworks mühendisleri daha çok YARN üzerine yoğunlaşmışlardır. MapReduce ile çalışma yapısın değiştirilerek YARN ile Hadoop'un en son yeniliklerinin çoğunun temelini atmış oldular[13].

[12] http://www.cloudera.com/
[13] http://hortonworks.com/hadoop/yarn

MapR, standart olarak açık kaynak olarak gelmekle birlikte bazı kısıtlamalara sahiptir. Bu kısıtlamaları kaldırmak ücrete tabidir. Genellikle standart sürüm ile üstesinden gelinemeyen durumlar olmaktadır ve bunun için bazı eklentileri satın almak gerekmektedir. Mesela hata ayıklama, sistem güvenliğinin sağlanması ve teknik destek gibi konular.

Her üç firmada danışmanlık, eğitim ve teknik destek sunmaktadır. Fakat sadece Hortonworks'un %100 açık kaynak kodlu olduğu bilinmektedir. MapR bir adım daha ileri giderek, HDFS bileşenini değiştirir ve yerine MapRFS denilen kendi özel dosya sistemini kullanır[14]. MapRFS üzerinde veriler daha güvenilir ve en önemlisi daha verimli yönetimini sağlar.

Hortonworks ve Cloudera arasındaki benzerlikler aşağıdaki gibi sıralanabilir.

- Her ikisi de kurumsal yapılar için hazır olduklarını ifade etmektedirler. Kullanıcılarına güvenlik ve istikrarı garanti etmektedirler.
- Her ikisi de katılımcı bir topluluk oluşturmuştur. Aynı zamanda karşılaşılan zorluklara bu topluluk sayesinde yardımcı olmaktadırlar.
- Her ikisi de master-slave yapısını desteklemektedir.
- Her ikisi de MapReduce ve YARN'ı desteklemektedir.

Hortonworks ve Cloudera arasındaki farklar aşağıdaki gibi sıralanabilir.

- Cloudera'nın duyurduğu kadarıyla uzun vadeli hedefi: veri ambarı ihtiyacını azaltarak "Kurumsal veri merkezi" haline gelmektir[15]. Hortonworks ise Hadoop'un dağıtıcısı ve hizmet sağlayıcı olarak kalmayı planlamaktadır. Veri merkezi amacı için TeraData şirketi ile iş ortaklığına gitmiştir[16].
- Clodera CDH Windows Server üzerinde çalışabilir olsa da HDP Windows Server üzerinde yerli bir bileşen olarak kullanılabilir. Ayrıca Windows tabanlı bir Hadoop kümesi HDInsight aracılı ile Windows Azure üzerinde çalışmaktadır.
- Kolay ve gerçek zamanlı erişim için Clodera özel bir yazılı kullanır, Hortonworks ise Ambari kullanmaktadır.

[14] http://doc.mapr.com/display/MapR/MapR+Overview
[15] http://www.cloudera.com/why-cloudera.html
[16] http://hortonworks.com/partner/teradata/

- Hortonworks açık kaynak lisansı sahipken Cloudera, ticari bir lisansa sahip. Cloudera da ücretsiz olarak kendi açık kaynaklı projelerin kullanımına izin verir, ancak paket yönetim paketi Cloudera Manager veya başka herhangi bir özel yazılım içermez.
- Hortonworks %100 ücretsizdir. Clodera 60 günlük deneme süresi verir.

Master düğümler için Hortonworks Sandbox tercih edilmiş. Hadoop sistemlerinin genel kabul gören yapısı gereği düğümdeki makinelerin benzer sistemler olması ve daha kararlı şekilde çalışması sebebi ile Slave düğümler için de Hortownroks Sandbox sistemini üzerinde barındıran işletim sistemi olan Linux CentOS 6.7 tercih edilmiştir.

3.2. PAZAR SEPETİ ANALİZİ (PSA)

Geçmiş veriler analiz edilerek strateji ve yaklaşım belirleme, ürünlerin bir birleri ile birlikteliklerine bakılarak verilecek kararlar ile gelecekte nasıl bir yol haritası izleneceğini belirleyen yöntemlere birliktelik kuralları denir. Birliktelik kuralı ile veri madenciliği uygulamasına pazar sepeti analizi örnek verilebilir[21]. Birliktelik kuralları, işlemlerden oluşan ve her bir işlemin de elemanlarının birlikteliğinden oluştuğu düşünülen bir veri tabanında, bütün birliktelikleri tarayarak, sık tekrarlanan birliktelikleri ortaya çıkarmaktır[22].

Müşterilerin satın aldıkları faturalar içerisindeki ürünler incelenerek müşterinin almış olduğu ürünlerin bir birlerine olan birliktelikleri belirlenir. Bu sayede müşterinin yapmış olduğu satın alma alışkanlıkları tespit edilmektedir. Ortaya çıkan bu veriler ile daha fazla nasıl kazanç elde edilir, daha farklı nasıl fırsatlar yaratılır gibi pazarlama çalışmaları yapılır. Örnekleyecek olur isek müşterilerin meyve suyu ve çikolata satın alımlarının %60'inde bu ürünler ile birlikte bisküvi de satın almış ise yapılacak raf diziliminde bu bilginin dikkate alınması gerekmektedir. Bu sayede birliktelik örüntüsünden yararlanılarak kazan elde edilmiş olur. Bu tür birliktelikler belirlenirken ürünlerin birden fazla kez aynı sepette satın alınması gerekmektedir.

Genellikle perakende veya toptan satış yapılan noktalarda satış verileri toplanmaktadır. Toplanan bu veriler müşteri, ürün, Fatura, miktar, fiyat gibi verileri içermektedir. Toplanan bu veriler üzerinde hangi ürünün hangi ürün ile hangi sıklıkta satıldığının bilgisine erişmek için kullanılabilir. Pazar sepeti verisinde yer alan bir kayıtların fatura

numarası tekrarsızdır ve tarih bilgisinin yanında bu faturada hangi ürünlerin satın alındığı bilgisi, miktarı ve fiyatı da bulunmaktadır[23].

Pazar sepet analizinde (market basket analysis) amaç, satışlar arasındaki ilişkileri bulmak ve buna bağlı kuralları çıkarmaktır. Bu ilişki bilinir ise satış yapan firmanın kârını arttırması da olasıdır. Eğer X ürününü alan müşterilerin Y ürününü de aldığı belirleniyor ise ve bu yüksek bir ihtimal ise bu sonuçtan yola çıkarak farklı sonuçlar ve kazanımlar elde edilebilmektedir. Örneğin bu bilgi ışığında X ürününü alan bir müşteri Y ürünü almamış ise o müşteri büyük ihtimalle Y ürünü alma eğilimindedir. Buna benzer veri analizleri yaparak ürün bazında gelecek ayların satış verileri belirlenebilir. Bu sayede birlikte satın alınan ürünler için özendirme (promosyon) uygulaması ve reyon dizilişleri yapılabilir, müşteriler satın aldıkları ürünlere göre gruplandırılabilir, yeni bir ürün için potansiyel müşteriler belirlenebilir. Özellikle sektörde perakende noktalarına satış yapan üretici veya dağıtım şirketlerinin satış temsilcileri için sahada satış yaparken de bu bilgiler çok değerlidir. Örneğin satış esnasında müşteri önceliklerine göre ürün erişimi veya anlık özendirme (promosyon) veya ürün önerimi gibi bilgiler daha hızlı aksiyon alınmasını sağlayacaktır.

Pazar sepeti analizi ile birliktelik kuralları çıkarımı ilk olarak Agrawal ve diğerleri[22] tarafından 1993 yılında ele alınmıştır. Bu bağlamda X ve Y'nin nesne kümesi oluşu X ⇒ Y (X birliktelik Y) şeklinde ifade edilmektedir. Kuralları oluşturabilmek için destek (support) ve güven (confidence) değerlerini kullanarak, kullanıcı tarafından belirlenmiş minimum destek ve minimum güven değerlerinden yaygın birlikteliklerin belirlenmesi amaçlanmıştır.

Birliktelik kuralındaki temel kavramlar aşağıda belirtildiği gibidir.

Öğeler Kümesi: Bir ya da birden fazla öğeden oluşan kümenin adıdır.

Destek sayısı: Öğeler kümesinin tüm verileri içeren veri kümesinde görülmesinin tüm veri kümesine oranıdır.

Destek: Veride, belirtilen öğenin karşılaşılma sıklığıdır. Öğeler kümesinin içinde bulunduğu birlikteliklerin toplam birliktelik sayısına oranıdır. Destek(X=>Y) şeklinde gösterilmektedir.

$$Destek(X) = \frac{X \text{ öğesinin bulunduğu işlem Sayısı}}{Toplam\ işlem\ sayısı} \quad (3.1)$$

$$Destek(X => Y) = \frac{X\ ve\ Y\ nin\ bir\ arada\ bulunduğu\ işlem\ Sayısı}{Toplam\ işlem\ sayısı} \quad (3.2)$$

Güven: X ürünü alan bir müşterinin Y ürününü satın alma olasılığını vermektedir. Bu değer ürün birlikteliklerini yani destek sayısının doğruluk oranıdır. Güven(X=>Y) şeklinde gösterilmektedir.

$$Güven(X => Y) = \frac{X\ ve\ Y\ nin\ bir\ arada\ bulunduğu\ işlem\ Sayısı}{X \text{ öğesinin bulunduğu işlem Sayısı}} \quad (3.3)$$

Yaygın öğeler: Destek değeri en küçük destek sayısından büyük veya eşit olan öğeler kümesidir.

Destek ve güven değerleri 0 ile 1 arasındadır veya yüzde olarak ifade edilmektedir. 1'e ne kadar yakın bir değere sahipler ise ürünler arasında ki ilişki o derece yüksek olasılıklı ve güçlüdür denilir. Güven değeri 1 ise bu ürünler kesinlikle birlikte satılmaktadırlar. Bu sebepten güven değerine alt sınır ne kadar büyük verilirse o kadar kesin doğru sonuçlar elde edilir. İki öğenin birlikteliğinin kesine yakın olabilmesi için hem destek hem de güven ölçütünün yüksek olması gerekmektedir.

Örnek olarak farklı alış verişleri temsilen Tablo 3.1'deki veriler arasında Pazar Sepeti Analizi yapacak olur isek;

Tablo 3.1: Örnek pazar sepetini oluşturan faturalar.

İşlem ID	Elemanlar
İ1	E1,E2,E3
İ2	E1,E2
İ3	E2,E4
İ4	E2,E3,E4

1 Elemanlı öğeler kümesi aşağıdaki gibidir.

{E1} → Destek Sayısı(E1, E2)=2

{E2} → Destek Sayısı(E1, E3)=4

{E3} → Destek Sayısı(E1, E2)=2

{E4} → Destek Sayısı(E1, E2)=2

2 Elemanlı öğeler kümesi aşağıdaki gibidir.

{E1,E2} → Destek Sayısı(E1, E2)=2

{E1,E3} → Destek Sayısı(E1, E3)=1

{E2,E3} → Destek Sayısı(E1, E2)=2

{E2,E4} → Destek Sayısı(E1, E2)=2

{E3,E4} → Destek Sayısı(E1, E2)=1

$$Destek(\{E1\},\{E2\}) = \frac{Destek\ Sayısı(\{E1,E2\})}{Toplam\ Alışveriş\ Sayısı} = \frac{2}{4}$$

$$Güven(\{E1\},\{E2\}) = \frac{Destek\ Sayısı(\{E1,E2\})}{Destek\ Sayısı(E1)} = \frac{2}{2}$$

Yukarıdaki örnekte görüldüğü gibi bu veri kümesi üzerinde yapılan analiz neticesinde E1 ürününün her 2 alış verişten birinde alındığı ortaya çıkmaktadır. Ayı zamanda E1 ürününü alan her müşterinin E2 ürününü kesin aldığı ortaya çıkmaktadır.

Birliktelik kuralını bulmak için kullanılan bazı algoritmalar vardır. Bu algoritmalar genellikle iki adımda ele alınmaktadır. İlk olarak analizi yapanlar tarafında belirlenen destek sayısı alt sınırı verilerek bu sınır üstünde ki destek oranına uyan öğeler kümesi bulunur. Bu kümelerin genel adı sık geçen öğe kümesi olarak belirlenmiştir. Verilen örneklerde A adet ürün veya öğe varsa genellikle 2A adet sık geçen öğe kümesi bulunur.

İkinci adım ise sık geçen öğe kümelerini tarayarak güvenlik değerinin yüksek olduğu ürün birliktelikleri bulunmaktadır. Bu adımdaki her X öğe kümelerine ait boş olmayacak şekilde alt kümeler bulunur. Boş olmayan X alt kümeleri A ile gösterilecek olur ise A kümesi için X ⇒ (A - X) gerektirmesi, A kümesinin destek değeri X kümesinin destek değerine oranı minimum güvenirlik eşiği ölçütünü sağlıyorsa X ⇒ (A - X) birliktelik

kuralı olarak üretilir. Minimum destek sınırı göz önüne alınarak üretilmiş sonuç uzayında, güvenirlik alt sınırına göre tarama yapılarak bulunan birliktelik kuralları büyük oranda aranan bilgiyi içermektedir.

Birliktelik kurallarının bulunmasında performans ölçümlerini etkileyen en önemli aşama birinci adımdır. Sık geçen öğe kümeleri belirlendikten sonra, birliktelik kurallarının hesaplanması basit bir adımdır. Öğeler arasında ki örüntü bulunduktan sonra destek, güven değerlerinin bulunmasını kapsar. Destek oranı X öğesinin ne sıklıkta olduğunu, güven oranı Y öğesinin hangi oranda X öğesi ile birlikte olduğunu belirtir. Bir alt kümenin birlikteliğinin doğruluğu yukarıda belirtildiği gibi destek ve güven oranlarının olabildiğince yüksek olmasına bağlıdır.

Birliktelik kurallarının belirlenmesinde çeşitli algoritmalar mevcuttur. Bu algoritmalar aşağıdaki gibidir.

- Sıralı Algoritmalar
    - AIS Algoritması
    - SETM Algoritması
    - Apriori Algoritması
    - Apriori-TID Algoritması
    - Apriori-Hybrid Algoritması
    - OCD Algoritması
    - Partirtioning Algoritması
    - Sampling Algoritması
    - DIC Algoritması
    - CARMA Algoritması
    - FP-Growth Algoritması
- Paralel ve Dağıtılmış Algoritmalar
    - Count Distribution Algoritması
    - Paralel Data Mining Algoritması
    - Distributed Mining Algoritması
    - Common Candidate Partitioned Database Algoritması
    - Data Distibution Algoritması
    - Intelligent Data Distibution Algoritması

- Hash-Based Parallel mining of Association Rules Algoritması
- Parallel Association Rules Algoritması
- Candidate Distribution Algoritması
- Skew Handling Algortiması
- Hybrid Distribution Algoritması

Bu çalışmada veri madenciliği yöntemlerinden birliktelik kuralları üzerinde durulmuş ve özellikle pazar sepeti analizi olarak yaygın kullanıma sahip olan birliktelik kuralı ele alınmıştır. Çalışmada hızlı tüketim sektöründe dağıtım yapan Hedef Gıda A.Ş.'nin 2014 ve 2015 yıllarına ait satış verileri temel alınarak ürünler arasındaki birliktelikler ve müşteri bazında birliktelikler belirlenmeye çalışılmıştır. Apriori Algoritması temel alınmıştır sebebi paralel algoritmaların Apriori algoritmasından türetilmesidir.

Apriori birliktelik kuralları içerisinde en çok bilinen algoritmadır. Bu algoritma sıralı işletildiğinde sık geçen öğe kümelerini bulmak zor ve tekrar eden taramalar yapmayı gerektirmektedir. İlk olarak bir elemanlı destek oranının alt sınır şartını sağlayan sık geçen öğe kümeleri bulunur. Sonraki taramalarda ise potansiyel sık geçen öğe kümelerini bulmada kullanılır.

Tarama sırasında destek oranları hesaplanır ve her geçişte o geçişte üretilen sık geçen öğe kümeleri olur. Sık geçen öğe kümeleri bir sonraki geçiş için aday küme olurlar. Bu işlemler sık geçen öğe kümesi bulunmayıncaya dek sürer.

Bu çalışmada MapReduce ile Sepet analizi yapılacağından Paralel Birliktelik kural belirleme algoritması kullanılmıştır. HDFS ve MapReduce kullanıldığı için yukarıda belirtilen algoritmaları salt bir tanesi uygulanmamıştır. Hem veri paralel işleme hem de görev paralel işleme kullanılmıştır.

3.3. K-MEANS ALGORİTMASI

Kümeleme, verileri gruplara ayırma işlemidir. Bu ayrıma işleminde eğer örneklemeler ve sınıflamalar belirli ise gözetimli sınıflama olarak adlandırılır. Eğer sınıflar belli değil ve öğelerin özelliklerine göre sınıflama yapılacak ise bu kümelemeye gözetimsiz sınıflama denir. Genel olarak her gelen öğenin sınıfının belirlenmesi için temel alınan özelliklerine bakılarak hangi sınıfa daha çok benzerlik gösteriyor ise öğe o sınıfa alınır ve bu işlem

gözetimli sınıflama işlemidir. Eğer sınıflar da belli değil ise hem öğelerin özellikleri inceleniyor hem de sınıflar belirleniyor ise gözetimsiz sınıflama yapılıyor demektir. Kümeleme işlemleri öğelerin taşıdığı nicel veya nitel özelliklere bakarak yapılmaktadır. Sınıflandırma işlemleri genellikle bilim, finans, uzay ve veri madenciliği alanlarında kullanılır. İstatistiksel işlemlerinde k-means ve k-medoids sınıflama işlemleri yoğun olarak kullanılmaktadır ve aynı zamanda bu işlemler için bazı paket programlar mevcuttur. Bilişim dünyasında daha çok bilimsel çalışmalar ve son yıllarda iş dünyası içim yapay zekâ alanında makine öğrenmesi için gözetimsiz sınıflama işlemlerinde kullanılmaktadır[24].

Kümeleme işlemlerinin özellikleri aşağıdaki gibidir;

- Ölçeklenebilir olmaları.
- Farklı veri türleri ile kullanılabilmeleri
- Düzgün şekilli olmayan kümleri de bulabilmeleri
- Gürültü içeren veriler ile kullanılabilmeleri
- Analiz sırası hangi elemandan başlanırsa başlansın sonuç değişmemelidir.
- Çok boyutlu veritabanlarına uygulanabilmelidir.
- İşlevseldirler
- Kolay yorumlanabilir sonuçlar üretirler.

Kümeleme analizlerinde veriler matris şeklindedir ve farklı veri türlerine göre farklı yöntemle analiz edilirler. Genellikle sınıflandırma işlemi doğrusal uzayda, öğelerin özelliklerinin uzaklıklarına göre yapılmaktadır. Uzaklık hesaplaması ile yapılmaktadır. En çok kullanılan uzaklık hesaplama yöntemi Öklid yöntemidir.

Gözetimsiz sınıflama işlemlerinde öncelikle gelen öğelerden sınıflar oluşturulur. Sonra her gelen öğe bir sınıfa atanır ve son olarak sınıfların tekrar belirlenme işlemi yapılarak tekrardan sınıflama işlemi yapılır. Bu işlem sınıflar değişmeyene kadar devam edilir. Sınıf belirleme ve sınıflara öğeleri atama işlemleri mesafeye göre yapıldığından mesafelerin veri türlerine göre nasıl hesaplanabileceğine göz atmakta yarar vardır.

Sayısal Değişkenler: Bu değişkenler doğrusal bir uzayda temsil edilen özelliklerdir. Bu tür ölçekler için uzaklık ya da komşuluk mesafesi hesaplama formüller ile yapılır ve bu formüller aşağıda ki gibidir.

- Öklid formülü
- Manhattan uzaklığı
- Minkowski uzaklığı

İkili (Boolean) Değişkenler: Evet veya Hayır olan değişkenlerdir. "0" ya da "1" değerini alabilirler değişkenlerdir. Bu değişkenler bir özelliğin değerini değil sadece o özelliğin olup olmadığını belirtmektedir. Örneğin bir kişinin öğrenci olup olmaması veya havanın güneşli olup olmadığını belirtir. Ya evet veya hayır cevabı sebebi ile kesinlik vardır. Hesaplama için Tablo 3.2 kullanılır. Bu tabloda i ve j öğeleri arasında olan uzaklık bir matris ile ifade edilmektedir.

Tablo 3.2: İkili değişkenlerde uzaklık hesaplama tablosu.

		Nesne j		
		1	0	*toplam*
Nesne i	1	a	b	$a+b$
	0	c	d	$c+d$
	toplam	$a+c$	$b+d$	p

Nominal, Ordinal ve Oran Değişkenleri

Nominal değişkenler sonlu sayıda değer alırlar. Bu özellikler bir durum belirli değişkenlikler arasında ayrıştırma ihtiyacını karşılamak içindir. Örnek olarak arabanın renklerine göre gruplanması işlemi Nominal değişkenli gruplamaya örnektedir.

Ordinal değişkenler de nominal değişkenlerde olduğu gibi sonlu sayıda değer içerir. Fakat ordinal değişkenler anlamlı bir sıra içermezler. Bu durumda bu sınıflandırma işleminde her bir değere farklı bir sayı verilerek temsil edilirler.

Oran ölçekli değişkenler doğrusal olmayan ölçek üzerinde yapılan ölçümlerin sonuçlarıdır. Oranın yüksekliği ilgili sınıfa yakınlığını belirtmek içinde kullanılabilmektedir.

Bazı özellikler karışık tür değişkenlerdir. Bunlar yukarıda bahsedilen değişkenlerin bir kaçını içerisinde barındırabilmektedir. Hesaplama için iki farklı yöntem kullanılır. Ya her gurubu kendi türü içerisinde sınıflamalı veya veri türü için genel bir hesaplama yapılmalı.

Kümeleme için yapılacak sınıflandırma işleminin niteliğine bakılarak bir yaklaşım sergilenir ve hangi yöntemim seçileceğine karar verilir. Kümeleme için x elemanlı bir kümede, girdi olarak alınan *k* sayısı küme sayısını belirtmektedir. Bu değer k<x şartına uygun olmak zorundadır. Belirtilen kümede ki elemanları k adet kümeye bölme işlemi yapılır. Bu işlem için k-means veya k-medoids gibi algoritmalar kullanılmakla birlikte biz bu çalışmada k-means algoritmasını kullandık.

K-means en eski sınıflara ayırma yöntemlerinden biri olarak 1957 yılında ilk kez Hugo Steinhaus tarafından öne sürüldü. Asıl geliştirilmesi 967 yılında J.B. MacQueen tarafından yapılmıştır[17].

K-means algoritmasının temel amacı x elemanlı bir kümenin, program çalışmadan verilen k adet kümeye ayrıştırılmasıdır. Bu sınıflandırmada ki amaç alt kümelerin elemanlarının kendi içlerinde benzerliklerinin fazla kümeler arası benzerliklerin az olmasıdır. Kümelerin kendi içerisindeki benzerliği, küme merkezi ile kümedeki diğer elemanlar arasındaki uzaklıkların ortalamasıyla ölçülmektedir[25].

Çok yaygın kullanılan gözetimsiz öğrenme olan k -means'in yapısında verilen k küme merkezi etrafında gruplana ve sadece bir kümeye ait elemandan oluşur. Bu özelliği sayesinde kesin bir sonuç üretir. Yani bir elaman bir kümeye üye olabilir. Küme merkezi kümenin kendisini temsil eder. Eşit büyüklükte olan kümeleri bulma işleminde başarılıdır.

[17] https://en.wikipedia.org/wiki/K-means_clustering

Diğer sınıflandırma algoritmaları farklı türdeki veriler üzerinde iyi sonuçlar vermesi mümkündür. Fakat k-means algoritması her veri türünde çalışması ile bir adım daha önderdir. Algoritmanın dezavantajı ise aşağıdaki gibidir:

- Algoritmanın başlamadan k küme sayısı verilmelidir. Eğer k değeri belirlenmemiş ise deneme yanılma ile bulunur.
- Aşırı gürültüler algoritmanın sonucunu etkilediğinden gürültüye çok duyarlıdır.
- Çakışan kümelerde iyi sonuçlar alınamamaktadır.
- Sayısal verilerin kümelenmesi için çok başarılı iken kategorik veriler için başarılı sonuçlar vermemektedir.

K- means algoritması ilk başta küme sayısı olan k değerinin verilmesi ile başlar. Daha sonra ilk gelen öğelerden k adet alınarak merkez olarak belirlenirler. Fakat bazı durumlarda merkez öğeleri rastgele seçilir. Tüm elemanlar okunup belirlenen küme merkezlerine atama işlemi gerçekleştirilir. Bu işlemler yapılırken uzaklık için veri türüne göre uzaklık hesaplaması yapılır. Bu sayede elemanın hangi küme merkezine yakınsa o kümeye atanması sağlanmış olur.

Noktalar arası uzaklığın bulunması için yaygın olarak Öklid kullanılır ve iki özellik varsa tek düzlem kullanılır. İkiden fazla özellik var ise gruplamalarda çok boyutlu düzlem kullanılır. Algoritmanın geometrik gösteriminde küme sınırları yukarıda anlatıldığı şekilde belirlenmektedir. Bilgisayar programları ile geliştirilen k-means algoritmalarında ise, düzlemler yerine noktalar arasındaki uzaklıklar hesaplanarak, noktaların merkeze yakınlığı dikkate alınmaktadır. Kümenin elemanları merkezlere atandıktan sonra ikinci aşama sonlanmış olmaktadır.

Son aşama olarak her kümeye eklenen öğe ile kümenin tüm elemanlarının ağırlıklı ortalaması bulunur. Bu sayede küme merkezi bulunur. Ağırlıklı ortalama kümenin elemanlarının tümünün boyutlarında ki değerlerin ortalamasının alınması ile hesaplanmış olur. Algoritmada ilk döngü sonrasında bulunan küme merkezleri artık eleman değildir sadece küme merkezini temsil etmektedir. Bundan sonraki seçim işlemlerinde küme merkezini bu yeni değerler tarafında temsil edilmiş olur. Döngüler değiştikçe bazı elemanlar farklı kümelere dâhil edilebilir çünkü küme merkezleri de değişmektedir. Sınıflandırma işlemi, küme merkezlerinin değişmemesine kadar devam eder. Yapılan

algoritmada belli bir döngü sayısı sonrasında kümeler kararlı bir hal alır. Algoritmayı sıralı bir hale getirecek olur isek aşağıdaki gibi hal alır.

- Küme elemanlarının değerini belirtir k sayısı girilir.
- k küme sayısı kadar küme merkezi seçilir.
- Tüm elemanların merkezlere olan uzaklığı seçilir.
- Elemanlar yakın olduğu merkeze atanır.
- Kümeye ait tüm elemanların merkezleri bulunur ve yeni küme merkezi belirlenir.
- Son bulunan küme merkezini temsil eden değerler ile bir önceki orta küme merkezleri aynı oluncaya kadar 2, 3, 4 ve 5. adımlar tekrarlanır.

Bu sayede k-means algoritması tamamlanmış ve küme merkezleri ve kümeye ait elemanlar bulunmuş olur. Bu alt kümelerden bir örüntü veya benzerlik elde etmek ve farklılaşmaları tespit etmek için analizin son adımına görselleştirmeye ve raporlamaya geçilir.

4. BULGULAR

4.1. PAZAR SEPETİ ANALİZ UYGULAMASI

Bu çalışmanın temel amacı hızlı tüketim sektöründe dağıtım yapan Hedef Gıda A.Ş şirketinin iki yıllık verisi üzerinde Pazar Sepeti Analizi yaparak ürün birlikteliklerini bulup daha etkin özendirme (promosyon) ve kampanyalar düzenlenmesini sağlamaktır. Bu sayede daha etkin pazarlama teknikleri geliştirmektir. Aynı zamanda bu etkin analizleri yapmak için klasik yöntemler ve Hadoop üzerinde HDFS ve MapReduce kullanılarak yapılan analiz sonuçlarını süre ve kaynak olarak karşılaştırarak elde edilen kazancı ölçümleyebilmektir.

Yapılan yayınca (literatür) taramalarında genellikle aynı veri formatında çalışmalar ile karşılaşılmıştır. Bu çalışmalarda temel alınan çalışma Woo'nun [26] yapmış olduğu "Market Basket Analysis Algorithm on Map/Reduce in AWS EC2" isimli çalışmadır. İlgili makalede yapısal olmayan veriler kullanılmıştır. Bu veriler her bir satırda bir faturanın içerisindeki ürünleri tek satırda tutulacak şekilde bir yapıda olan veri seti ile çalışılmış. Yaptıkları çalışmalarda 1 adet map ve 1 adet reduce işlemi kullanılmış ve combine işlemi kullanılmamıştır. Temel alınan bu çalışmada Tablo 4.1'deki sonuçlar elde edilmiştir.

Tablo 4.1: Temel alınan çalışmadaki süre ölçümleri.

	3.4 M (200MB)	6.7 M (400MB)	13 M (800MB)	26 M (1600MB)
1	1,600,339.00	3,124,972.00		
2	1,099,115.00	1,885,996.00	3,471,285.00	5,632,922.00
5	986,587.00	1,091,723.00	1,325,612.00	2,529,631.00
10	978,587.00	986,329.00	1,102,558.00	1,290,564.00
15	1,003,506.00	1,060,552.00	1,028,739.00	1,172,775.00
20	990,033.00	1,045,166.00	1,028,399.00	1,131,514.00

Makalede yazılan algoritma ve veri yapısı incelendiğinde, yapısal olmayan ve her bir faturanın bir satırda tutulması sonucunda map işleminde paralel işlenemeyecek bir kısımdan söz etmek gerekmektedir. Bu veri yapısına bağlı olarak map işleminde okunan satırlar kendi içerisinde ürün eşleşmelerini bulmak için paralel olmayan bir yapıda döngüye girmekte ve bundan ötürü algoritmanın çalışma süresi uzamaktadır.

Bu durum göz önüne alınarak bu çalışmada yapısal veri kullanılmış ve her bir satırda tek bir faturanın sadece bir ürünü tutacak şekilde veri yapısı kullanılmıştır. Bunun sayesinde map işlemi sadece gelen verileri istenilen formatta olmasa da işleme fırsatı yaratılmaktadır. Ayrıca en önemli kazanımlar combine işlemi kullanılarak sağlanmıştır. Map ve reduce işlemi arasına combine eklenerek döngü kullanılan kısımda paralel işlenebilmesi sağlanmıştır.

Bu analizi yapmak için satış verileri kullanılmıştır. Kullanılan satış verileri temel olarak Müşteri bilgisi, Fatura bilgisi, Ürün Bilgisi ve miktar alanlarını barındırmaktadır. Bu bilgiler SQL sunucular üzerinde tutulmakta ve bir ERP sistemi tarafından oluşturulmaktadır. Dolayısı ile bu veriler Tablo 4.2'de gösterildiği gibi tamamen yapısal verilerdir (Structural Data).

Tablo 4.2: Çalışmada kullanılan yapısal veri seti örneği.

Müşteri	Fatura	Ürün	Miktar (Adet)
180087	2014v1_99999	29300	12
180087	2014v1_99999	30551	6
180087	2014v1_99999	30553	6
180087	2014v1_99999	30583	6
180087	2014v1_99999	30581	6
180087	2014v1_99999	30589	6
180087	2014v1_99999	29298	6
47341	2014v1_99998	949	12
47341	2014v1_99998	27457	12
47341	2014v1_99998	955	6
47341	2014v1_99998	968	6
47341	2014v1_99998	27459	6
47341	2014v1_99998	978	6
47341	2014v1_99998	11467	6
180310	2014v1_99996	30581	12
180310	2014v1_99996	31514	6
180310	2014v1_99996	30603	6
180310	2014v1_99996	30575	6
180310	2014v1_99996	32304	6

Tablo 4.2'deki veri kümesinde ve tüm çalışmada müşteri ve ürün bilgileri tekrarsız anahtar değerleri ile temsil edilmektedir. Bunun birincil sebebi bilgi güvenliğinin

sağlanmasıdır. Günümüzde saha araştırmalarının ücret karşılığında satıldığı bir ortamda müşteri ve ürün satış bilgilerinin paylaşılması pek doğru olmayacaktır. Aynı zamanda yapılan analizlerde veri boyutunu düşürerek daha hızlı analiz edilmesi sağlanmış olmaktadır.

Bu veri kümesinde yapısal olarak her bir müşterinin farklı zamanlarda farklı fatura bilgileri ile satın almış olduğu ürünleri ve miktarlarını göstermektedir. Bu arada her bir müşteri bir veya birden fazla satın alma gerçekleştirmiştir. Bunun yanında her bir faturada bir veya birden fazla ürün satın alınmış olabilir.

Çalışmada yukarıdaki veri kümesinde göründüğü gibi olan veri seti 5 milyon satırdan oluşmaktadır. Bu veri kümesindeki öğe sayıları aşağıdaki gibidir.

- Müşteri sayısı: 7.541
- Ürün sayısı: 4.852
- Fatura sayısı: 300.455

Bu da demek oluyor ki 7.541 müşteri toplamda 300.455 defa alışveriş yapmış ve bu alış verişler 4.852 ürünü kapsamaktadır.

Yapılan çalışmalarda klasik yöntemler ile analiz için kullanılan SQL Server için 5 milyon satırlı yapısal veriler kullanılarak ölçümler yapılmıştır ve ilerleyen bölümlerde bahsedilecektir. Hadoop üzerinde yapılan çalışmalarda ise farklı algoritmalar için denemeler yapılmıştır.

Öncelikli Hortonworks HDP 2.3 üzerinde küme (cluster) yapılandırması yapılarak çalışma için ortam hazırlanmıştır. Bu ortamdaki makinaların özellikleri yukarıda da bahsedildiği gibidir;

Master düğümü için ayrılan kaynak bilgileri

- 8 Core CPU
- 10 GB RAM

Slave düğümler için ayrılan kaynak bilgileri

- 4 Core CPU
- 4 GB RAM

Küme üzerindeki master düğümde tüm Hortonworks HDP 2.3 için kurgulanan servis ve istemciler kurulmuş durumdadır. Slave düğümler için ise tüm istemci uygulamalarının yanında DataNode (HDFS), NodeManager (Yarn), Metricks Monitor servisleri kurulmuştur.

DataNode: Küme içinde verilerin dağınık tutulması için gerekmektedir.

NodeManager: Küme içerisinde MapReduce görevlerinin dağıtık bir yapıda işlenmesi için şarttır.

Metrics Monitor: İlgili bilgisayarın izlenmesi ve sağlıklı çalışıp çalışmadığının gözlenmesi için gerekli ölçümlerin yapılarak görselleştirildiği servistir.

Kurulum ve küme yapılandırması sonrasında veriler HDFS üzerine yüklenmiştir. Veriler standart olarak HDFS üzerinde 3 kopya halinde tutulmaktadır. Ölçüm ve testler yapıldığında kümedeki eleman sayısı değiştirildiğinde veri yükleme işlemleri tekrar edilmiştir.

Pazar Sepeti Analizi için klasik yöntemler için karşılaştırma yapabilmek için SQL Server üzerinde analizler yapılmıştır. Bu yöntem ile verilerin defalarca tekrar okunması ve işlenmesi gerekmektedir. SQL Server üzerinde ise aşağıda Şekil 4.1'de gösterilen kod ile yine birden fazla kez aynı tablonun bir biri ile ilişkilendirilmesi ile sonuç alınabilir gibi görünmektedir.

```
Select Data1.Islem Islem,Data1.Urun ,Data2.Urun,Count(*) Frekans
from
Veri5M Data1  Join
Veri5M Data2 On Data1.Islem=Data2.Islem
Where Data1.Urun<>Data2.Urun
Group By Data1.Islem,Data1.Urun ,Data2.Urun
```

Şekil 4.1: İki öğeli pazar sepeti analizi için SQL sorgusu.

Fakat aslında 4 Core 2.40 GHz işlemcili ve 16 GB RAM bulunan bir SQL sunucuda bu sorgulama sonucunda yeterli kaynak olmadığı belirtir bir hata alınır. Çünkü bu sorgulama ciddi RAM ihtiyacı gerektirir ve bu RAM ihtiyacını karşılamak için belirli bir seviye üzerindeki sunucular kullanılmalıdır. Ancak aşağıdaki gibi bir Sorgu cümleciği kullanılarak RAM yerine disk kullanılarak hesaplama yapılır ve sonuç bir tabloya doldurulur. Bu durumda da ciddi anlamda satır sayısı ortaya çıktığından disk kullanımı artmakta ve süre uzamaktadır. Örnek olarak Tablo 4.2'deki gibi bir veri yapısında örnek alınan veri kümesinin toplam boyutu 323 MB iken T-SQL ile farklı kodlarla doldurulduğunda ortaya çıkan verinin toplam boyutu 3.255 MB (3.2 GB)'dır. Aynı zamanda bu sorgunun çalışma süresi 1 Saat 58 Dakika 47 saniyedir.

Bu boyuttaki küçük verilerin bile analiz edilmesinde bu denli fazla kaynak gerekiyor ise veri boyutunun veya miktarının hızla arttığı günümüzde klasik yöntemlerin ihtiyacı karşılamada yetersiz kalacağı ortay çıkmaktadır. Bu sebepten klasik yöntem yerine Hadoop üzerinde dağıtık bir yapıda paralel algoritma kullanmak kaçınılmaz olmaktadır. Bu aşamada öncelikli olarak Pazar Sepeti Analizi için algoritmanın hem verileri hem de görevleri dağıtık olarak saklanmalı ve işlenmelidir. Bunun içinde Pazar Sepeti Analizi için algoritmasını MapReduce'a uyarlamak gerekmektedir.

MapReduce yapısında tüm veriler paralel olarak tek tek Map işlemine tabi tutulur ve map işlemi çıktı olarak aşağıdaki gibi her bir anahtar için değerler listesi oluşturur;

$$Map(k1, v1) \rightarrow list(k2, v2) \tag{4.1}$$

Bu işlem sonrasında Map işlemin için önceden tanımlanış bir Combiner yok ise direkt olarak Reduce adımına geçilir. Fakat bu çalışmada daha verimli sonuçlar alınması için Combiner tanımlanmış ve Map işlemi sonrasında Combine adımına geçilmiştir. Yukarıdaki bölümlerde bahsedildiği gibi Combiner'lar aslında Reducer gibi işlemektedir. Aralarındaki tek fark ise Reducer'lar çıktı olarak HDFS sistemine yazar fakat Combiner'lar çıktılarını Reducer'a gönderirler. O halde Combiner adımında işlem aşağıdaki gibi olacaktır.

$$Combine\big(k2, list(v2)\big) \rightarrow list(k3, v3) \tag{4.2}$$

Son olarak Combiner sonrasında Reduce adımında Reducer'a iletilen değerler işlenerek müşteri temelinde 2 öğeli birliktelik kümleri bulunur ve HDFS üzerinde belirtilen dosyaya çıktı olarak alınır.

$$Reduce\big(k3, list(v3)\big) \rightarrow list(v4) \tag{4.3}$$

MapReduce yapısında tüm veriler paralel olarak Record Reader'lar tarafından okunduktan sonra map görevi yerine getirilir. Yazılan mapper sınıfı ile sıra ile satır-satır okunan verinin nasıl bir şekilde işleneceği kararını veriliyor olmalıdır. Bu aşamada aslında Tablo 4.2'deki yapıda bir verinin okunacağı düşünülürse bu yapısal verilerin işlenmesi için İşlem ID bilgisi key olarak kabul edilmeli ve her bir satırdaki ürünler ise value olarak kabul edilmelidir. Mevut veri yapısında her İşlem ID bir faturayı temsil etmekle birlikte aynı zamanda bir İşlem ID sadece bir müşteriye ait olmaktadır. Yani bir fatura sadece bir müşteriye kesilmektedir. Bu çalışmada her bir müşterinin alımlarına bakarak ürün birlikteliği inceleneceği için Map işleminde key değeri olarak İşlem ID ile Müşteri bilgisini birleştirerek anahtar değeri üretmek programlama açısından, müşteri temelinde analiz yapıldığında hız kazandıracaktır ve aynı zamanda Combine aşamasında müşteri bazında gruplama yapılabilmesini de kolaylaştıracaktır.

Bu sayede her bir faturayı içerisine ürünleri de ekleyerek gruplamış olunmaktadır. Combiner olarak yine gruplanmış her bir fatura içerisindeki ürünler eşleştirilerek key olarak ürün eşleştirmesini yazılıyor olunacaktır ve her bir ürün çiftinin 2 elemanlı öğeler küme sayısını bulabilmek için ise her bir eşleşmeye value olarak 1 yazılmaktadır. Bu adımda 2 elemanlı öğe kümesine sadece müşteri bilgisini ekleyerek, müşteri gruplaması yapılır. Bu sayede Reduce aşamasında müşteri bazında her bir ürün eşleştirmesinden kaç tane olduğunu bulunabilecektir.

Reduce aşamasına girdi olarak müşteri bilgisi, 2 elemanlı öğe kümesi ve 1 değeri gelir. Her bir müşteri ve 2 elemanlı öğe kümesi Reduce aşmasına kümenin oluştuğu işlem sayısınca tekrarlı gelir. Bu tekrar sayısı 2 elemanlı öğe kümesinin frekansını verir. Algoritmanın modeli aşağıdaki gibidir.

function Map **is** $Satir_n$

Split(Ayrıştır): $Satir_n$.Split(SPLITTER) **To** Liste

let (Atama): Müşterii(Liste0)+ SPLITTER + İşlemj (Liste1) **To** Key

let (Atama): Ürün (Liste2) **To** Value

output record (Key, Value)

end function

function Conbiner **is** K1, list(V1)

Split(Ayrıştır): K1.Split(SPLITTER) **To** Liste

let (Atama): Müşterii(Liste0) **To** Müsteri

for each $Value_i$ **in** list(V1) **do**

for each $Value_j$ **in** list(V1) **do**

let $value_i$ + SPLITTER + $Value_j$ **To** tempKey

let Müsteri + SPLITTER + tempKey **To** Key

output record (Key, 1)

repeat

repeat

end function

function Reduce **is** K2, list(V2)

let 0 **To** Toplam

for each Value **in** list(V2) **do**

let Toplam + V2 **To** Toplam

repeat

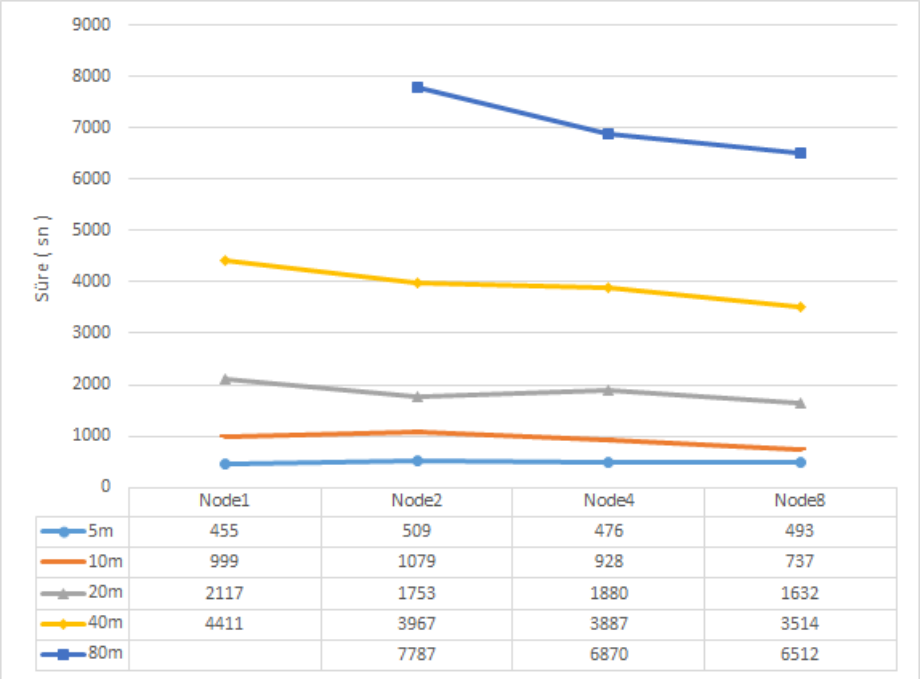
output record (K2, Toplam)

end function

Bu çalışma aşamasında farklı algoritmalar denenmiştir. Müşteri temelinde ürün birlikteliklerinin bulunması için adımlar aşağıdaki gibidir.

- Veri dosyasından sıra ile verilerin okunarak istenen şekilde formatlanması
  - Anahtar ve değer alanlarının belirlenmesi
- Aynı fatura içerisindeki ürünlerin küme sayısına göre eşleştirilmesi
  - Öğe sayısına göre kümlerin oluşturulması
- Aynı müşteriye ait eşleştirmelerin sayısının bulunması

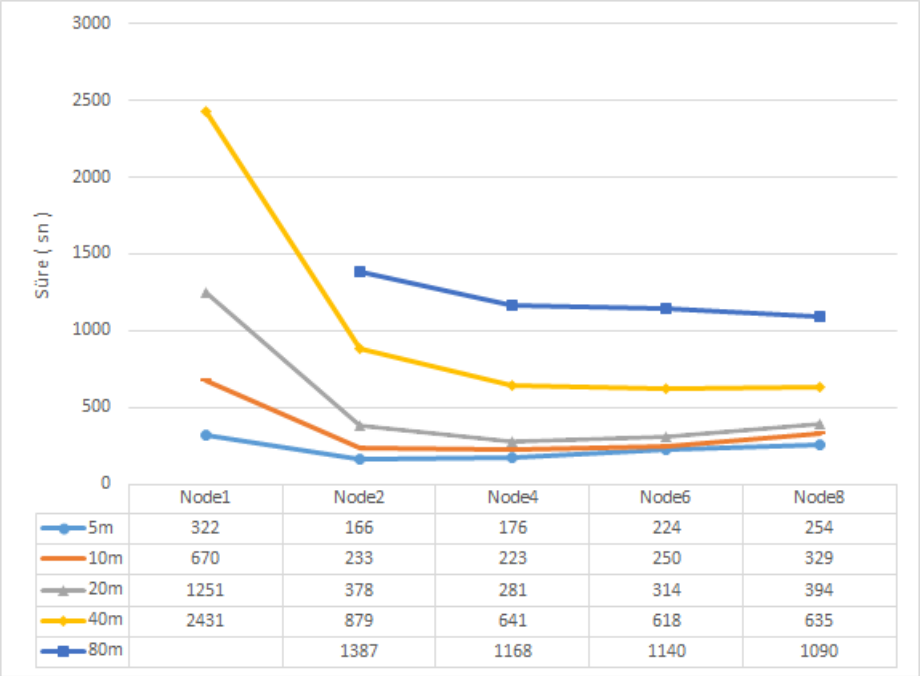
Üç adımlı bu işlemler için iki farklı görev yapılandırma yoluna gidilmiştir. Bu görevler birer mapper ve birer reducer sınıflarından oluşmaktaydı. Her bir reducer sınıfı dosyaya çıktı ürettiği için ilk MapReduce işleminden sonra bir ara çıktı üretilmesi gerekiyordu ki bu işlem sürenin uzamasına sebebiyet vermekteydi. İki görev oluşturularak yapılan çalışmalar sonucunda Hadoop kümesi üzerindeki düğüm sayısı ve veri boyutuna göre alınan sonuçlar Şekil 4.2'deki gibidir.



	Node1	Node2	Node4	Node8
5m	455	509	476	493
10m	999	1079	928	737
20m	2117	1753	1880	1632
40m	4411	3967	3887	3514
80m		7787	6870	6512

Şekil 4.2: Hadoop MapReduce ile iki görevli yapılan çalışma sonuçları.

İki görev tanımı kullanılarak bazı verilerin elde edilmesinden ve sonuçlarda istenilen ve yapılana makale taramalarında elde edilen değerlere ulaşılamamasından dolayı programlama modelini değiştirerek yeni denemeler yapılmıştır. Bu denemeler arasında en verimli sonuçlar bu çalışmada kullanılan Map adımında hem mapper hem de combiner kullanılması olmuştur. Dolayısı ile görev teke indirgenmiştir. Bu sayede Şekil 4.3'teki sonuçlar elde edilmiştir.



	Node1	Node2	Node4	Node6	Node8
5m	322	166	176	224	254
10m	670	233	223	250	329
20m	1251	378	281	314	394
40m	2431	879	641	618	635
80m		1387	1168	1140	1090

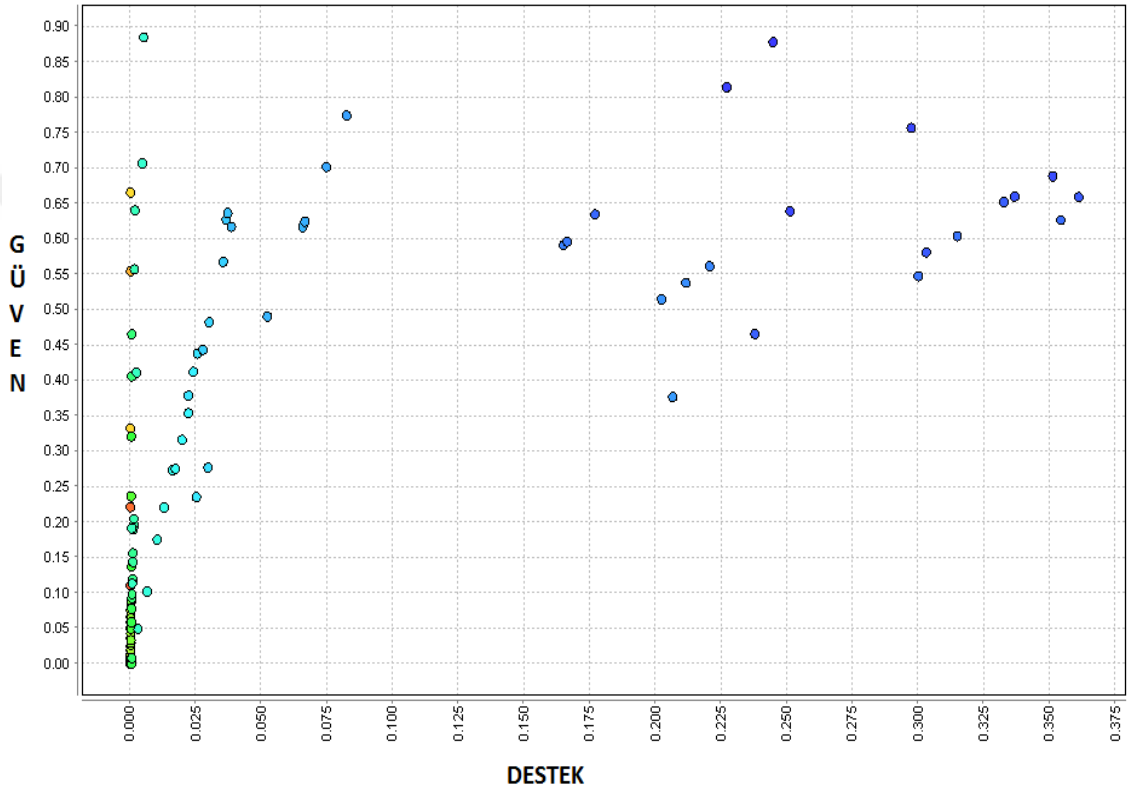
Şekil 4.3: Hadoop MapReduce ile tek görevli yapılan çalışma sonuçları.

Şekil 4.3'te görülen sonuçlara göre PSA (Pazar Sepeti Analizi) yapıldığında bir düğüm ile yapılan denemelerde elde edilen sonuçlar aşağıdaki gibidir.

- Bir elemanlı Hadoop kümesinde düğüm sayısı sabit tutularak veri miktarı arttırıldığında, analiz süresi veri sayısı ile doğru orantılı olarak artmaktadır.
- Hadoop kümesinde düğüm sayısı 2'ye çıkarıldığında, 5 milyon satırlı verinin analizi için harcanan süre ve bir düğüm ile analiz edilmesi için harcanan süre arasında ciddi farklar görülmemektedir.
- Hadoop kümesinde düğüm sayısı arttırılarak veri miktarı da arttırıldığında her düğüm sayısı artışında süre olarak ciddi kazanımlar elde edildiği gözlenmiştir.
- Veri boyutu sabit tutularak Hadoop kümesindeki düğüm sayısı arttırıldığında sürelerin azaldığı gözlenmiştir. Fakat süre artışı belirli bir düğüm sayısında doyum noktasına ulaşmış görünmektedir.
- Bu çalışmada her veri boyutuna göre doyum noktası değişmektedir. 5 milyonluk veride 2 düğüm, 10 ve 20 milyonluk veride 4 düğüm, 40 milyonluk veride 6 düğüm ve son olarak 80 milyonluk veride 8 düğümlü bir yapının doyum noktası olduğu gözlemlenmiştir.

- Elde edilen sonuçlar ile birlikte bu çalışmada kullanılan veri seti ve algoritma modeline göre en verimli düğüm sayısının değişmesi Hadoop ve MapReduce sayesinde çok ciddi kaynak ve zaman kazanımı olduğunu göstermektedir.

Çalışma neticesinde elde edilen ürün birlikteliklerini inceleyecek olursak, genel ürün birlikteliklerinin Güven ve Destek değerleri Şekil 4.4'te görülmektedir.



Şekil 4.4: Ürün birliktelik dağılımı.

Yukarıdaki grafik incelendiğinde bazı ürünlerin birlikteliklerinin Güven değerinin 1'e çok yakın olduğu yani bu ürünlerin %87 oranında sepette (faturada) birlikte bulundukları sonucu çıkmaktadır. Güven değeri yüksek olan ve Destek değerleri en yüksek ürün birlikteliklerinin ilk 10'u Tablo 4.3'de gösterilmiştir. Veri güvenliği nedeni ile ürün isimleri paylaşılmamıştır. Onun yerine kategori bazında elde edilen sonuçlar paylaşılmıştır.

Tablo 4.3: Güven ve Destek değeri en yüksek birliktelikler.

Kategori	Kategori	Güven	Destek
Pil ve şarj Ürünleri	Tıraş Bakım	88%	24%
Pil ve şarj Ürünleri	Ağız Bakım	82%	23%
Tıraş Bakım	Ağız Bakım	76%	30%
Deterjan	Saç Bakım	69%	35%
Deterjan	Bebek Bezi	66%	34%
Ağız Bakım	Bebek Bezi	66%	36%
Deterjan	Ağız Bakım	65%	33%
Tıraş Bakım	Ağız Bakım	64%	25%
Pil ve şarj Ürünleri	Bebek Bezi	64%	18%
Bebek Bezi	Saç Bakım	63%	35%

Tablo incelendiğinde "Pil ve şarj Ürünleri" kategorisindeki ürünler toplam fatura sayısının %24'ünde bulunduğu görülmektedir. Aynı zamanda bu kategorideki ürünleri alan her 100 müşteriden 88'i kesinlikle "Tıraş Bakım" kategorisindeki ürünleri aldığı sonucu ortaya çıkmıştır.

Destek sayısı en yüksek olan ürünler incelendiğinde "Ağız Bakım" ve "Bebek Bezi" kategorisindeki ürünleri Tablo 4.4'te görüldüğü üzere toplam fatura sayısının %36'sında eşleştiği görülmektedir.

Tablo 4.4: Destek değeri en yüksek olan ürünün birliktelikleri.

Kategori	Kategori	Güven	Destek
Ağız Bakım	Bebek Bezi	66%	36%
Bebek Bezi	Saç Bakım	63%	35%
Deterjan	Saç Bakım	69%	35%
Deterjan	Bebek Bezi	66%	34%
Deterjan	Ağız Bakım	65%	33%
Ağız Bakım	Saç Bakım	60%	31%
Ağız Bakım	Bebek Bezi	58%	30%
Ağız Bakım	Saç Bakım	55%	30%
Tıraş Bakım	Ağız Bakım	76%	30%
Tıraş Bakım	Ağız Bakım	64%	25%

Tablo incelendiğinde toplam fatura sayısının %30'unda bulunan "Tıraş Bakım" ve "Ağız Bakım" ürünleri birlikte bulunmaktadır. Ayrıca "Tıraş Bakım" ürünlerinin satışlarının

%76'sında "Ağız Bakım" ürünleri ile birlikte satılmıştır. Tabloda 1.sütunda ki kategoriyi alınan faturalarda 2. sütundaki ürünlerin birlikte satın alınma oranları verilmiştir.

Bu tez çalışmasında yapılan Pazar Sepeti Analizi ile ürün birliktelikleri, Hortonworks HDP 2.3 ile Hadoop platformu üzerinde MapReduce metodolojisi uygulanarak HDFS'te tutulan dosyalar üzerinde bulunan verileri kullanarak bulunmuştur. Bulduğumuz ürün birliktelikleri, bu çalışmada verileri kullanılan şirket ile paylaşılmıştır. Çalışmada elde edilen ürün birliktelikleri ilgili şirketin üç farklı birimde kullanılacağı değerlendirilmektedir.

Bu birimlerden ilki ve en önemlisi satış birimidir. Satış biriminde sahada aktif satış yapan satış temsilcileri tabletler ve bu tabletler üzerinde koşan mobil uygulamalar kullanmaktadırlar. Bu mobil uygulama ile çalışma sonucunda bulduğumuz analiz sonuçları adapte edilerek daha etkin satış teknikleri geliştirilmesi mümkündür. Ürün birliktelik sonuçları ve mobil uygulama entegrasyonu yılsonunda tamamlanması planlanmaktadır.

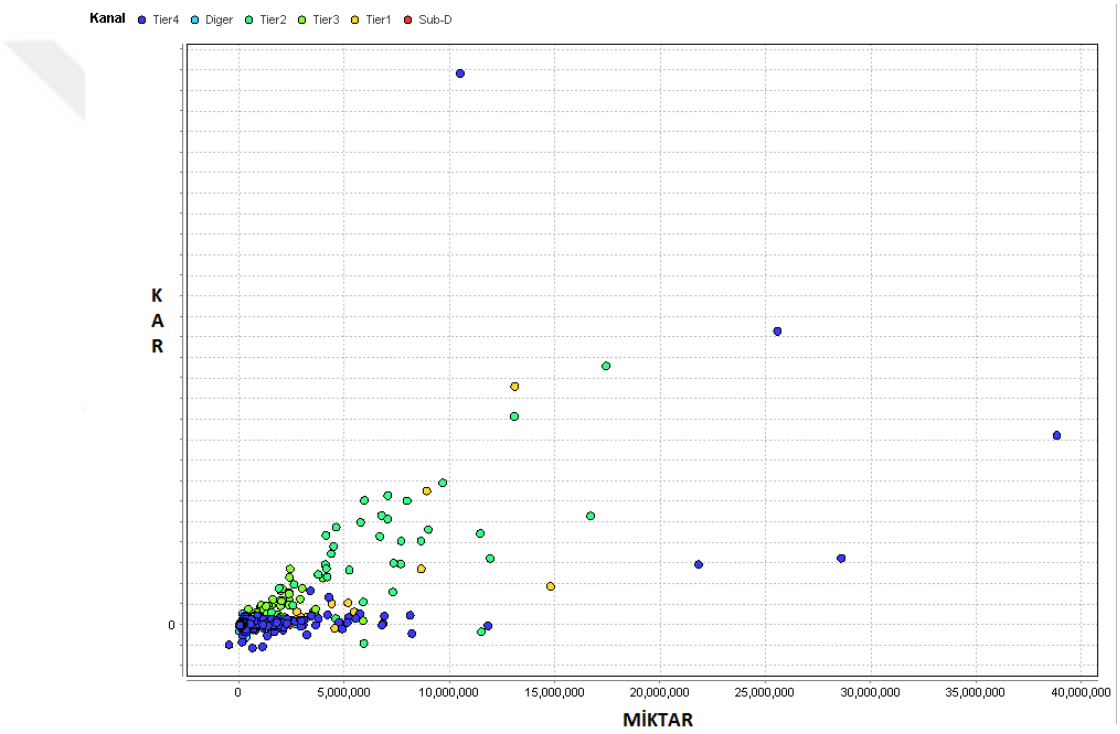
İkinci olarak bu analiz sonuçları Ticari Pazarlama birimi tarafından kullanılarak daha etkin özendirme (promosyon) ve kampanyalar üretilebilmektedir. Son olarak yine Lojistik birimi tarafında n bu veriler değerlendirilerek ürün birlikteliklerini de göz önüne alarak depolarda ürün yerleştirmesi yapılarak depo içinde her bir sevkiyat için yapılan ürün toplama işlemleri basitleştirilmiş olur. Bu sayede Güven seviyesi yüksek olan yani birlikteliği yüksek olan ürünlerin toplanması depodaki maliyetleri de azaltacaktır.

4.2. K-MEANS İLE GRUPLAMA

Kümeleme işlemleri için seçilen k-means algoritması kullanılarak öncelikle müşterilerin, satış miktarı ve karlılığa göre gruplanması planlanmıştır. Fakat öncelikle kümenin orta nokta sayısı olan k değeri belirlenmelidir. Bunun için k değerin belirlenmesi yapılacak analize göre belirlenmektedir. FMCG sektöründe müşteriler belirli ölçütlere göre gruplanırlar ve bu gruplara göre ticari şartlar ve uygulanan kampanyalar belirlenmektedir. Müşteri gurupları (kanal) belirlenirken müşterilerin büyüklük, alım yaptıkları miktarlar, karlılık oranları gibi ölçütlere bakılmaktadır. Çalışmada kullanılan verilerde bulunan müşteriler temel 4 kanala ayrılmaktadır bunlar aşağıdaki gibidir,

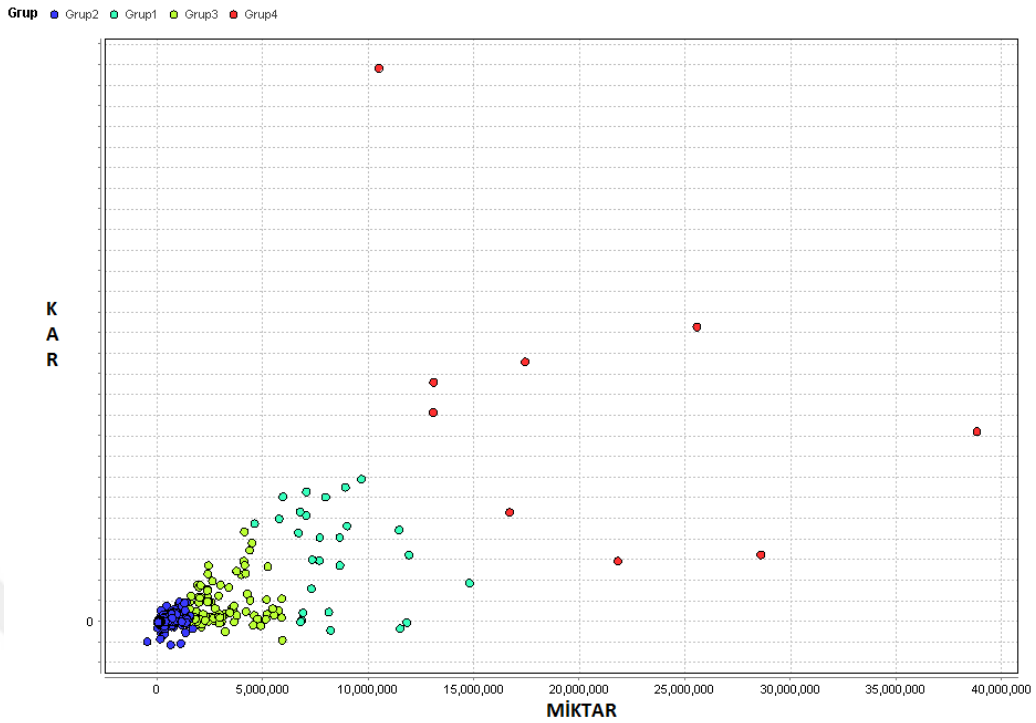
- Tier1
- Tier2
- Tier3
- Tier4

Yukarıda belirtilen kanalların yanında Sub-D diye adlandırılan alt dağıtım şirketleri ve kategorize edilmemiş müşteriler vardır ve bu müşterilerin gurubu Diğer olarak kabul edilmiştir. Müşterilerin kanal bazında karlılık ve miktara göre dağılımı Şekil 4.6'da gösterildiği gibidir.



Şekil 4.5: Analiz öncesinde müşteri kanallarının karlılık ve miktar dağılımı.

Yukarıdaki bilgiler ışığında k küme sayısı bu analize 4 olarak kabul edilmiş ve analiz yapılmıştır. Yapılan karlılık ve miktar analizi 7.193 müşteri için yapılmış ve 18 tekrar sonucunda orta noktaların değişmediği gözlenmiştir. Bunun neticesinde oluşan yeni gruplama Şekil 4.7'deki gibi ortaya çıkmaktadır.



Şekil 4.6: K-means çalışması neticesinde oluşan yeni gruplar.

Bu sonuçlara göre k-means ile yapılan analiz neticesinde müşterilerin bazılarının olması gerekenden farklı kanallarda çalıştırıldığı ve bu kanalların düzeltilmesinin uygulanacak ticari şartlar açısından belirli kazanımlar sağlayacağı ve müşteri memnuniyetinin oluşacağı sonucuna ulaşılmıştır.

5. SONUÇ

Teknolojinin geldiği noktaya bakıldığında sürekli bir bilgi üretme ve bilgiyi depolama çağında yaşamaktayız. Bilgilerin depolanması birçok kurum ve kuruluş için bir zarurettir. Bu bilgiler veritabanlarında veri olarak farklı formatlarda tutulmaktadırlar. Fakat bu verilerin işlenip bilgiye dönüştürülmesi ve kullanılması artık eskisi kadar kolay değildir. Çünkü veri boyutu hem çok fazladır hem de el yordamıyla incelenmeyecek kadar karmaşıktır. Büyük veriler incelenip bir sonuca ulaşıldığında toplumsal hayatı etkileyecek sonuçlar elde edilmeye başlanmıştır. Günümüzde büyük veri, iş, sağlık, eğitim, sosyal yapı ve politika hakkında düşünceleri değiştirmeye başlamıştır.

Büyük verinin devasa boyutlara ulaştığı ve şirketlerin bu verileri analizi için ciddi kaynak harcadığı günümüzde, şirketler verileri analiz ederek ilk bakışta görünmeyen verilere ulaşmaya çalışmaktadırlar. Bu bağlamda çalışmada, verileri inceleyerek ilk bakışta görünmeyen verilere erişmek amaçlanmıştır. Ayrıca yapılan çalışma ile veri boyutu ve düğüm sayısına göre ölçümler yapılmıştır. Sıradan makinalar ile Hadoop üzerinde analiz yapılmasının getirdiği kazanımları ve süre olarak yapılan kazanımlar tartışılmıştır. Tez çalışmasında hızlı tüketim sektöründe üretilen büyük veriler üzerinde birliktelik kuralı ve kümeleme analizleri yapılmıştır. En önemli tartışma konularından biri Pazar Sepeti Analizinin, MapReduce modeline nasıl uyarlanacağıdır. Tartışma konularından diğeri ise paralel işleme ile elde edilecek kazançtır. Çalışmada kullanılan veri setleri sırası ile 5, 10, 20, 40 ve 80 milyon satırlık satış verilerini barındırmaktadır. Bu satış verilerinin analizi sırası ile 1, 2, 4, 6 ve 8 Hadoop kurulmuş makinadan oluşan bir küme ile yapılmıştır.

Çalışmada temelde Woo'nun [26] yapmış olduğu çalışma temel alındığı için hem yapısal veriler, hem de yapısal olmayan veriler üzerinde analizler yapılmıştır. Yapısal olmayan verilerde iki farklı görev kullanılmıştır. Birinci görev yapısal olan veri setini yapısal olmayan hale getirmek için kodlanmıştır. İkinci görev ise yapısal olmayan veri setinin MapReduce ile Pazar sepeti analizi için kullanılmıştır. Sonuçlar ilgili bölümde bahsedildiği gibi en iyi kazanım 8 düğümlü yapıda sağlanmıştır.

İkinci analiz olarak tek bir görev ile yapısal veri ile yapılan analizdir. Bu analizde ise temel farklılık map ve reduce görevlerinin arasına combiner yani bir nevi ara reduce işlevi

koyulmuştur. Bu sayede görevin çalıştırılması ve tam bir paralel işlemenin devreye alınması sağlanmıştır. Aynı zamanda Woo'nun [26] çalışmasına oranla daha düşük düğüm sayısında daha az sürelerde verilerin analiz edildiği ve sonuca ulaşıldığı görülmüştür.

Bir makine ile yapılan denemelerde analiz süresinin yaklaşık olarak veri sayısının artışı oranında olduğu gözlenmiştir. Fakat paralel çalışma için Hadoop kümesindeki düğüm sayısı arttırıldığında belirli kazanımlar sağlanmıştır. Örneğin 5 milyon satırlık veri setinde çalışma yapıldığında en verimli süre 2 düğüm ile elde edilmişken 10, 20 milyon satırlık veri setleri ile çalışıldığında 4 düğüm ile daha kısa sürelerde analiz tamamlanmıştır. 40 Milyonluk veri setinde 6 düğüm, 80 milyonluk veri setinde 8 düğüm sayısı ile en optimum sonuçlar elde edilmiştir. Ulaşılan sonuçlara bakıldığında bu çalışmada kullanılan veri seti ve boyutlarına bağlı olarak verimli düğüm sayısı veri miktarına göre değişmekte olduğu görülmektedir. Bir makine ile 40 milyon satırlık veri setinde yapılan analizlerde elde edilen analiz süresi 2.431 saniye olarak ölçülmüştür. Aynı veri boyutunda 6 makine ile paralel olarak yapılan analiz süresi 618 saniye olarak elde edilmiştir. Buna göre 6 makine ile yapılan çalışmanın bir makine ile yapılan ölçümlere göre daha başarılı olduğu ve Hadoop üzerinde paralel işleme ile süreden %74 kazanç sağlandığı görülmüştür.

Çalışmada k-means algoritmasının MapReduce metoduna uyarlanması ve etkin analizlerin yapılabilirliği incelenmiştir. Yapılan incelemelerde k-means algoritmasının uyarlanması başarılı bir şekilde gerçekleştirilmiştir. Ayrıca elde edilen ve sonuçların analizi neticesinde mevcut müşteri gruplarının oluşturulma dinamikleri yerine k-means ile müşteri gruplarının oluşturulmasının karlılığı artıracağı ve daha net sonuçlar üreteceği gözlemlenmiştir.

Bu tez çalışması ile ilgili olarak gelecekte yapılması planlananlar aşağıdaki gibi belirlenmiştir. Üretici temelinde birbirinden bağımsız olarak pazar sepeti analizlerinin yapılması ve elde edilen ürün birliktelik sonuçlarının saha satış uygulamasına entegrasyonu, birliktelik kurallarının farklı algoritmalarının MapReduce tekniğine uyarlanması ve karşılaştırılması, müşterilerin ödeme alışkanlıklarına ve satın alma miktarlarına göre kümeleme algoritması çalışılması ve günlük satış tahminlerinin yapılması.

KAYNAKLAR

[1]. Marr, B., http://www.forbes.com/sites/bernardmarr/2015/09/30/big- data-20-mind-boggling-facts-everyone-must-read/#e36afaf6c1d3 [Ziyaret Tarihi: 4 Şubat 2016].

[2]. Gobble, M. A. M., 2013, Big Data: The Next Big Thing in Innovation, ResearchTechnology Management, January-February: 64-66.

[3]. Argüden, Y, Erşahin B., 2008, Veri Madenciliği, Arge Danışmanlık Yayınları, İstanbul, 978-975-93641-9-9

[4]. Ohlhorst, F., 2013, Big data analytics : turning big data into big Money, New Jersey.

[5]. Cukier, K., 2010,Data, data everywhere: A special report on managing information, The Economist Newspaper.

[6]. Gantz, J. and Reinsel, D., 2013, Digital Universe in 2020 in United States, USA.

[7]. SMC, Small and Midsize Companies Look to Make Big Gains With 'Big Data' According to Recent Poll Conducted on Behalf of SAP, http://global.sap.com/corporateen/news.epx?PressID=19188, [Ziyaret Tarihi: 14 Şubat 2016].

[8]. Schönberger, V. M. ve Cukier, K., 2014, Big Data: A Revolution That Will Transform How We Live, Work, and Think, London.

[9]. Rubistein, I.S.,2013, Big Data: The end of privacy or a new beginning?. Inter-national Data Privacy.

[10]. Beyer, M. A. ve Laney, D.,2012, The importance of big data: A definition. Stamford, Gartner.

[11]. Dumbill, A., 2013, Making Sense of Big Data, Big Data.

[12]. Gandomi, A. and Haider, M., 2015, Beyond the hype: Big data concepts, methods, and analytics, International Journal of Information Management, 35:137-144.

[13]. The World Factbook, 2013, https://www.cia.gov/library/publications/download/download-2013/", USA.

[14]. Cukier, K., 2010, "Data, data everywhere: A special report on managing information", The Economist Newspaper, February.

[15]. Kaisler, S., Armour, F., Espinosa, J. A., Money, W., 2013, Big data: Issues and challenges moving forward, 2013 46th Hawaii International Conference on System Sciences (HICSS), IEEE.

[16]. Arinto Murdopo, Jim Dowling, Next Generation Hadoop: High Availability for YARN http://www.slideshare.net/arinto/next-generation-hadoop-high-availability-for-yarn, [Ziyaret Tarihi: 7 Nisan 2016].

[17]. White, T., 2012, Hadoop: The Definitive Guide , Third Edition, UK, s.45-46

[18]. Borthakur, D., 2008, HDFS Architecture Guide, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf, [Ziyaret Tarihi: 2 Nisan 2016].

[19]. Miner, M., Shook A., 2013, MapReduce Design Patterns, Chaprter 2, Hendricson, M, CA, 20-61.

[20]. Dinçer E., 2006, Veri Madenciliğinde K-means Algoritması ve Tıp Alanında Uygulanması, p. 24-64.

[21]. Frawley, W. J., Gregory, PiatetskyShapiro, Matheus, Christopher J., 1991, Knowledge Discovery in Databases: An Overview. California: AAAI Press Copublications.

[22]. Agrawal, R. Ve Shafer, J.C., 1996, "Parallel mining of association rules: Design, Implementation and Experience", IBM Research Report RJ.

[23]. Han, J. ve Kamber, M., 2000, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, Burnaby.

[24]. Dinçer E., 2006, Veri Madenciliğinde K-means Algoritması ve Tıp Alanında Uygulanması, p. 24-64.

[25]. Han, J. ve Kamber, M., 2000, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, Burnaby.

[26]. Woo, J., 2012, Market Basket Analysis Algorithm on Map/Reduce in AWS EC2, Los Angeles

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı	Serdar ÇETİNKAYA
Uyruğu	Türkiye Cumhuriyeti
Doğum tarihi, Yeri	20/10/1983, Akyaka
Telefon	0543 695 35 65
E-mail	Sserdarcetinkaya@gmail.com

Eğitim

Derece	Kurum/Anabilim Dalı/Programı	Yılı
Yüksek Lisans	İ.Ü. Fen Bilimleri Enstitüsü/Bilgisayar Mühendisliği Anabilim Dalı/ Bilgisayar Mühendisliği Programı	2013-2016
Lisans	Kırgızistan – Türkiye Manas üniversitesi	2001-2006
Lise	Esenyurt Lisesi	1997-2000