T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

K-MEANS İLE DBSCAN ALGORİTMASI'NIN PARALELLEŞTİRMESİ VE HADOOP ÜZERİNDE BÜYÜK VERİ ANALİZİNDE KULLANILMASI, PERFORMANS VE YETERLİLİK KARŞILAŞTIRMASI

Yüksek Lisans Tezi

Tezi Hazırlayan:
Furkan KAYIM

İSTANBUL, 2015

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

K-MEANS İLE DBSCAN ALGORİTMASI'NIN PARALELLEŞTİRMESİ VE HADOOP ÜZERİNDE BÜYÜK VERİ ANALİZİNDE KULLANILMASI, PERFORMANS VE YETERLİLİK KARŞILAŞTIRMASI

Yüksek Lisans Tezi

Tezi Hazırlayan:
Furkan KAYIM

Öğrenci No:
130820014

Danışman:
Doç. Dr. Gökhan SİLAHTAROĞLU

İSTANBUL, 2015

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum **"K-means ile DBSCAN Algoritması'nın Paralelleştirmesi ve Hadoop Üzerinde Büyük Veri Analizinde Kullanılması, Performans ve Yeterlilik Karşılaştırması"** başlıklı çalışmanın, bilimsel ahlak ve geleneklere uygun bir şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmanın içinde kullanıldıkları her yerde atıf yapıldığını belirtir ve bunu onurumla doğrularım. 15/05/2015

Aday: **Furkan KAYIM**

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ/PROJE SAVUNMA SINAVI SONUÇ TUTANAĞI

Beykent Üniversitesi
Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Aşağıda tez/proje adı belirtilen yüksek lisans öğrencisi 130820146 no'lu FURKAN KAŞIM'in 06/2015 tarihinde yapılan tez/proje savunma sınavı[1] sonucunda 45 dakika süreyle sunduğu ve savunduğu tezi/projesi hakkında[2] oybirliğiyle, KABUL kararı verilmiştir.

Bilgilerinize saygılarımızla arz ederiz.

Anabilim Dalı : BİLGİSAYAR MÜH
Programı : BİLGİSAYAR MÜH
Tez/Proje Başlığı[3] :

Tez/Proje Sınav Jürisi	Öğretim Üyesi	İmza
Danışman	: Doç. Dr. Gökhan Silahtaroğlu	
Üye	: Doç Dr. Kazım Sarı	
Üye	: Yrd. Doç. Dr. Ediz Saykol	

---

[1] Jüri üyeleri söz konusu tezin kendilerine teslim edildiği tarihten itibaren en geç bir ay içinde toplanarak öğrenciyi tez savunma sınavına alır. Belirlenen günde yapılamayan jüri toplantısı, katılanların hazırladığı bir tutanakla enstitü yönetimine bildirilir. Bu durumda jüri en geç onbeş gün içinde toplanarak adayı tez savunma sınavına alır. Tez savunma sınav süresi en az 45 dakikadır. Yüksek lisans tez savunma sınavı, tez çalışmasının sunulması ve bunu izleyen soru-yanıt bölümlerinden oluşur ve dinleyiciye açıktır. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-3)

[2] Tez sınavının tamamlanmasından sonra jüri, tez hakkında "kabul", "düzeltme" veya "red" kararı verir. Jüri başkanı, jüri üyelerince imzalanmış sınav tutanağını, tez sınavını izleyen üç gün içinde ilgili enstitü yönetimine teslim eder. Tezi başarısız bulunan öğrencinin Enstitü ile ilişiği kesilir. Tezi hakkında düzeltme kararı verilen öğrenci en geç üç ay içinde gerekli düzeltmeleri yaparak ve yönetmelikte belirtilen usullere uygun olarak tezini aynı jüri önünde yeniden savunur. Bu savunma sınavında da tezi kabul edilmeyen öğrencinin enstitü ile ilişiği kesilir.(Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-4)

[3] İleride doğabilecek aksaklıklarınengellenmesiiçintezinbaşlığınınyazılmasıgerekmektedir.

TEŞEKKÜR

Eğitim hayatım boyunca hiçbir destekten kaçınmayan değerleri aileme ve çalışmam boyunca benden desteğini esirgemeyen Sayın Nurdan Bayraktar'a sonsuz teşekkürlerimi sunarım.

K-MEANS İLE DBSCAN ALGORİTMASI'NIN PARALELLEŞTİRMESİ VE HADOOP ÜZERİNDE BÜYÜK VERİ ANALİZİNDE KULLANILMASI, PERFORMANS VE YETERLİLİK KARŞILAŞTIRMASI

Tezi Hazırlayan: **Furkan KAYIM**

ÖZ

Hayatımızdaki her türlü eylem bilgisayar üzerinden yürütülmeye başlamıştır. Neredeyse tüm sektörlerde yapılan işin merkezine yerleşen bu teknoloji, iş süreçlerinin yürütülmesinde kolaylaştırıcı ve süreci hızlandırıcı adımları da gün geçtikçe gerekli kılıyor. Nitekim bilgisayar temelli çalışmalarda karşılaşılan aksaklıklar uzun vadede şirketlerin ciddi oranda kayıplar yaşamasına neden olurken; yaşanan bu olumsuzlukların özünü giderek büyüyen verilerin kontrolsüz yönetimi oluşturuyor.

Büyük verilerle yapılan her türlü işlem veri saklama, veri analizi, verilerin gösterimi gibi pek çok soruna neden olabilmektedir. Bu sorunlar başta veri kayıpları olmak üzere pek çok olumsuzlukla sonuçlanabilmekte, bu da alanda yapılacak çalışmanın gerekliliğini bir kez daha hissedilir kılmaktadır.

Bu tezde büyük verilerle çalışma yöntemleri araştırılarak, bu verilerle daha hızlı çalışma yapılabilmesi ve daha kararlı sonuçlar alınabilmesine yönelik uygulamalara yer verilmiştir. Bu bağlamda öncelikle büyük verilerle veri madenciliği algoritmalarının birlikte nasıl kullanılabilecekleri performans değerlendirmeleri ile ele alınmış ve veri madenciliği algoritmalarından DBSCAN ve K-means algoritmalarının paralelleştirmesi incelenmiştir. Ardından Hadoop teknolojisi araştırılarak Pig, Hive, Impala performans karşılaştırılması yapılmış, Hadoop teknolojilerinin hangi projelerde kullanılabileceği irdelenmiştir. Hadoop üzerinde veri madenciliği algoritmalarının Mahout ile çalıştırılabileceği de ayrıca görülmüştür.

Anahtar Kelimeler: Bulut Bilişim, Büyük Veri, Hadoop, K-means, DBSCAN, Impala, Hive, Pig, Mahout

PARALLELIZATION OF K-MEANS AND DBSCAN ALGORITHMS AND USE ON ANALYSIS OF BIG DATA ON HADOOP AND PERFORMANCE AND COMPETENCE COMPARISON

Presented By: **Furkan KAYIM**

ABSTRACT

All actions in our life are now carried out through computers. This technology, which is now at the center of businesses performed in almost all sectors, day by day necessitates steps that facilitate and accelerate performance of business processes. Indeed, faults encountered in computer-based works result in significant losses for companies in the long run; while uncontrolled management of growing data constitute the essence of these problems.

All operations performed with large data can cause many problems such as data storage, data analysis and data display. These problems may end up with many problems, especially data loss, and once again indicates the need for carrying out works in this field.

In this thesis, methods for working with big data have been explored and applications for performing fast operations and obtaining more stable results with such data have been provided. In this context, methods for joint use of data mining algorithms for large data have been primarily considered with performance evaluations and parallelization of data minin algorithms DBSCAN and K-means have been analyzed. The Hadoop technology has been analyzed and performance comparison has been made with Pig, Hive and Impala, also projects have been examined where Hadoop technologies could be used. It has also been observed that data mining algorithms on Hadoop can be used with Mahout.

Keywords: Cloud Computing, Big Data, Hadoop, K-means, DBSCAN, Impala, Hive, Pig, Mahout

İÇİNDEKİLER

ÖZ i
ABSTRACT ii
İÇİNDEKİLER iii
TABLOLAR DİZİNİ v
ŞEKİLLER DİZİNİ vi
1.GİRİŞ 1
2.VERİ MADENCİLİĞİ 4
2.1.Veri Madenciliği Tanımı 4
2.2. Veri Madenciliğinin Tarihi 5
2.3. Veri Madenciliğinin Kullanıldığı Alanlar 7
2.4. Veri Madenciliğinde Karşılaşılan Problemler 7
2.5. Veri Madenciliği Uygulaması 8
2.6. Veri Madenciliği Metotları 9
2.7. Veri Madenciliği Uygulama Alanları 9
2.8. Veri Ambarı 11
3. KÜMELEME ANALİZİ 14
3.1. Kümeleme Algoritmaları Tanımı 14
3.2. K-means Algoritması 14
3.2.1 K-means Algoritması Java Kodlama 15
3.3. DBSCAN Algoritması 22
3.3.1 DBSCAN Algoritması Avantajı 25
3.3.1 DBSCAN Algoritması Dezavantajı 25
4. HADOOP ve BÜYÜK VERİ 26
4.1 Büyük Veri Nedir? 26
4.2 Veri Tipleri 29
4.3.Büyük Verinin Zorlukları 29
4.4 Bulut Bilişim Nedir? 30
4.5 Bulut Bilişim Yatay Büyüme 30
4.6 Dağıtık Hesaplama (Distributing Computing) 30
4.7 Hadoop Nedir? 31
4.8 Hadoop Büyük Veri Problemini Nasıl Çözer? 31
4.9 Hadoop Kullanıcıları 32

4.10 Temel Hadoop Cluster Yapısı 33
4.12 Hadoop ve Rdbms Kıyaslaması 33
4.13 HADOOP Ekosistemi 35
4.13.1 Pig 35
4.13.1.1 Pig Özellikleri 35
4.13.1.2 Pig Kalıp Kod 36
4.13.1.3 Pig Avantajları 36
4.13.2 Hive 37
4.14.1 Hive Özellikleri 37
4.14 Hive ile Pig Karşılaştırması 37
4.15 Impala 38
4.16 HUE 38
4.17 Hadoop Distributed File System(HDFS) 39
4.18 HBase 39
4.19 Apache Mahout 40
4.20 FLUME 42
4.21 Sqoop 42
4.22 Oozie 43
4.23 Hadoop Teknolojilerinden Nerede Ne Kullanılmalı? 43
4.24 Hadoop için örnek bir kullanım senaryosu 44
4.25. Hadoop – MapReduce Mimarisi 44
4.25.1 Map fonksiyonu 47
4.25.2 Reduce fonksiyonu 48
5. ANALİZ SONUÇLARI 50
5.1 Hadoop Impala, Hive Performanslar Değerlendirmesi 50
5.2 Hive, Impala, İlişkisel Veri Tabanı Performans Testi 53
5.3 K-means Paralleştirme Performanslar Değerlendirmesi 54
5.4 Mahout ile K-means Algoritmasının Çalıştırılması 56
5.5 DBSACAN Algoritması Paralelleştirme Performans Değerlendirmesi 56
SONUÇLAR VE ÖNERİLER 58
KAYNAKLAR 61
EKLER 66

TABLOLAR DİZİNİ

Tablo 1. Mahout 0.10.0 özellikleri [37] 40

Tablo 2. Server Özellikleri ...……..... 50

Tablo 3. Hadoop Impala ve Hive performans testi tablo boyutları ……..... 50

ŞEKİLLER DİZİNİ

Şekil 1. Veri madenciliği ve bazı disiplinleri 4
Şekil 2. Veri madenciliğinin tarihsel süreci [9] 6
Şekil 3. Örnek Satışlar Tablosu (Veri tabanı) [19] 11
Şekil 4. Örnek Veri Ambarı Parçası [19] 12
Şekil 5. Veri tabanlarında bilgi keşfi aşamaları [18] 12
Şekil 6. K-means algoritmasının adımları [23] 15
Şekil 7. DBSCAN algoritmasında farklı "K" değerleri için elde edilen kümeler [25]... 24
Şekil 8. Metrikler 26
Şekil 9. Büyük veri ile bağlantılı terimler [27] 27
Şekil 10. Verilerin dağılımı 27
Şekil 11. Büyük veri 28
Şekil 12. Veri Tipleri 29
Şekil 13. Bulut Bilişim[28] 30
Şekil 14. Dağıtık Hesaplama 31
Şekil 15. Temel Hadoop cluster yapısı 33
Şekil 16. Hadoop Rdms kıyaslaması 33
Şekil 17. Pig kalıp Kod[58] 36
Şekil 18. Hive ile Pig Karşılaştırması [33] 38
Şekil 19. Hue 39
Şekil 20. Sqoop[39] 43
Şekil 21. Oozie 43
Şekil 22 Hadoop için örnek bir kullanım senaryosu 44
Şekil 23 MapReduce kelime sayısı bulma[59] 46
Şekil 24. Impala ve Hive 16GB Tablo boyutu ile 52
Şekil 25. Impala ve Hive 32GB Tablo boyutu ile 52
Şekil 26. Hive, Impala,PL/SQL performans değerlendirme 544
Şekil 27. K-means 15 cluster ile paralelleştirme 546
Şekil 28. DBSCAN paralelleştirme performansları 57
Şekil 29 Hadoop Ekosistemi [51] 66

1. GİRİŞ

Günümüzde teknolojinin hızla gelişmesi ile birlikte veri miktarı hızla artmakta, yazılımın olmadığı bir alana ise oldukça nadiren rastlanmaktadır. Televizyonlar, elektrikli süpürgeler, buzdolapları, arabalar ve uydular gibi teknoloji ürünü olan tüm araçlar tamamen yazılım ürünüdür. Yazılımın hayatımızın her alanında karşımıza çıkması ve bunun bir getirisi olarak yoğun veri dolaşımı kaçınılmazdır.

Verilerin bu kadar çok kullanılması ihtiyaçları da farklılaştırmıştır. Bu ihtiyaçlardan biri de eldeki verileri kullanarak geleceğe yönelik tahminlerde bulunulması sürecidir. Veri madenciliği ifadesiyle kavramsallaştırılan bu süreç, algoritmaların eldeki verilerle analiz edilerek geleceğe yönelik tahminler sunulmasını sağlamaktadır. Veriler arası ilişkilerin tespitinde pek çok fonksiyon bulunduğu gibi verilerin kümelenmesinde veri madenciliği algoritmaları kullanılabilmektedir. Nitekim hayatımızdaki hızla artan verilerin saklanması ve işlenmesinde ortaya çıkan problemler bu algoritmalarla giderilebilmektedir.

Cisco, Görsel Ağ Endeksi'nin (VNI) 2014- 2019 Küresel Mobil Veri Trafiği Öngörü Raporu'na göre gelecek dört yıl içerisinde dünya çapında mobil veri trafiğinin 30 eksabayttan yılda 292 eksabayta ulaşması beklenmektedir [1]. Giderek artan ve mobil alanı da aşarak "büyük veri" olarak ifade edilen bu verilerin büyük oranının ise son iki yılda ortaya çıktığı görülmektedir.

Alt yapısal olarak büyük verinin saklanması ve işlenmesi için büyük donanımlara ihtiyaç duyulmakta, büyük veriler ile uğraşılan uygulamalarda da performans problemleri ile karşılaşılmaktadır. Bu süreçte verilerin saklanmasında ve işlenmesinde dev sunucular, veritabanları ihtiyaçları karşılamaya yardımcı olsa dahi bunların da yetersiz olduğu noktalar bulunmaktadır. Bu noktada oluşan ihtiyaçlar Hadoop, Hive, Impala gibi teknolojilerle karşılanmıştır.

Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System (HDFS)) büyük boyutlu verilerin dağıtık olarak kaydedilmesi ve işlenmesi için kullanılan sistemlerden biridir [2] ve büyük veri setlerini işlemek için yüksek maliyetli süper

bilgisayarların kullanımı yerine standart bilgisayarlardan oluşan bir ağ üzerinde bu sistemin kullanılması önem kazanmıştır. Bunun amacı ise, kaynak kullanımını optimize etmek ve maliyeti düşürmektir [3].

Büyük veri problemine yardımcı olan teknolojilerden biri olan Hadoop, pahalı ve güçlü bilgisayarları kullanmayı zorunlu tutmayan ve yatay olarak büyüyen bir yapıdır. Bu yapı büyük veriye alt yapı sağlamaktadır. Birden fazla bilgisayarın tek bir bilgisayar gibi mevcut problemi çözmesine sağlayan bir teknoloji olan Hadoop ile bir problem birden fazla parçalara ayrılır ve her bir parça farklı bilgisayarlar tarafından çözülmüş olur.

Hadoop'un genelde web tabanlı verilerin dağıtık olarak kaydedilmesini ve kaydedilen veriler üzerinde aramalar yapılmasını sağlama amaçlı kullanıldığı görülmekte [2], bu şekilde elde edilen veriler HDFS'ye kaydedildiğinde indekslenerek ve sıkıştırılarak çok büyük boyutlu dosyalar halinde kaydedilmektedir. Yani Hadoop çok büyük boyutlu dosyalar ile çalışmak için optimize edilmiştir [4].

Bu çalışmada ilk olarak popüler kümeleme algoritmalarından K-means ve DBSCAN algoritmasının paralelleştirilmesinin performans sonuçları ile Hadoop'ta Impala, Hive, Pig gibi teknolojilerin performans sonuçları karşılaştırmalı olarak incelenecektir. Bu sayede yapılacak veri madenciliği algoritmalarının paralelleştirilmesi ile performansının artırılması hedeflenmektedir. Hadoop'un paralelleştirmeyi kolaylaştıran bir ortam hazırlarken, Hive ve Impala gibi sorgularımızı çalıştırabilme durumu irdelenecektir.

Çalışmanın ikinci bölümünde veri madenciliği tanımı ve terimleri hakkında bilgi verileceği gibi, veri madenciliğine neden ihtiyaç olduğu ve tarihi süreci detaylandırılmıştır. Veri ambarı ile veri ambarının kullanım nedenlerine ise ayrıca yer verilmiştir.

Çalışmanın üçüncü bölümünde veri madenciliği tekniklerinden kümeleme analizleri hakkında bilgi verilmiş, kümeleme analizi tekniklerinden K-means, DBSCAN algoritmaları incelenmiştir.

Çalışmanın dördüncü bölümünde Hadoop ve büyük veri kavramları incelenmiş, "büyük veri nedir" sorusuna cevap aranmıştır. Bununla birlikte Hadoop temel mantığı üzerinde durularak Hadoop teknolojileri incelenmiş, bu teknolojilerin kullanım alanları ve nerelerde kullanılmaları gerektiği sorgulanmıştır. Aynı alanda kullanılan Hadoop teknolojilerinin ise kıyaslanması yoluna gidilmiştir.

Çalışmanın beşinci bölümünde büyük veri problemine performans değerlendirmeleriyle bakılmış, büyük veri probleminin çözümünde veri madenciliği algoritmalarının paralelleştirilmesi ve Hadoop gibi teknolojiler kullanılması çözüm yolu olarak incelenmiştir.

Çalışmanın sonunda sonuç ve önerilere yer verilmiş, özellikle tezin beşinci bölümünden elde edilen performans değerlendirmeleri ve yapılan kıyaslamaların özü sunulmuştur.

2.VERİ MADENCİLİGİ

2.1 Veri Madenciliği Tanımı

Veri madenciliği, geleceğin tahmin edilebilmesi için gerekli bağlantı ve kuralların programlar sayesinde bulunması ve analiz edilmesidir. Veri madenciliği çok büyük miktardaki verilerin arasındaki ilişkiyi inceleyerek birbirleri ile bağlantılarını çıkarmaya yardımcı olur. Aynı zamanda veri tabanları içerisinde gizli kalmış bilgilerin çıkarılmasına yardımcı olan verilerin analiz edilmesini sağlayan teknikler bütünüdür.

Veri madenciliği tekniklerinin uygulama alanları oldukça geniştir. Bu alanlardan bazıları aşağıdaki Şekil 1'de gösterildiği üzere "istatistik, yapay öğrenme, veri tabanı sistemleri, veri görselliği, yapay sinir ağları" gibi disiplinler bulunmaktadır.

Şekil 1. Veri madenciliği ve bazı disiplinleri

Veri madenciliği tekniklerinin kullanılması ile verilerle iş yapan işletmelerin daha etkin kararlar almasına yardımcı olacak karar destek maddelerinde iyileştirmeler yapılabilmektedir. Veri madenciliği sistemleri kullanılmadan önceki zamanlarda klasik karar destek sistemlerinin kullanıldığı araçlardan faklı olarak bu

tekniklerle çok daha kapsamlı ve otomatize edilmiş analizler yapmaya yönelik birçok farklı özellik bulunmaktadır.

Veri madenciliği tekniklerinin kullanılması ile işletmelerin kullanmaya başladığı en önemli özellik, veri grupları arasındaki benzer eğilimlerin ve davranış kalıplarının belirlenmesidir. Bu süreç otomatize edilmesine paralel bir şekilde hayata geçirilebilmektedir. Veri madenciliği tekniklerin bu fonksiyonu genellikle hedef pazarlara yönelik pazarlama faaliyetlerinde kullanılabilmektedir. Yaygın olarak kullanılan bir başka özelliği de yeni ortaya çıkmış veri ambarları içerisinde bulunan ve ilk etapta görülemeyen bilgilerin ortaya çıkarılmasıdır.

Bugüne kadar farklı kaynaklarda veri madenciliğinin pek çok tanımıyla karşılaşılmıştır. Bu kaynaklardan bazılarına göre veri madenciliğinin tanımı şöyledir:

- Veri madenciliği, ham datanın tek başına sunamadığı bilgiyi çıkaran, veri analizi sürecidir. [5]
- Veri madenciliği, büyük veri yığınları arasından gelecekle ilgili tahminde bulunabilmemizi sağlayabilecek bağlantıların, bilgisayar programı kullanarak aranması işidir. [6]
- Veri madenciliği istatistik, veritabanı teknolojisi, örüntü tanıma, makine öğrenme ile etkileşimli yeni bir disiplin ve geniş veritabanlarında önceden tahmin edilemeyen ilişkilerin ikincil analizidir [7].
- Veri madenciliği, oldukça tahminci anahtar değişkenlerin binlerce potansiyel değişkenden izole edilmesini sağlama yeteneğidir [8].

Bu tanımlardan bakarak şu şekilde bir tanım yapmak mümkündür: Veri madenciliği, çok büyük miktarda verinin depolandığı veri tabanlarından, hedefler doğrultusunda, gelecek ile ilgili tahminler yapmamızı yarayan, anlamlı olan veriye ulaşma ve veriyi kullanma işidir.

2.2. Veri Madenciliğinin Tarihi

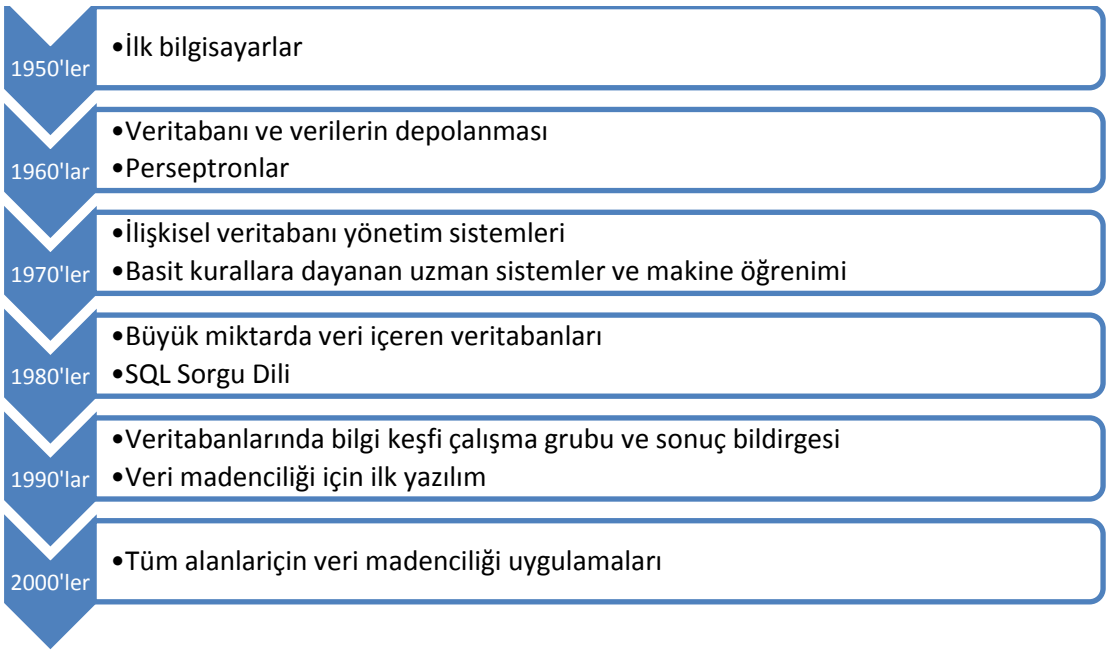
Günümüzde neredeyse her eve bilgisayar girmiştir ve internet erişimi hemen hemen her yerden sağlanmaktadır. Disk kapasitelerinin artması, her yerden bilgiye

ulaşma olasılığı, bilgisayarların çok büyük miktarlarda veri saklamasına ve daha kısa sürede işlem yapmasına olanak sağlamıştır. Geçmişten günümüze veriler her zaman yorumlanmış, bilgi elde etmek istenmiştir ve bunun için de donanımlar oluşturulmuştur [9]. Bu sayede bilgi, geçmişten günümüze taşınır hale gelmiştir.

50'li yıllarda bilgisayarlar günlük hayatımızda sayımlar için kullanılmaya başlamıştır. 60'lı yıllardan sonra ise artık veri kaydetmek içinde kullanıldığı görülmektedir.

Minsky ve Papert, günümüzde sinir ağları olarak bilinen perseptron'ların sadece çok basit olan kuralları öğrenebileceğini göstermişlerdir [10]. 1970'lerde İlişkisel Veri Tabanı Yönetim Sistemleri uygulamaları kullanılmaya başlanmıştır. Bilgisayar uzmanları bununla beraber basit kurallara dayanan uzman sistemler geliştirmiş ve basit anlamda makine öğrenimini sağlamışlardır [9].

Günümüzde veri madenciliğine bilgisayarların tüm alanlarda kullanılmaya başlamasıyla birlikte ihtiyaç artmıştır. Aşağıdaki veri madenciliğinin tarihi gelişimini anlatan şekilden de anlaşılacağı üzere veriler her alana giderek artmıştır. Bu da verilerin saklanması ve işlenmesi için veri madenciliği tekniklerine olan ihtiyacını artırmıştır.



Şekil 2. Veri madenciliğinin tarihsel süreci **[9]**

2.3. Veri Madenciliğinin Kullanıldığı Alanlar

Büyük hacimde veri bulunan her yerde veri madenciliği kullanmak mümkündür [11]. Günümüzde karar verme sürecine ihtiyaç duyulan birçok alanda veri madenciliği uygulamaları yaygın olarak kullanılmaktadır [12]. Örneğin pazarlama, biyoloji, bankacılık, sigortacılık, borsa, perakendecilik, telekomünikasyon, genetik, sağlık, bilim ve mühendislik, kriminoloji, sağlık, endüstri, istihbarat vb. birçok dalda başarılı uygulamaları görülmektedir [13].

Bugün dünyada ve Türkiye'de hemen her sektörde veri madenciliği teknikleri kullanılmaktadır:

- Taşımacılık-Ulaşım sektörü
- Konaklama ve otelcilik sektörü
- Satış ve Pazarlama: Müşteri sınıflandırma, hedef müşteri belirleme ve benzeri tekniklerde
- Bankacılık: Kredi onaylama ve fraud kontrol gibi sistemlerin çoğunda
- Sigortacılık: Poliçe onaylama vb işlemlerde
- Üretim ve planlama alanının tamamına yakınında
- Sistem yönetimi ve yardım masası işlemlerinde
- Borsa
- Eğitim sektöründe

2.4. Veri Madenciliğinde Karşılaşılan Problemler

Veri madenciliği teknikleri kullanılırken büyük verilerde bazı problemlerle karşılaşılabilir. Bu problemlerden bazıları;

- Verinin çok fazla olması ver kirli verinin çok oluşu
- Veri miktarın artması ile birlikte donanım yetersizliği
- Bilgisayar ağlarındaki sorunlardan dolayı bilgi güvenliği sorunları
- Bilimsel hesaplamalarda verilerin kirli ve büyük olmasından kaynaklana problemler
- Ticari eğilimlerden dolayı karşılaşılan problemler

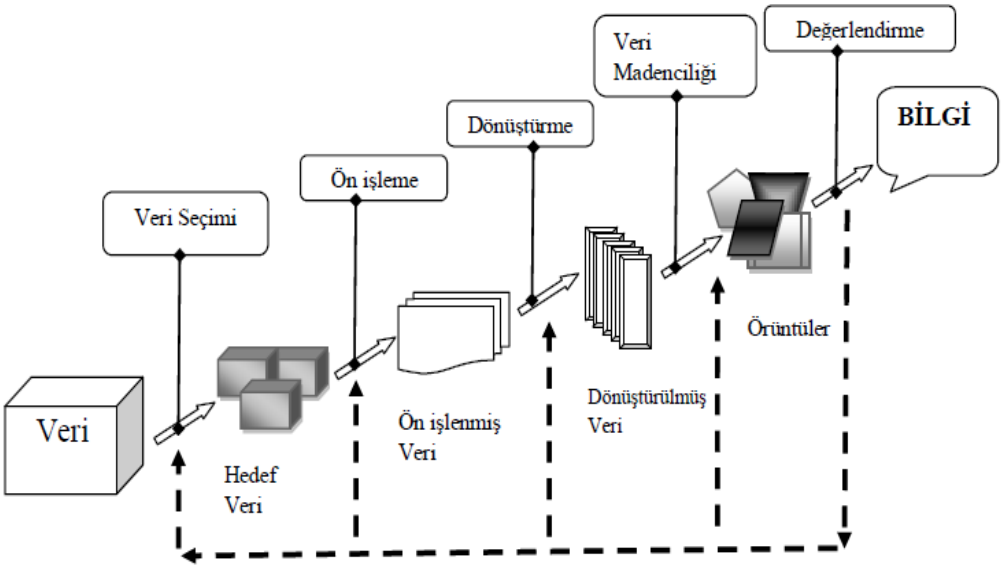
Veri madenciliği tekniklerinde karşılaşılan problemleri yukarıdaki gibi sınıflayabiliriz. Bu problemlerin her birinde karşımıza çıkan "atık veri" problemine veri ambarı ile çözüm getirilmeye çalışılmıştır. İlerleyen bölümlerde veri ambarına değinilecektir.

Gürültü ve kayıpları gidermek içinde ayrı veri madenciliği teknikleri kullanılmaktadır. Gürültüler ve kayıplar veri toplanması sırasında karşılaşılan sistemsel ve dış kaynaklardan dolayı verinin tam elde edilememesi problemidir.

Bir diğer problemde sınırlı veriye ulaşım problemidir. Veri madenciliği tekniklerin tam olarak uygulanması için tam ve kesin bilginin olması gerekmektedir. Fakat teknoloji her ne kadar gelişse de tam ve kesin bilgiyi elde etmek imkânsızdır.

2.5. Veri Madenciliği Uygulaması

Veri madenciliği tekniklerin uygulanması aşama aşama Şekil 2' de gösterilmiştir [14].



Şekil 3. Bilgi keşfi sürecinde veri madenciliği[14]

Veriden bilgi çıkarım aşaması yukarıdaki şekilde anlatılmıştır. Bu adımlardan bahsedecek olursak;

- Büyük veri içinden veri ayıklaması: Veriler çok fazladır bu veriler içerisinden işlem yapılmak istenen veriler ayıklanmalıdır.

- Ön işlenmiş veri: Veri ambarı ile istenilen veriler içerisinden kirli veriler ayıklanmalıdır. Eksik veriler tamamlanabiliyor ise tamamlanmalıdır.
- Dönüştürülmüş veri: Verilerin ayıklanması ve temizlenmesinden sonra elde edilen veri ile işlem yapılabilecektir.
- Veri madenciliği: Temizlenmiş ve ayılanmış veriye veri madenciliği teknikleri uygulanarak daha doğru sonuçlar elde edilebilir. Uygun veri madenciliği tekniği seçilerek istenilen bilgiye ulaşılabilir.

2.6. Veri Madenciliği Metotları

Veri madenciliği metotlarının sayısı her geçen gün artmakla birlikte mevcut metotlarda gelişmeye devam etmektedir. Bu metotlardan bir kısmı veri madenciliğinin yaygın kullanılan klasik metotlarda denilebilecek istatiksel veriler elde etmek için kullanılan metotlardır.

Yeni nesil yöntemlerde ise yine istatiksel analizleri temel alarak makine öğrenmesi ve yapay zeka gibi yöntemler görülmektedir.

Veri madenciliği metotları 3 modele göre ayrılabilir;

1. Sınıflama (Sınıflandırma) ve Regresyon (Gerileme),

2. Kümeleme (Kümeleme),

3. Birliktelik Kuralları (Association Rules),

Yukarıdaki 3 başlığa göre ayrılabilir. Sınıflama ve regresyon modelleri tahmin edici, kümeleme ve birliktelik kuralları modelleri tanımlayıcı modellerdir [15].

2.7. Veri Madenciliği Uygulama Alanları

Dünyada verilerin artması ile birlikte veri madenciliğine olan ihtiyaç ve talepte artmaktadır. Bilgisayar programlarının hemen her sektörde kullanılmasına ihtiyaç artmaktadır. Bilgisayar programları kullanılan sektördeki gelişmeler hızla artmaktadır. Tüm dünyada olduğu gibi ülkemizde de veri madenciliğine verilen

önem ve gösterilen ilgi her geçen yıl artmaktadır. Veri madenciliğinin kullanım alanları genişleyerek yayılmaktadır [9].

- **Mühendislikte Veri Madenciliği:**

Bu alanda yapılan çalışmalar mevcuttaki veri madenciliği algoritmalarını geliştirmek ve yeni veri madenciliği algoritmalarını geliştirmek üzerinedir.

- **Tıp alanında veri madenciliği:**

Bu alanda yapılan çalışmalar hastalık tahmini üzerine gerçekleşmektedir.

- **Bankacılık ve borsa alnında veri madenciliği:**

Bu alanda yapılan çalışmaları kendi içerisinde 4'e ayrılmaktadır. Bunlar; Pazarlama, Risk yönetimi, Fraud Detection, Customer Retention'dır. Veri madenciliği algoritmalarının en etkili kullanıldığı sektörlerden birisidir. Bu alanda müşteri ile ilgili çalışmalar, kredi çalışmaları, güvenlik çalışmaları gibi çalışmalar yapılmaktadır.

- **Eğitim alanında veri madenciliği:**

Eğitim alanında kullanılan veri madenciliği teknikleri daha çok örencilerin başarı oranını artırmaya yönelik çalışmaları içermektedir. Öğrencilerin başarı ve başarısızlıkları eldeki verilere göre önceden tahmin edilmeye çalışılmakta, mevcut durumda iyileştirmelerle yapılmaya çalışılmaktadır.

- **Ticari alanda veri madenciliği:**

Ticari alanda yapılan analizlerin çoğunluğu müşteri analizi ve pazar analizine yönelik çalışmaları içermektedir. Ticari alanda yapılan çalışmalar pazar payını büyültmeye yönelik çalışmalardır. Bu çalışmalar için eldeki müşteriyi kaybetmeme yeni müşteriler kazanma, müşteri memnuniyeti gibi amaçları vardır. Bununla birlikte son yıllarda hızla gelişen teknoloji, artan markalı ürün ve kurumlar meşruiyet kavramının önemini artırmış [16], kurumlar meşruiyet çabaları doğrultusunda da veri madenciliğini iş süreçlerine dâhil etmek durumunda kalmıştır.

- **Telekomünikasyon alanında veri madenciliği:**

Bu alanda yapılan çalışmalar yeni müşteri kazanımı ve eldeki müşterilerin ihtiyaçlarının tahminine yönelik çalışmalardır.

2.8. Veri Ambarı

Veri ambarları, veri madenciliği ile eşanlı olarak anılan ve veri madenciliği sürecinin gerçekleştirildiği veriyi sağlayan özel bir veri tabanıdır. Tanım olarak veri ambarı, pek çok farklı kaynaktan ve genellikle de farklı yapıda verinin depolandığı ve hepsinin de aynı birleşik çatı altında kullanılmasının ümit edildiği yapılardır. Ayrıca, veri ambarı pek çok farklı kaynaktan elde edilen veriyi aynı çatı altında analiz etme imkânı tanır [17].

Veri ambarları bir karar verme mekanizması veya diğer adıyla karar destek sistemi olarak kullanılmaktadır. Veri ambarı üzerinde veri madenciliği, çok boyutlu veri analizi (Online Analytical Processing - OLAP), müşteri ilişkileri yönetimi (CRM), istatistiksel analiz ve raporlama işlemleri gerçekleştirilir [18]. Veri ambarı normal veri tabanlarından farklı olarak daha hızlı performans işlemleri için kullanılır.

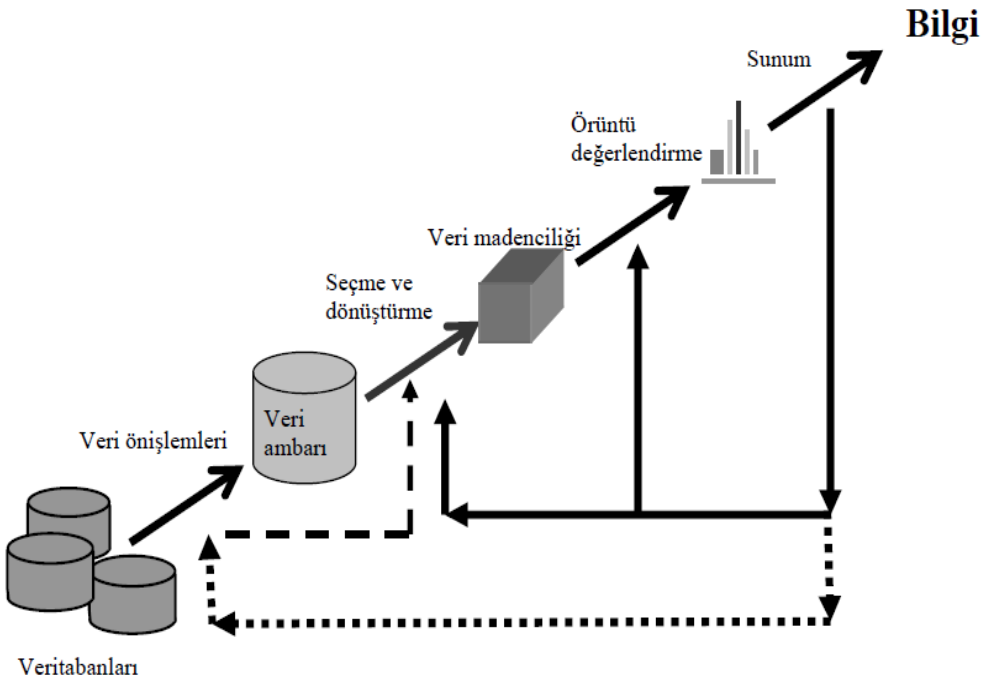
Müşteri ID	Ürün ID	İşlem No	Miktar	...
123	45600	1	1	
123	45602	1	1	
123	45601	1	2	...
kayıtsız	45602	2	1	
333	45601	3	3	...
...	...	...	...	...
...	...	...	...	...

Şekil 3. Örnek Satışlar Tablosu (Veri tabanı) **[19]**

Burçlar	Marka	Alışveriş Günü	Alışveriş Miktarı	...
Kova	Aaa	Pazartesi	255	
Oğlak	bbb	Pazartesi	523	...
...	...	...	...	...
...	...	...	...	...

Şekil 4. Örnek Veri Ambarı Parçası **[19]**

Veri madenciliği metotları veri ambarı üzerinde daha iyi sonuçlar elde edilebilmektedir. Veri ambarı çabuk erişim, özel veri işlemleri, özet çıkarım işlemleri, daha hızlı erişim işlemleri için kullanılırken ilişkisel veri tabanları günlük rutin işlemlerde crud (create,read,update,delete) işlemlerinde kullanılmaktadır.



Şekil 5. Veri tabanlarında bilgi keşfi aşamaları [18]

Şekil5'te veri tabanlarındaki verinin bilgiye dönüşmesi aşaması gösterilmiştir. Veri bilgiye dönüşürken genel olarak 4 aşamadan geçtiği söylenebilmektedir. İlk olarak veri önişlem ile üzerindeki gürültüler temizlenir ve veri ambarlarına aktarılır.

Veri ambarındaki bilgiler veri madenciliği tekniklerinin uygulanabilmesi için seçme ve dönüştürme işlemi uygulanır. Bu seçme işlemi veri madenciliği metodunun türüne göre değişebilir. Veri madenciliği teknikleri ile elde edilen veriler insanlar için anlamlı verilere dönüştürülür. Veri madenciliği ile insanlar için anlamlı olan veriler daha sonra örüntü değerlendirme adımında tekrar elenerek insanların işine yaracak veriler seçilir. Son aşamada elde edilen veriler insanlara sunulur ve bilgi elde edilir.

3. KÜMELEME ANALİZİ

3.1. Kümeleme Algoritmaları Tanımı

Veri madenciliğinde kümeleme yaygın şekilde kullanılan verileri sınıflamaktadır. Aynı zamanda kümeler içindeki verileri gruplayan, bu sayede aynı küme içindeki verilerin diğer kümedekilere göre daha benzer kılan tekniktir [18]. Kümeleme analizi mevcut verilerin parçalara ayrılıp gruplanması işlemine denilebilir.

Kümeleme analizi, nesnelerin altdizinlere gruplanmasını yapan işleme denir. Böylece nesneler, örneklenen kitle özelliklerini iyi yansıtan etkili bir temsil gücüne sahip olmuş olur. Kümeleme, bir denetimsiz öğrenme yöntemidir [20].

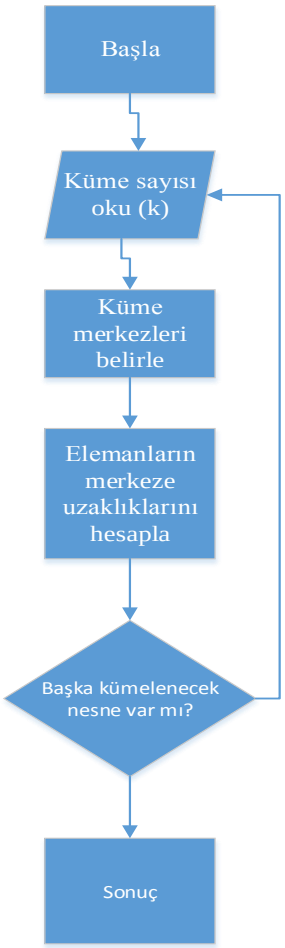
3.2. K-means Algoritması

Verilen nesneleri niteliklerine göre K adet kümeye ayıran en meşhur kümeleme algoritmalarından biridir. Kümeleme işlemi verilerin benzer ve en yakın oldukları küme merkezi civarında toplanmasıyla olur. K-means algoritmasında K değeri probleme göre belirlenebilir veya belirlenmez. Hata kareler ölçütü gibi bir kümeleme ölçütünün olması gerekir. K-ortalamalar algoritması K kümelerini, her bir kümeyi temsil edecek bir nesnenin keyfi seçimiyle başlatır [21]. Kalan her nesne bir kümeye atanır ve kümeleme kriteri küme ortalamasını hesaplayabilmek için kullanılır. Bu ortalamalar yeni küme noktaları olarak ele alınır ve her bir nesne kendisine en benzer olan kümeye yeniden atanır. Bu kümeler yeniden hesaplanır ve kümelerde hiç bir değişim gözlenilmediği duruma ve değişim istenen hata düzeyinin altına düşürülünceye kadar bu döngü devam ettirilir [22].

K-means algoritması uml diyagramı şekil 7 verilmiştir. Bu diyagrama göre bir sonraki bölümde java programlama dili kullanılarak kodlama yapılmıştır. Buradaki akışa göre ilk olarak k küme sayısı belirlenmesi gerekmektedir. Bu K değeri dışarıdan alınabilir veya K sayısı hesaplanması için ayrı bir algoritma da kullanılabilir. K belirlendikten sonra K sayısı kadar rastgele küme merkezi belirlenir. Bu belirlenen merkezlere elemanların uzaklıkları hesaplandıktan sonra elemanlar yakın oldukları merkezin kümesine dahil edilir. Kümelerin ortalamaları

belirlendikten sonra yeni küme merkezleri hesaplanır. Bu işlem bütün elemanlar için yapılana kadar devam eder.

K-means algoritması 7 adımdan oluşmaktadır. Bu adımlar; başla komutu ile açılırken; küme sayısı oku, küme merkezlerini belirle, elemanların merkeze uzaklığını hesapla, elemanları yakın oldukları küme merkezlerine göre kümele, başka kümelenecek nesne var mı kontrol et komutları ile devam eder ve sonuç komutu ile kapanmaktadır.



Şekil 6. K-means algoritmasının adımları [23]

3.2.1 K-means Algoritması Java Kodlama

K-means algoritmasının java ile kodlanması aşağıdaki gibidir.

```
/**

 *
```

```
 * @author furkan.kayim
 */
package tr.com.kayimsoft.yukseklisanstez;
import moa.cluster.Cluster;
import moa.cluster.Kümeleme;
import moa.cluster.SphereCluster;
import java.util.ArrayList;
import java.util.List;
import moa.cluster.CFCluster;
public class KMeansApp {
    public static Kümeleme kMeans(Cluster[] merkez, List<? extends Cluster>
veri) {
        int k = merkez.length;
        int boyut = merkez[0].getMerkez().length;
        ArrayList<ArrayList<Cluster>> Kümeleme =
              new ArrayList<ArrayList<Cluster>>();
        for (int i = 0; i < k; i++) {
           Kümeleme.add(new ArrayList<Cluster>());
        }
        int tekrar = 100;
        while (tekrar-- >= 0) {
```

```
// noktalar clustera atanıyor
for (Cluster nokta : veri) {
    double minmesafe = mesafe(nokta.getMerkez(), merkez[0].getMerkez());
    int closestCluster = 0;
    for (int i = 1; i < k; i++) {
        double mesafe = mesafe(nokta.getMerkez(), merkez[i].getMerkez());
        if (mesafe < minmesafe) {
            closestCluster = i;
            minmesafe = mesafe;
        }
    }
    Kümeleme.get(closestCluster).add(nokta);
}
// yeni merkez hesaplama ve cluster listesini temizleme
SphereCluster[] yeniMerkez = new SphereCluster[merkez.length];
for (int i = 0; i < k; i++) {
    yeniMerkez[i] = merkezHesap(Kümeleme.get(i), boyut);
    Kümeleme.get(i).clear();
}
```

```
            merkez = yeniMerkez;
        }
        return new Kümeleme(merkez);
    }
    private static double mesafe(double[] noktaA, double[] noktaB) {
        double mesafe = 0.0;
        for (int i = 0; i < noktaA.length; i++) {
            double d = noktaA[i] - noktaB[i];
            mesafe += d * d;
        }
        return Matematik.sqrt(mesafe);
    }
    private static SphereCluster merkezHesap(ArrayList<Cluster> cluster, int
boyut) {
        double[] res = new double[boyut];
        for (int i = 0; i < res.length; i++) {
            res[i] = 0.0;
        }
        if (cluster.size() == 0) {
            return new SphereCluster(res, 0.0);
        }
```

```
    for (Cluster nokta : cluster) {
        double[] center = nokta.getMerkez();
        for (int i = 0; i < res.length; i++) {
            res[i] += center[i];
        }
    }
    // Normalizasyon
    for (int i = 0; i < res.length; i++) {
        res[i] /= cluster.size();
    }
    // Yarıçap hesabı
    double yariCap = 0.0;
    for (Cluster nokta : cluster) {
        double dist = mesafe(res, nokta.getMerkez());
        if (dist > yariCap) {
            yariCap = dist;
        }
    }
    return new SphereCluster(res, yariCap);
}
```

```
public static Kümeleme gaussianMeans(Kümeleme gtKümeleme,
Kümeleme Kümeleme) {
    ArrayList<CFCluster> microclusters = new ArrayList<CFCluster>();
    for (int i = 0; i < Kümeleme.size(); i++) {
      if (Kümeleme.get(i) instanceof CFCluster) {
        microclusters.add((CFCluster) Kümeleme.get(i));
      } else {
        System.out.println("Unsupported Cluster Type:" +
Kümeleme.get(i).getClass()
            + ". Cluster needs to extend moa.cluster.CFCluster");
      }
    }
    Cluster[] merkez = new Cluster[gtKümeleme.size()];
    for (int i = 0; i < merkez.length; i++) {
      merkez[i] = gtKümeleme.get(i);
    }
    int k = merkez.length;
    if (microclusters.size() < k) {
      return new Kümeleme(new Cluster[0]);
    }
    Kümeleme kMeansResult = kMeans(merkez, microclusters);
```

```
k = kMeansResult.size();
CFCluster[] res = new CFCluster[k];

for (CFCluster microcluster : microclusters) {
   // en yakın kmeans cluster ı bul
   double minmesafe = Double.MAX_DEĞER;
   int closestCluster = 0;
   for (int i = 0; i < k; i++) {
      double mesafe = mesafe(kMeansResult.get(i).getMerkez(),
microcluster.getMerkez());
      if (mesafe < minmesafe) {
         closestCluster = i;
         minmesafe = mesafe;
      }
   }
   // clustera ekle
   if (res[closestCluster] == null) {
      res[closestCluster] = (CFCluster) microcluster.copy();
   } else {
      res[closestCluster].add(microcluster);
   }
```

```
        }
        // res temizle
        int sayi = 0;
        for (int i = 0; i < res.length; i++) {
            if (res[i] != null) {
                ++sayi;
            }
        }
        CFCluster[] temiz = new CFCluster[sayi];
        sayi = 0;
        for (int i = 0; i < res.length; i++) {
            if (res[i] != null) {
                temiz[sayi++] = res[i];
            }
        }
        return new Kümeleme(temiz);
    }
}
```

3.3. DBSCAN Algoritması

DBSCAN algoritması, Ester, Kriegel, Sander ve Xu tarafından KDD'96 konferansında sunulmuştur [24]. Bu algoritma, nesnelerin komşuları ile olan mesafelerini hesaplayarak belirli bir bölgede önceden belirlenmiş eşik değerden daha

fazla nesne bulunan alanları gruplandırarak kümeleme işlemini gerçekleştirmektedir [25]. İngilizce olarak “Density Temelli Spatial Kümeleme of Applications ile Noise” kelimelerinin baş harflerinin kısaltılmasından oluşan algoritma Türkçeye ’Uygulamaların Gürültü ile Yoğunluğu Tabanlı Mekansal Kümelenme’ şeklinde çevrilebilir. Diğer kümeleme algoritmalarından farklı olarak “Nokta Yoğunluğu (point density)” kavramı ile veriler ilişkilendirilmektedir.

Aşağıda psido kodu verilen bir DBSCAN algoritması için program parçacığı bulunmaktadır. DBSCAN algoritmasının çalışma mantığı bu program ile anlatılmıştır. Program 4 farklı parametre almaktadır.

Public DBSCANalgoritması(DataPoint[] points, Distance distance, double epsilon, int minPoints);

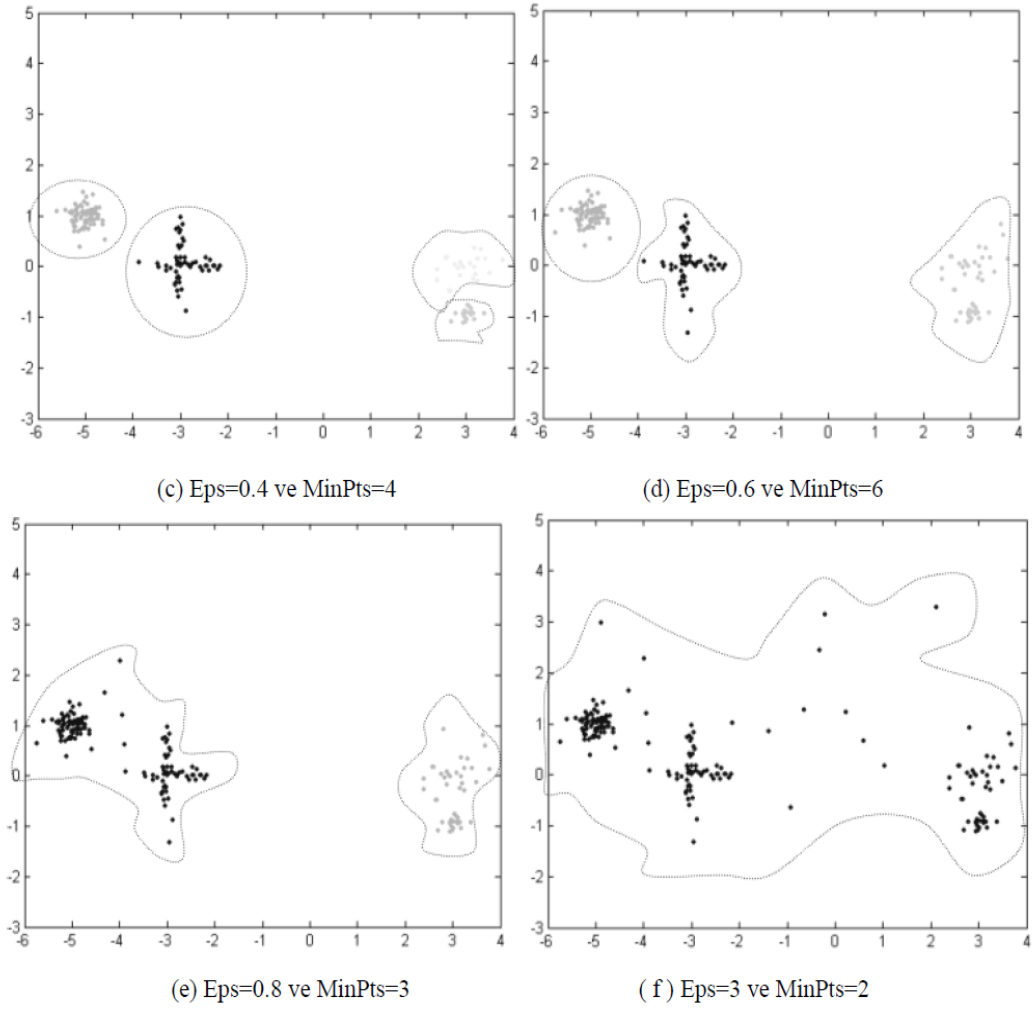
değişkenlerin ne yaptığını açıklamak gerekirse:

Points: Kümeleme yapmak istediğimiz veri setini ifade eder.

Distance: Ne kadar uzaklık için kümeleme yapılacağı bu parametre ile belirlenir.

Epsilon: Genellikle çok küçük pozitif bir sayıyı ifade eden parametredir. Bu değer datapoint p’nin epsilon komşuluk değerinin belirlenmesi için kullanılır. P noktası epsilon değerinden küçük ya da değerine eşit uzaklıktaki noktalar ile komşudur. Bu yüzden “epsilon komşuluğu” şeklinde ifade edilebilir.

Minpoints: Eğer bir nokta bir clustera ait fakat çekirdek noktalardan biri değilse bu durumda sınır(border) noktalardan biri olmaktadır.



Şekil 7. DBSCAN algoritmasında farklı "K" değerleri için elde edilen kümeler **[25]**

Şekil 8'de epsilon ve minpoints değişkenlerine bağlı olarak kümelemenin nasıl değiştiğini incelemek mümkündür [25].

Algoritma çalışma mantığını adım adım incelemek gerekirse;

1- Rastgele daha önce işlem yapılmamış bir nokta seçilir.
2- Seçilen bu rastgele noktanın epsilon uzaklığı içerisindeki komşuları bulunur.
3- Eğer komşu sayısı minpoints sayısına eşit ya da büyük ise bir küme oluşturulur; minpoint sayısından küçük ise bu nokta gürültü olarak işaretlenir. Fakat ileriki safhalarda gürültü olarak işaretlenen bu nokta başka bir noktanın oluşturabileceği kümeye dahil edilebilir.

4- Bir nokta bir kümeye dahil edildi ise bu durumda tüm epsilon komşuları da bu kümeye dahil edilir. Tabii ki bu eklenenler için de geçerlidir. Bu işlem eklenecek nokta kalmayana kadar devam ettirilir ve böylece kümenin tamamı oluşturulmuş olur.
5- Yeni üzerinden geçilmemiş rastgele bir nokta seçilir ve döngü tekrarlanır.

3.3.1 DBSCAN Algoritması Avantajı

DBSCAN algoritmasında kümelerinin sayısı için bir şey belirtmek gerektirmemektedir. Küme sayısını minpoints ve eps ye göre kendisi belirler. Bu algoritma minpoints v4 eps olmak üzere sadece iki parametre gerektirir. Algoritma gelişigüzel şekilli kümeleri bulabilirken veritabanlarında kullanılmak üzere tasarlanmıştır ve bölgesel sorguları çözümlerin hızlandırır.

3.3.1 DBSCAN Algoritması Dezavantajı

DBSCAN algoritması uygulandıktan sonra yoğunlukları büyük farklı veri setleri ile iyi küme olamamaktadır. Kümelemede sınır noktaları bir kümeden diğer kümeye erişilebilir. Bununla birlikte DBSCAN kalitesi distance değişkenine bağlı olması bir dezavantajdır.

4. HADOOP ve BÜYÜK VERİ

Teknoloji geliştikçe hayatımızdaki verilerin büyüklüğünde artmaktadır. Şekil 9'da metrikler verilmiştir. Bu metriklerden bazıları kullanılsa da bazılarını daha önce hiç duyulmamıştır. Teknolojinin bu kadar hızlı geliştiği günümüzde duyulmayan metrikleri bile kullanıyor olmamız kaçınılmazdır.

Byte (B)	1	1 byte
Kilobyte (kB)10^3	1,000	1 thousand bytes
Megabyte (MB)10^6	1,000,000	1 million bytes
Gigabyte (GB)10^9	1,000,000,000	1 billion bytes
Terabyte (TB)10^{12}	1,000,000,000,000	1 trillion bytes
Petabyte (PB)10^{15}	1,000,000,000,000,000	1 quadrillion bytes
Exabyte (EB)10^{18}	1,000,000,000,000,000,000	1 quintillion bytes
Zettabyte (ZB)10^{21}	1,000,000,000,000,000,000,000	1 sextillion bytes
Yottabyte (YB)10^{24}	1,000,000,000,000,000,000,000,000	1 septillion bytes
Brontobyte 10^{27}	1,000,000,000,000,000,000,000,000,000	1 octillion bytes
Geopbyte 10^{30}	1,000,000,000,000,000,000,000,000,000,000	1 decillion bytes

Million	10^6
Billion	10^9
Trillion	10^{12}
Quadrillion	10^{15}
Quintillion	10^{18}
Sextillion	10^{21}
Septillion	10^{24}
Octillion	10^{27}
Nonillion	10^{30}
Decillion	10^{33}
Undecillion	10^{36}

Şekil 8. Metrikler

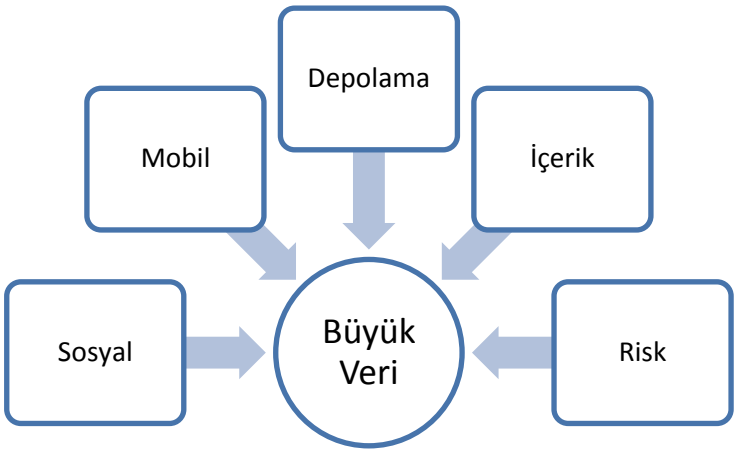
4.1 Büyük Veri Nedir?

Dünyada her gün 2.5 quintillion byte veri oluşmakta ve günlük olarak bu rakam sürekli artmaktadır. Son iki yılda yaratılmış olan tüm verilerin %90'ı son iki yılda yaratılmıştır. Veriler, çeşitli sensor kaynaklarından, sosyal medya gönderilerinden, dijital fotoğraf ve videolar, cep telefonu gps sistemleri, sağlık verileri gibi kaynaklardan oluşmaktadır .Dünyada bir dakikada neler olduğuna kısaca bakıldığında FireFox'tan 1,700 download gerçekleştirilmekte, Youtube'a 600 yeni video eklenmekte, Linkedin'e 100 yeni üye katılmakta, Twitter'a 320 yeni üye katılmaktadır. Bu durum bir dakika içerisinde oluşan verilerin dahi çok büyük boyutlu olabileceğini göstermektedir [26].

Büyük veri kavramı aslında yeni bir kavram olmayıp geçmiş dönemlerde de farklı şekillerde karşımıza çıkabilmektedir. Son yıllarda internet kullanımın yaygınlaşması, sanal ortamda sosyalleşmenin patlama yapması, akıllı cihazların yaygınlaşması, teknolojinin ulaşılabilirliğinin artması ve ucuzlaması veri miktarındaki artışını tetiklemiştir. Eski yaklaşıma göre iş sahibinin ihtiyaçları belirlenir ve bu ihtiyaçlara göre geliştirmelere yapılırken günümüzde bilgi

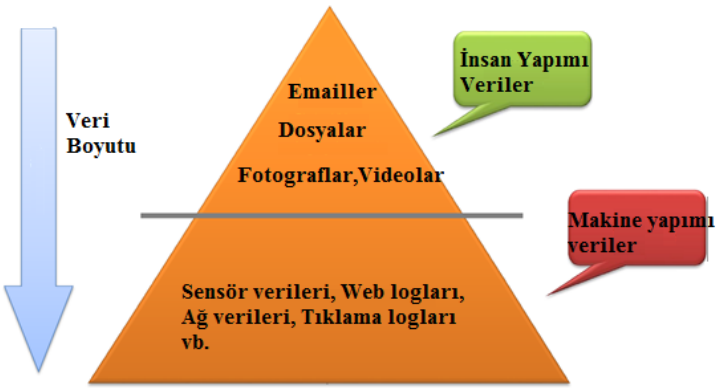
teknolojilerinin sunduğu platform ile iş sahibinin özgürce veriyi keşfetmesini sağlayacak altyapılar sunmak zorundadır [27].

Geleneksel veri tabanları güncelliğini korumakta, bununla birlikte artan veri miktarı katlanarak artmaya devam ettiği için veriyi bulunduğu yerde işleyecek IBM-DB2 Analytical Machine, Oracle-Exadata, IBM-Netezza gibi yazılımlara ihtiyaç artmaktadır. JAQL, HBASE, NoSQL, Hadoop, Cassandra gibi yazılım ve donanımı içinde bulunduran çözümler ile büyük veri probleminin çözümü aranmaktadır.



Şekil 9. Büyük veri ile bağlantılı terimler **[27]**

Şeki10'da büyük veri denildiğinde ilk akla gelen terimler vurgulanmıştır ve bunlara ekleme yapmak mümkündür. Aslında büyük veri donanıma göre de değişiklik göstermekte ve normal bir kişisel bilgisayar için büyük olan veriler sunucular için küçük kalabilmektedir.



Şekil 10. Verilerin dağılımı

Şekil 11'de oluşan verilerin dağılımı anlatılmıştır. Küçük verileri e-mail, dosyalar, fotoğraflar, videolar oluştururken büyük verileri web logları, sensör dataları, ağ dataları oluşturmaktadır.

Şekil 11. Büyük veri

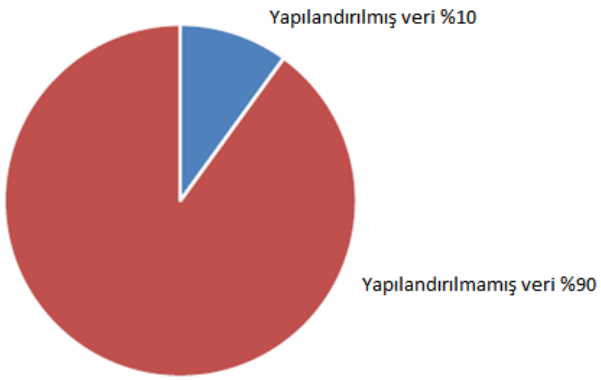
Büyük Verinin Dönüştürülmesi

- **Hız:** Periyodik ve gerçek zamanlı olarak veri miktarı hızlı bir şekilde artmaktadır.
- **Çeşitlilik:** Üretilen verinin %80′i yapısal değil ve her yeni üretilen teknoloji, farklı formatlarda veri üretebiliyor [27]. Veriler çok farklı kaynaklardan çok farklı çeşitlilikte üretilmektedir.
- **Doğrulama:** Bu kadar çok verinin doğru kişilere doğru şekilde ulaşıyor olması gerekmektedir.
- **Değer:** Çeşitliliği ve üretim hızı çok fazla olan veri doğru kişilere doğru kısımları ulaşırsa değerli olacaktır.

Büyük veriden değer elde edilirken bazı adımlardan geçmesi gerekmektedir. Doğru verinin doğru kısımlarının güvenli bir şekilde son kullanıcıya ulaşmış haline de değer denir.

4.2 Veri Tipleri

Veri tipleri yapılandırılmış ve yapılandırılmamış veriler olmak üzere ikiye ayrılmaktadır. Şekil 13'de dünyadaki verilerin dağılımı gösterilmiştir. Görüldüğü üzere verilerin çoğunluğunu yapılandırılmamış veriler oluşturmaktadır.



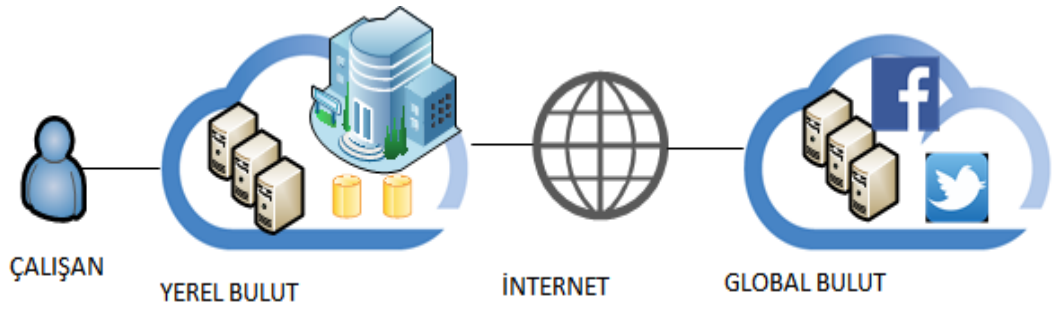
Şekil 12. Veri Tipleri

4.3 Büyük Verinin Zorlukları

Büyük veriler altyapı, uygulama, tecrübe ve değer odaklı zorlukları bulunmaktadır. Bu zorluklar:

- Altyapı: Büyük verinin saklanıp muhafaza edilebilmesi güvenliğinin sağlanabilmesi için güçlü ve büyük donanım ihtiyacı ortaya çıkmıştır.
- Uygulamalar: Uygulamalarda büyük verilerle yapılan işlemlere ihtiyaç varsa bu işlemlerde performans problemleri ortaya çıkabilmektedir.
- Tecrübe: Günümüz teknolojisi ve bu teknolojilerle uğrasan personeller büyük verilerle uğraşacak yeterli tecrübeye sahip değildir.
- Değer Kazandırma: Büyük verilerle işlem yapılan uygulamalarda sonuç odaklı yaklaşım ve uygulamalara değer kazandırma problemleri ortaya çıkmaktadır.

4.4 Bulut Bilişim Nedir?



Şekil 13. Bulut Bilişim[28]

Bulut bilişim (cloud hesaplama) veya işlevsel anlamıyla çevrimiçi bilgi dağıtımı; bilişim aygıtları arasında ortak bilgi paylaşımını sağlayan hizmetlere verilen genel addır. Bulut bilişim bu yönüyle bir ürün değil, hizmettir. Temel kaynaktaki yazılım ve bilgilerin paylaşımı sağlanarak, mevcut bilişim hizmetinin bilgisayarlar ve diğer aygıtlardan elektrik dağıtıcılarına benzer bir biçimde bilişim ağı (tipik olarak İnternet'ten) üzerinden kullanılmasıdır [28]. Bulut bilişim teknolojileri aşağıdaki gibidir:

–Amazon EC2: virtual machines at 8¢/hour, billed hourly

–Amazon S3: storage at 12.5¢/GB/month

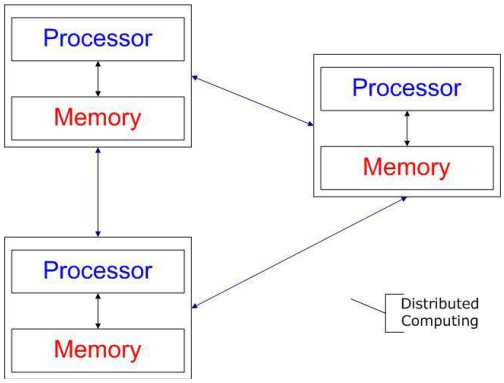
–Windows Azure: applications using Azure API

4.5 Bulut Bilişim Yatay Büyüme

Bulut bilişim yatay ve dikey anlamda çok hızlı büyüyen ve esnek yapısı nedeniyle her türlü gelişmeye hızlı cevap veren bir sistemdir. Normal bilgisayarlar dikey olarak büyüme gösterirken süper bilgisayarlar daha çok yatay büyüme göstermektedir. Bulut bilişim sayesinde büyük pahalı sunucular yerine küçük makinelerle daha verimli donanımlar ortaya çıkarılmıştır.

4.6 Dağıtık Hesaplama (Distributing Computing)

Dağıtık hesaplama bir hesaplama konseptidir ve birden fazla bilgisayarın tek bir bilgisayar gibi bir problemi çözmesine denir. Bir problem birden fazla parçalara ayrılır ve her bir parça farklı bilgisayarlar tarafından çözülür.



Şekil 14. Dağıtık Hesaplama

4.7 Hadoop Nedir?

Hadoop büyük veri dosyalarının paralel şekilde işlenmesi için kullanılan MapReduce yönteminin açık kaynak kodlu gerçekleştirimidir [29]. Bilgisayar kümeleri üzerinde çalışan dağıtık programlar için alt yapı desteği sunmaktadır [3]. Hadoop'un sunduğu diğer önemli bölümden biri de Hadoop Dağıtık Dosya Sistemi'dir (HDFS). Bu özellikten ilerleyen bölümlerde bahsedilmiştir.

Hadoop büyük veri setlerini datığıtık hesaplama ile işlemeye yarayan bir framework'tür. Bir veya birçok bilgisayarı organize ederek tek bir bilgisayara kapasitesi gibi kullanmaya yaramaktadır. Apache firmasının en çok desteklediği projelerden biri olan bu teknoloji java programlama dili kullanılarak geliştirilmiştir.

Çalışma mantığında göre petabaytlarca veri binlerce noda ayrılıp üzerinde çalışabilir. En büyük avantajlarından biri de nodlarını oluşturacak bilgisayarların pahalı bilgisayarlardan oluşmamasıdır. HDFS ve MapReduce iki temel kısımdan oluşur. Bir diğer avantajı ise dağıtık hesaplamanın ortadan kaldırmasıdır. Hadoop teknolojisiyle geliştirilen projelere yardımcı olmak için hataları otomatik olarak yakalayabilecek şekilde gelişmiştir.

4.8 Hadoop Büyük Veri Problemini Nasıl Çözer?

Hadoop yatay olarak büyüyerek pahalı ve güçlü bilgisayarları zorunlu tutmadan büyük veriye alt yapı sağlamaktadır. Yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış veriler Hadoop ile tutulabilmektedir. Hadoop'un bir diğer özelliği de veriyi saklamaya ve dağıtık hesaplamaya aynı anada izin veriyor olmasıdır.

4.9 Hadoop Kullanıcıları

Dünyada birçok firma Hadoop'tan yararlanmaktadır. Bu firmalardan önce gelen bazıları ve kullanım amaçları ve özellikleri aşağıdaki gibidir.

EBay

Araştırmalar ve arama optimizasyonları için kullanılmaktadır.

532 nodes cluster (8 * 532 çekirdek, 5.3PB).

Facebook:

Internal logların kopyalarının saklanmasında, raporlama, analiz işlemleri ve makine öğrenmesinde kullanılmaktadır.

2 temel cluster bulunmaktadır.

A 1100-makina cluster 8800 çekirdek 12 PB depolama hacmi.

A 300-machine cluster 2400 çekirdek 3 PB depolama hacmi.

Her bir düğüm 8 çekirdek ve 12TB depolama hacmine sahip.

IBM

Bulut Hesaplama Clusterları

Üniversite Girişimi İnternet Ölçekli Bilgi İşlem Zorluklar Adres

Yahoo!

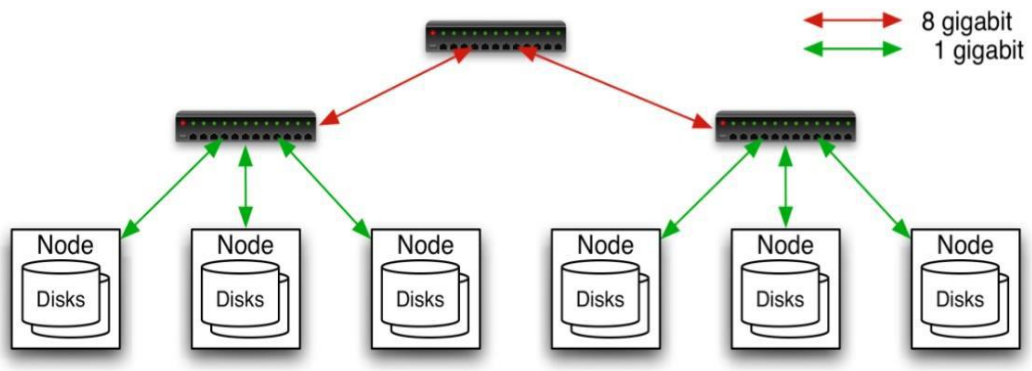
100,000 den fazla CPUs >40,000 bilgisayar var Hadoop içinde

En büyük cluster: 4500 nodes (2*4cpu boxes w 4*1TB disk & 16GB RAM)

Linkedin ,alibaba ve daha bir sürü büyük firma Hadoop kullanmaktadır.

Daha detaylı bilgi için [30].

4.10 Temel Hadoop Cluster Yapısı



Şekil 15. Temel Hadoop cluster yapısı

Şekil 16'da temel bir Hadoop cluster yapısı gösterilmiştir. Bir cluster yapısı için facebook firmasındaki cluster yapısının özellikleri aşağıdaki gibidir.

- 40 nodes/rack, 1000-4000 nodes in cluster
- 1 Gbps bandwidth in rack, 8 Gbps out of rack
- Node specs (Facebook):8-16

4.12 Hadoop ve Rdbms Kıyaslaması

	Geleneksel RDMS	Hadoop / MapReduce
Veri Boyutu	Gigabat - Terabayt	Petabayt - Heksabayt
Erişim	İnteraktif ve batch	Batch , interaktif değil
Güncelleme	Defalarca okuma ve yazma	Bir kere yazma, defalarca okuma
Yapı	Statik şema	Dinamik şema
Bütünlük	Yüksek	Düşük
Ölçekleme	Lineer değil	Lineer
Sorgu Cevap Süresi	Hemen	Gecikme olabilir

Şekil 16. Hadoop Rdms kıyaslaması

Şekil 17'de RDMS sunucu ile MapReduce mimaride geliştirilmiş sunucuların özellikleri belirtilmiştir. Şekilden de anlaşılacağı üzere iki sunucu farklı alanlarda kullanılabilir. RDMS sunucu klasik işlemlerde kullanılabilir. Hadoop sunucu işe büyük verilerde daha başarılı sonuçlar elde edilecektir.

Kargo treni: (Hadoop)

Hadoop bir kargo treni gibi düşünülmelidir. Bu bağlamda;

- Büyük verilerle çalışmak gerekiyorsa kullanılabilir.
- Çok pahalı bilgisayarlara ihtiyaç yoktur.
- Küçük sorgulara beklendiği kadar hızlı yanıt veremeyebilir. Hızlanması kolay değildir.
- Üzerinde her türlü veri saklanabilir. (Yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış veriler.)
- Çok büyük veriler saklanabilir, işlenebilir.
- Aynı anda hem saklamak hem de işlemek mümkündür.

Spor Araba(RDBMS)

RDBMS sunucuları spor, lüks bir araba gibi düşünülmelidir. İşlevi gereği büyük olmayan hızlı işlem yapılması gereken orta seviye projelerde kullanılması mümkündür. Bununla birlikte;

- Dakik olarak işlemler gerçekleştirilir.
- Lüks arabalarda olduğu gibi normal bilgisayarlarda olmayan özelliklere sahiptir.
- Çok hızlı işlem yapabilir.
- Çok pahalı cihazlardır.
- Koruması da kolay değildir.
- Uygun koşullarda saklanması gerekmektedir.

Sonuç olarak RDBMS ve Hadoop/MapReduce sunucularda farklı projelerde kullanılabilir. Hadoop sunucular büyük ve orta ölçekli projelerde tercih edilebilir. RDBMS sunucular ise küçük ve orta ölçekli projelerde tercih edilebilir.

4.13 HADOOP Ekosistemi

Hadoop ekosistemi içerisinde birçok hizmet sunulmaktadır. Bu hizmet temel olarak aşağıdaki başlıklardan oluşmaktadır.

Hadoop veri tabanları: Hbase, Hypertable

Operasyonel Servisler: Apache Ambarı, Apache Oozie, Apache Mahout

Veri Servisleri: Hive, Pig Jaql, Impala

Diğer Hizmetler: HDFS, HDFS Araçları, ETL Araçları, Flume, Sqoop, Hue

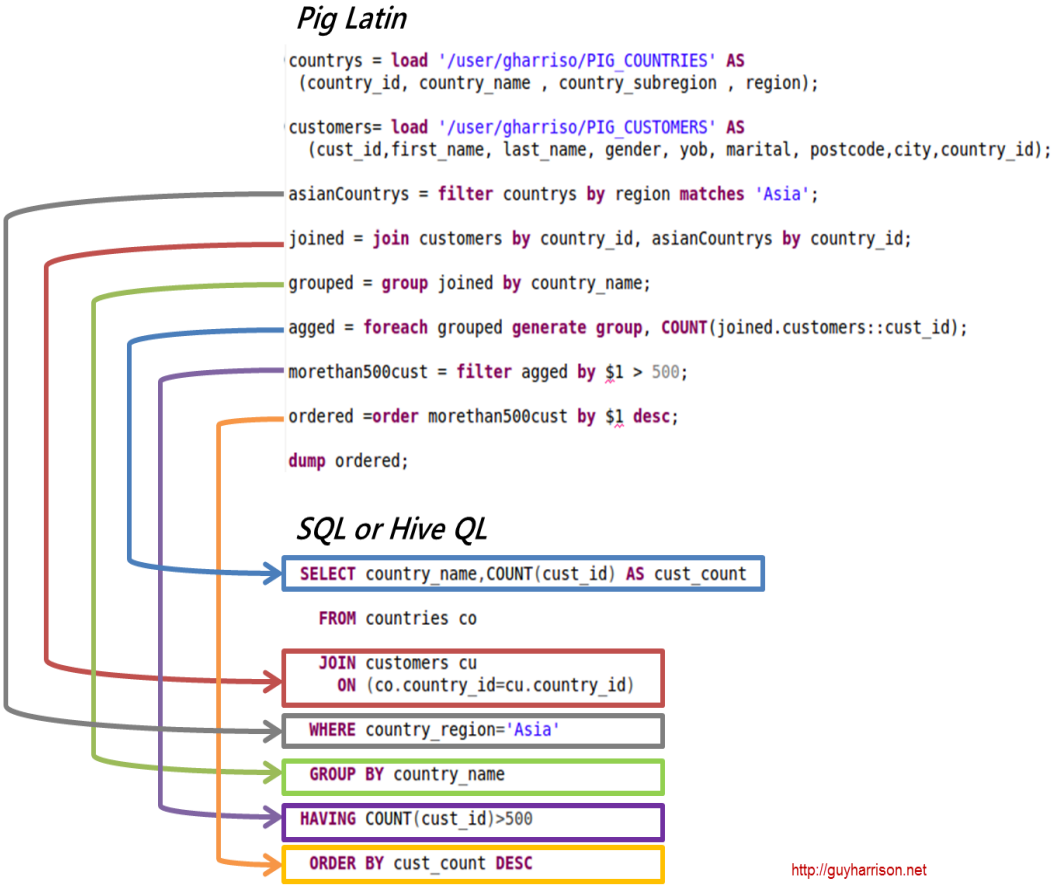
4.13.1 Pig

Pig, Yahoo araştırması sonucu başlamış bir Hadoop teknolojisidir. SQL operasyonlarına imkân sağlamaktadır. En büyük artılarından biri java programlama dili bilmeyenlere veriyi işleme ve çıktı alma imkânı sunmasıdır. Apache Pig, Apache Hadoop üzerinde prosedürel bir veri akışı yazmayı sağlayan bir veri işleme platformudur. Veri kullanıcılarına Hadoop'un güçlü, dağıtık ve esnek yapısına ileri seviye java kodları yazmadan daha üst bir katmandan erişim imkanı sağlayan bir yapıdır [31]. Pig, Hadoop üzerindeki büyük verileri geliştirmek için geliştirilmiş olup Pig Latin olarak isimlendirilen yüksek seviyeli bir dile sahip veri işleme aracıdır.

4.13.1.1 Pig Özellikleri

MapReduce yazım zorluğunu ortadan kaldırarak java programa dili bilmeyenler için de işlem yapma özelliği katmıştır. Basit bir dili olmakla birlikte yeniden kullanılabilirlik de sunmaktadır. Standart java dosya formatlarından olan Json, Metin, Binary vd. ile işlem yapmaya izin vermektedir.

4.13.1.2 Pig Kalıp Kod



Şekil 17. Pig kalıp Kod[58]

4.13.1.3 Pig Avantajları

- Pig kodları MapReduce mimarisinde çalıştığı için MapReduce mimarisinin tüm avantajları Pig için de geçerlidir.
- SQL de kullanılan fonksiyoniliteler Pig ile de gerçeklenebilmektedir. Avantajlı yanı Pig daha anlaşılır olmasıdır.
- Performans ve ölçeklenebilirlik açısından büyük farklılıklar ortaya çıkmaktadır.
- Yapılandırılmamış veriler için uygun bir kullanıma sahiptir.
- Bir diğer kullanım alanı ETL işleridir.

4.13.2 Hive

Sql kodları ile Hadoop mimarisi üzerinde sorgulama işlemleri yapmayı sağlayan bir teknolojidir. Ayrıca veri ambarı uygulamaları geliştirmek için de kullanılabilmektedir.

Apache Hive dağıtılmış depolama yazılımı ve Hadoop SQL ve depolama yetenekleri eklemek için 2008 yılında Facebook tarafından geliştirilmiştir [32].

4.14.1 Hive Özellikleri

- Günümüz bilgisayar teknolojileri ile uğraşanlar içerisinde sql'in bilinme oranı çok yüksektir. Hive MapReduce mimarisinde sql komutlarını çalıştırmayı sağladığından önemlidir.
- Standart sql-92 komutlarına yakın bir kullanımı vardır.
- Veri saklama HDFS üzerindedir.
- İnsert ve select daha çok desteklenmektedir update ve delete göre.
- Dosya tanımlama işlemi tablolar üzerinde olmaktadır.
- Standart java dosya formatlarından olan Json, Metin, Binary vd. ile işlem yapmaya izin vermektedir.
- Standart dosya formatları dışında map, struct ve array gibi veri tiplerini de destekler.
- Hive ile entegrasyon işlemleri kolay bir şekilde gerçekleştirilebilmektedir.
- Performans olarak ve ölçeklenebilirlik bakımından avantajlıdır.
- Büyük sorgularda anlık sorgulara göre daha başarılıdır.
- Kümeler üzerinde çalışmaktadır.

4.14 Hive ile Pig Karşılaştırması

Hive ile Pig karşılaştırıldığında Hive veri ambarını kullanırken Pig ETL kullanmaktadır. Hive'de dil SQL iken Pig, Pig Latin kullanmaktadır. Bununla birlikte Şekil 19'da da görüleceği üzere şema, dosya formatları, JDC/ODBC noktalarında da birbirinden ayrılmaktadır. Aynı zamanda join, genişleyebilme, zengin veri tipi ile NoSQL açısından da benzerdir.

Özellik	Pig	Hive
Amaç	ETL	Veri Ambarı
Dil	Pig Latin	SQL*
Şema	Esnek	Zorunlu
Zengin veri tipleri (map vs)	Var	Var
Join	Var	Var
Genişleyebilme	UDF	UDF
Farklı Dosya Formatları	UDF	SerDe
JDBC/ODBC	Yok	Var*
NoSQL entegrasyonu	Var*	Var*

Şekil 18. Hive ile Pig Karşılaştırması **[33]**

4.15 Impala

Hive ile benzer sorguları çalıştırılabildiği gibi birçok dosya formatını da desteklemektedir. Hive'nin daha hızlı çalışmasının sebebi sorguların MapReduce yapısına çevrilmezliğidir. Bunun yerine dağıtık sorgularla verilere direk erişerek daha hızlı işlem yapmaktadır. Hive ile Impala farklılıklarını saptamak amacı ile ilerleyen bölümlerde performans testi yapılmıştır.

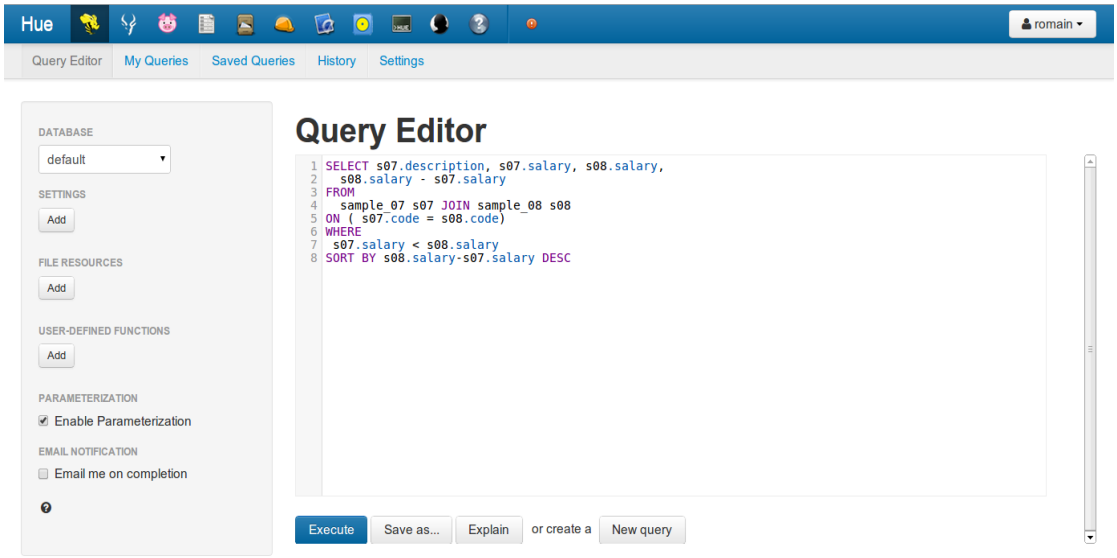
Impala, eski Cloudera Impala'dan geliştirilmiş olup, bugün Apache Impala olarak bilinmektedir ve Apache Hive'in indekslenmiş gecikme yetenekleri eklenmiş halidir [32].

4.16 HUE

Hue, Hadoop teknolojileri için bir editördür. Hue kurulumu için cloudera [34] sitesini incelemek yeterlidir. Bununla birlikte Hue'deki özellikleri şöyle sıralamak münkündür:

- Hive sorguları saklamaya yarar.
- Hive tabloları yaratmaya ve yüklemeye yardımcı olur.
- HDFS dosyaları ve dizinleri ile çalışmayı sağlar.
- MapReduce görevlerini oluşturma ve görüntülemek için kullanılabilir.
- Oozie kontrol panelinde iş akışları oluşturup kontrol etmek için kullanılabilir.

- Kullanıcıları kontrol etmek için kullanılabilir.



Şekil 19. Hue

4.17 Hadoop Distributed File System(HDFS)

Hadoop Dağıtık File System, büyük verileri işlemek için oluşturulmuş normal sunucuların küme oluşturarak dağınık dosya sistemidir ve üç parçadan oluşmaktadır. Bunlardan Namenode: HDFS üzerindeki verilerin yönetimini yapmaktan sorumludur. İkincil olan Secondary Namenode; adından anlaşılacağı üzere, Namenode'un yedeği olarak çalışmaktadır. Üçüncü parçayı oluşturan Datanode ise; gerçek verilerin saklandığı bölümdür. Hadoop sisteminde Namenode ve Secondary Namenode birer adet bulunur ve master makine üzerinde çalışırken, DataNode bir veya daha fazla sayıda olabilir ve slave makineler üzerinde çalışır [3].

HDFS açık kaynak kodlu bir proje olarak geliştirilmiştir. HDFS veri işlemede Google Dosya Sistemi'nde (Google File System (GFS)) [35]kullanılan MapReduce [36] programlama yöntemini temel almıştır. Bu yöntem büyük ölçekli veriler üzerinde hesaplamanın paralelleştirilmesini çok etkin bir hale getirmiştir.

4.18 HBase

HBase, Büyük veri üzerinde rastgele ve gerçek zamanlı hızlı veri okuma ve yazma amacıyla kullanılabilen HDFS üzerinde çalışan ilişkisel olmayan sütun-tabanlı veri tabanıdır [37].

Büyük veriler ve HDFS üzerinde çalışan ilişkisel veri tabanı dışında bir veri tabanına ihtiyaç duyulduğu zaman kullanılabilir. Ölçeklenebilirliği sağlamaktadır. Bu veri tabanı üzerinde tablo boyutları çok büyük olabilmektedir. Bu veri tabanının en önemli farklılığı hızlı erişim sağlaması ile ortaya çıkmaktadır.

4.19 Apache Mahout

Apache Mahout, Hadoop üzerinde çalışan büyük veri kümeleri üzerinde veri madenciliği algoritmalarını çalıştırmak amacıyla kullanılan bir makine öğrenme kütüphanesidir. Üzerinden bulunan hazır algoritmalar ile büyük veri probleminin çözümünde ile veri madenciliği algoritmalarının da kullanılmasına imkân sağlamaktadır.

Kümeleme, sınıflandırma, önerme vb. alanlarda uyarlanmış birçok makine öğrenme metodunu barındırmaktadır [37].

Tablo 1- Mahout 0.10.0 özellikleri [38]

Tek Makine	**MapReduce**	**Spark**	**H2O**	**Flink**
Mahout Matematik-Ölçekli Core Kütüphane and Ölçekli DSL				
Dağıtık ALS, SPCA, SSVD, thin-QR. Benzerlik Analiz.			x	x
Mahout İnteraktif Kabuk				
İnteraktif REPL Kabuk için Spark optimizasyonu Mahout DSL			x	
Kolaboratör Filitreleme				
Kullanıcı-Temelli Kolaboratör Filitreleme	x		x	
Madde-Temelli Kolaboratör Filitreleme	x	x	x	
Matriks Faktorizasyon ile ALS	x	x		
Matriks Faktorizasyon ile ALS on Kesin Geri Dönüş	x	x		

Tek Makine	**MapReduce**	**Spark**	**H2O**	**Flink**
Ağırlıklı Matriks Faktorizasyon, SVD++	x			
Sınıflandırma				
Locistik Gerileme - trained via SGD	x			
Naive Bayes / Complementary Naive Bayes		x	x	
Random İçinest		x		
Hidden Markov Modelleri	x			
Çok katmanlı Algılayıcı	x			
Kümeleme				
Canopy Kümeleme	Onaylandı	Onaylandı		
K-means Kümeleme	x	x		
Fuzzy K-means	x	x		
Streaming K-means	x	x		
Spectral Kümeleme		x		
Tekil Değer Ayrışma	x	x	x	x
Lanczos Algoritma	Onaylandı	Onaylandı		
Stochastic SVD	x	x	x	x
PCA (via Stochastic SVD)	x	x	x	x
QR Ayrışma	x	x	x	x
Topic Modelleri				
Latent Dirichlet Dağıtma	x	x		
Çeşit				
Sıra Benzerlik işi		x	x	

Tek Makine	MapReduce	Spark	H2O	Flink
Concat Metriks		x		
Sıralama		x		
Sparse TF-IDF Vektör den Metin		x		
XML Ayrıştırma		x		
Email Arşiv Ayrıştırma		x		
Lucene Birleşme		x		
Evrimsel İşlemler	x			

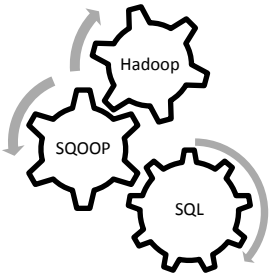
Apache Mahout birçok veri madenciliği algoritması ile Hadoop üzerinde çalıştırılmasına olanak sağlamaktadır. Açık kaynak kodlu olduğu için diğer algoritmalarında geliştirilmesi mümkündür.

4.20 FLUME

Hadoop üzerinde veri alış verişi için flume kullanılır. Flume, farklı kaynaklardan Hadoop sistemine, hızlı, güvenilir, hataya toleranslı ve dağıtık veri aktarmayı sağlar [37].

4.21 Sqoop

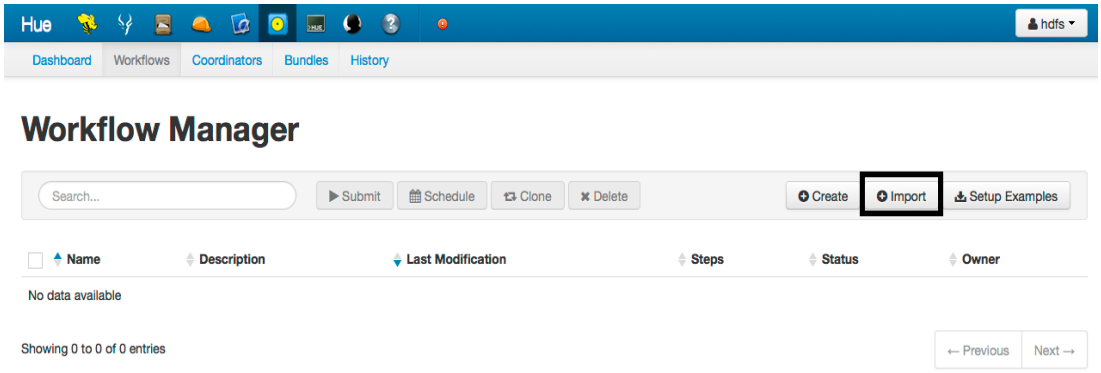
Apache Sqoop (TM) verimli bir şekilde ilişkisel veri tabanları gibi Apache Hadoop ve yapılandırılmış veri depolarına arasındaki toplu veri aktarımı için tasarlanmış bir tool'dur.



Şekil 20. Sqoop **[39]**

Sqooop ilişkisel veri tabanları ile Hadoop arasında veri alışverişinde kullanılır. İlişkisel veri tabanlarındaki veriyi Hadoop sistemine aktaramaya yardımcı olduğu gibi Hadoop üzerindeki veriyi de ilişkisel veri tabanına aktarmaya yardımcı olur.

4.22 Oozie



Şekil 21. Oozie

Oozie, Apache Hadoop işlerini yönetmek için bir iş akışı zamanlayıcı sistemidir. İş akışları tasarlamaya yönetmeye yarayan bir sistemdir.

4.23 Hadoop Teknolojilerinden Nerede Ne Kullanılmalı?

Veri toplamak için kullanılması gereken Hadoop teknolojileri:

– Flume – cloudera tarafından kullanılmaktadır.

– Scribe – facebook tarafından kullanılmaktadır.

Veri saklamak için kullanılması gereken Hadoop teknolojileri:

– HDFS dosya yönetim sistemidir.

– HBASE veri tabanıdır.

Veri Analizi için kullanılması gereken Hadoop teknolojileri:

– MapReduce

– Pig

– Hive

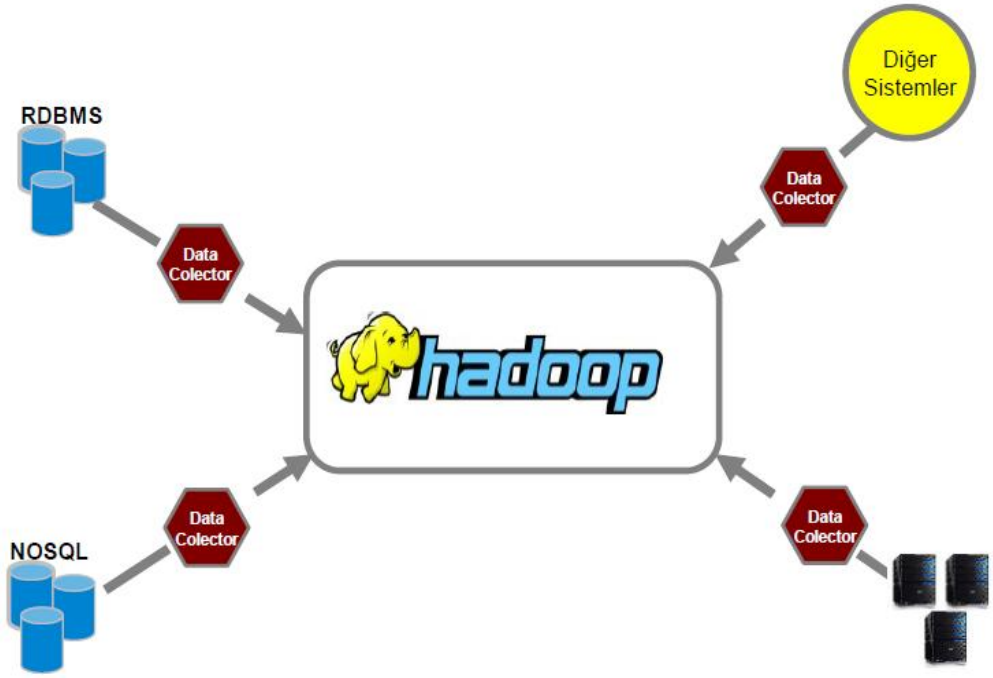
– Impala

Akıllı uygulamalar ve veri madenciliği algoritmaları kullanılması gerekiyorsa:

– Mahout

4.24 Hadoop için örnek bir kullanım senaryosu

Şekil 23.'de Hadoop için örnek bir kullanım senaryosu canlandırılmıştır. Buradan anlaşılacağı üzere Hadoop veri tabaları arasında veri alışverişi sağlamaktadır.



Şekil 22. Hadoop için örnek bir kullanım senaryosu

4.25. Hadoop – MapReduce Mimarisi

Hadoop, büyük veri dosyalarının paralel şekilde işlenmesi için kullanılan MapReduce yönteminin açık kaynak kodlu gerçekleştirimidir [29]. 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.

Hadoop, MapReduce mimarisinde geliştirilmiştir. Hadoop ayrıca bilgisayar kümesi için yüksek bant genişliği sağlayan dağıtık dosya sistemini (HDFS) kullanıcıya sunar [40]. Hem MapReduce hem de HDFS dosya sisteminin kullanımı ile birlikte düğüm hataları otomatik olarak düzeltilir ve kullanıcıya yüksek güvenlikli bir sistem sunulur [3]. MapReduce mimarisi sayesinde olası donanım problemlerinde kullanıcı desteği olmadan kurtarma işlemleri yapılabilir.

MapReduce zaman alan analitik sorguların büyük ölçekli veriler üzerinde toplu olarak işletilmesi için popüler bir yöntemdir [41]. Birçok bilgisayarın bir arada küme olarak çalışabilmesini sağlayan mimariye MapReduce mimarisi denir.

Google tarafından ortaya atılan ve günümüzde sıkça karşımıza çıkan MapReduce modeli [40], bilgisayar kümesi üzerinde dağıtık programlama yapılabilmesine olanak sağlamaktadır.

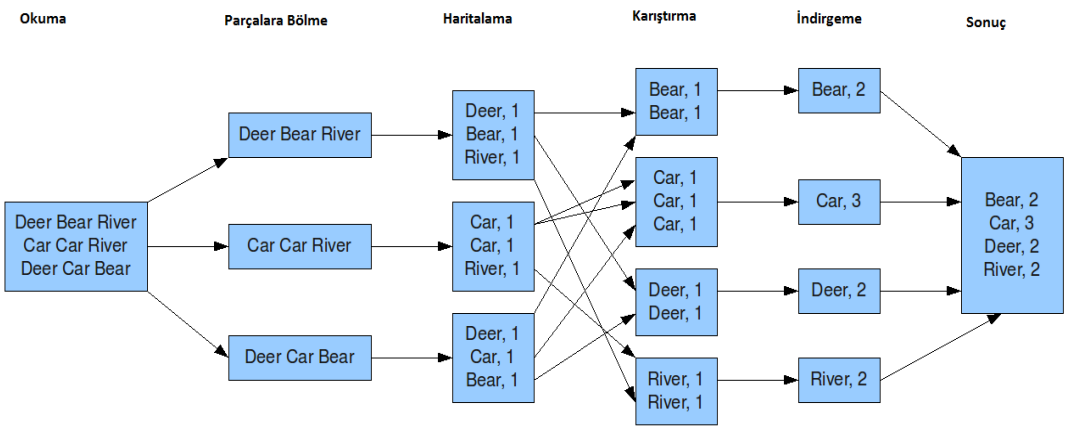
MapReduce mimarisinin çekirdeği map ve reduce fonksiyonlarıdır [42]. Kullanıcı bu fonksiyonları gerçekleyerek dağıtık bir uygulama geliştirmektedir. MapReduce mimarisi bu fonksiyonları dağıtık olarak çalıştırarak veri setlerinin paralel şekilde işlenmesine olanak sağlar. MapReduce mimarisi daha çok veri-yoğun işlemler için kullanılmaktadır [29]. Fakat kodlamasının ve anlaşılmasının kolay olması, otomatik olarak hata tespiti yapması ve gerekli hata ayıklama işlemini gerçekleştirebilmesi nedeniyle işlemci yoğun uygulamalar için de kullanılmaktadır.

MapReduce, map fonksiyonu ve reduce fonksiyonu olmak üzere iki temel fonksiyondan oluşmaktadır.

```
MapReduce(
  giris = '/test/input.native',
  cikis = '/test/output.native',
  map = fonksiyon(k, v){
```

```
  anahtar = v
  nVeri = dim(v)[1]
  deger = matriks(veri=1, nrow= nVeri, ncol=1)
  return(anahtarDeger(key, val)
 },
reduce = fonksiyon(k, v){
  anahtar = k[1, 1]
  deger = topla(k[, 2])
  return(anahtarDeger(anahtar, deger)
 }
)
```

Yukarıda MapReduce fonksiyonu için örnek bir kaba kod bulunmaktadır.



Şekil 23. MapReduce kelime sayısı bulma[59]

Şekil 24'de MapReduce mimarisin çalışma şekli gösterilen örnek bir kelime sayısı bulan program şekillendirilmiştir. MapReduce mimarında bir dosya işlenirken aşağıdaki adımlar sırası ile gerçekleşmektedir:

- Dosya okuma: Şekil 24 içinde 4 çeşit kelime bulunan bir text okunmuştur.
- Parçalama: Dosya 3 parçaya bölünmüştür.
- Haritalama: MapReduce mimarisindeki map adımının gerçeklemesidir.
- Karıştırma: Parçalardan benzer kelimler aynı gruba alınır.
- İndirgeme: MapReduce fonksiyonundaki reduce fonksiyonuna denk gelir.
- Sonuç: Sonuçta 4 çeşit kelimenin kaçar adet bulunduğu MapReduce mimarisi ile tespit edilmiştir.

4.25.1 Map fonksiyonu

```
map = fonksiyon(a, d){
   anahtar = ...
   deger = ...
   return(anahtarDeger(anahtar, deger))
   }
```

Yukarıda pisido kodu verilen map fonksiyonu anahtar ve değer olmak üzere 2 parametre almaktadır.

Map fonksiyonunun amacı girdi olarak gelen verilerin gruplanıp, sıralanıp çıktılarının reduce fonksiyonuna aktarılmasıdır [29]. MapReduce fonksiyonun girdisi olan Hadoop yapısında kullanılacak dosya HDFS dosya sisteminde olmalıdır.

```java
package my.pack;

import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException
    {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
         while (tokenizer.hasMoreTokens())
         {
             word.set(tokenizer.nextToken());
             context.write(word, one);
         }
    }
}
```

4.25.2 Reduce fonksiyonu

Bu fonksiyon map fonksiyonun çıktılarını girdi olarak almaktadır. Bundan dolayı map fonksiyonu reduce fonksiyonu için kaynak dosyadır denebilir. Reduce fonksiyonu çalıştıktan sonra çıktıları HDFS üzerinde saklanmaktadır.

Reduce fonksiyonu ile map fonksiyonunun aralarındaki girdi/çıktı ilişkisinden dolayı map fonksiyonun çıktısı ile reduce fonksiyonunun girdisi aynı olmalıdır. Reduce fonksiyonun işlevi girdi olarak aldığı gruplanmış ve sıralanmış veriyi alıp işlemesidir.

Yukardaki kod parçacığından da anlaşılacağı üzere map fonksiyonu girdi olarak map fonksiyonun çıktısı ile aynı olan anahtar/değer ikilisini almaktadır.

Map fonksiyonunda çıktı değeri olarak sadece int kullanılırken reduce fonksiyonu değer olarak int tipinde bir dizi almasının sebebi aynı anahtar değerine sahip çok fazla değer olmasıdır [3].

```java
package my.pack;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MyReducer extends Reducer <Text, IntWritable, Text, IntWritable>
{
    public void reduce (Text key, Iterable<IntWritable> values, Context context)
    throws IOException, InterruptedException
    {
        int sum =0;
        for (IntWritable value : values)
        {
            sum = sum +value.get();
        }

        context.write(key, new IntWritable(sum));
    }
}
```

Aşağıda bir diğer örnek reduce fonksiyon kodları bulunmaktadır:

```
public class DenemeReduceFonksiyonu extends MapReduceBase
        implements Reducer<Text, IntWritable, Text, IntWritable> {
                public void reduceMetod(Text anahtar, Iterator<IntWritable> deger,
                OutputCollector<Text, IntWritable> output, Reporter
                reporter)
                throws IOException {
                int maksimumDeger = Integer.MIN_VALUE;
                while (deger.hasNext()) {
                maksimumDeger = Math.max(maksimumDeger, deger.next().get());
                }
        output.collect(anahtar, new IntWritable(maksimumDeger));
        }
}
```

5. ANALİZ SONUÇLARI

5.1 Hadoop Impala, Hive Performanslar Değerlendirmesi

Hadoop üzerinde sql sorgularımızı çalıştırmak için Hive yaygın bir şekilde kullanılmaktadır. MapReduce kodları Hadoop üzerinde dağıtık sql sorgularının çalıştırılması için Hive'den çevrilmektedir. Bu çevrim sayesinde klasik RDMS sunucuları üzerinde çalıştıramayacağımız sorgulara çok kısa sürede cevap alabilmek mümkündür. Bunu kıyasladığımızda iyi bir donanıma sahip ve Mysql veritabanı kullanılan bir uygulamada 10 dakikada cevap aldığımız bir sorguya Hive sayesinde 130 saniyede cevap alınması gibi büyük farklar ortaya çıkmaktadır. Ancak günümüzde verilerin katlanarak artması Hive'in de yetersiz kalmasına sebep olmaktadır. Bununla birlikte MapReduce işlerine çevrilen Hive sorgularının aksine Impala sorguları çevrilememektedir. Bunun yerine dağıtık sorgu sistemlerinin tercih edilmesi yerinde olacaktır.

Yapılan testler ile farklı boyutlardaki veriler üzerinde hive ve impala performans yeterlilik testine tabi tutulmuştur. Testler aşağıdaki makinede gerçekleştirilmiştir.

Tablo.2- Server Özellikleri

CPU	Memory	Disk	L2 Cache	Fron-Side Bus Speed
Intel® Xeon® Processor E5472				
3 GHz, 4 Çekirdek	32GB	2TB	12MB	1600MHz

Aşağıda farklı türdeki sorgu örnekleri ve sorgu türlerine dair açıklamalar bulunmaktadır. Yapılan testler bu sorgular ile yapılmıştır.

Sorgu Türleri;

Q1: Tarama Sorgusu

Q2: Toplama sorgusu

Q3: İki tabloyu birleştirme sorgusu

Q4: Üç tabloyu birleştirme sorgusu

Örnek Sorgular;

Q1: SELECT SAYI(*) DEN sis_bos.aktif_epos_log WHERE name = 'Furkan Kayım';

Q2: SELECT transaction_type, sayi(*) cnt DEN sis_bos.aktif_epos_log GROUP BY transaction_type ORDER BY cnt DESC LIMIT 10;

Q3: SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue DEN (SELECT sis_bos.sis_ws_log.category AS book_category, SUM(sis_bos.sis_ws_log.price * sis_bos.aktif_epos_log.quantity) AS revenue DEN sis_bos.aktif_epos_log JOIN sis_bos.sis_ws_log ON(sis_bos.aktif_epos_log.book_id=sis_bos.sis_ws_log.id AND YEAR(sis_bos.aktif_epos_log.transaction_date) BETWEEN 2008 AND 2010) GROUP BY sis_bos.sis_ws_log.category) tmp ORDER BY revenue DESC LIMIT 10;

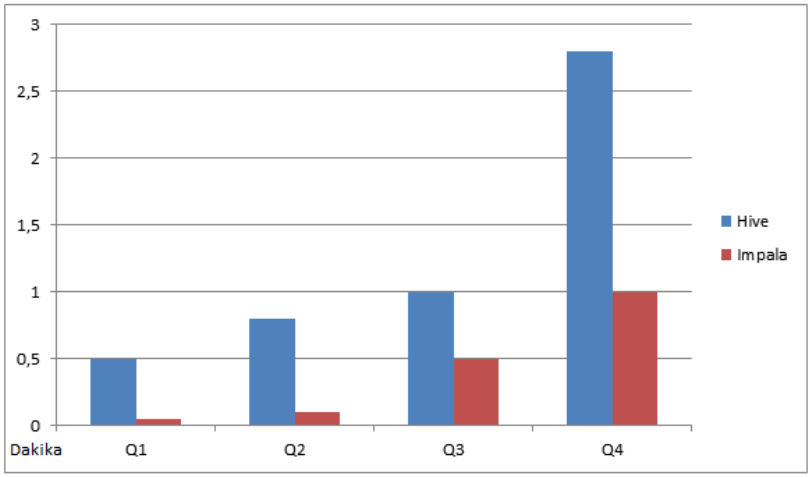
Q4: SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenueDEN (SELECT sis_bos.sis_ws_log.category AS book_category, SUM(sis_bos.sis_ws_log.price * sis_bos.aktif_epos_log.quantity) AS revenue DEN sis_bos.sis_ws_logJOIN [SHUFFLE] sis_bos.aktif_epos_log ON (sis_bos.aktif_epos_log.book_id = sis_bos.sis_ws_log.id)JOIN [SHUFFLE] sis_bos.sis_trx_conMetin_pool ON (sis_bos.aktif_epos_log.customer_id = sis_bos.sis_trx_conMetin_pool.id AND sis_bos.sis_trx_conMetin_pool.state IN ('WA', 'CA', 'NY'))GROUP BY sis_bos.sis_ws_log.category) tmp ORDER BY revenue DESC LIMIT 10

Tablo 3: Hadoop Impala ve Hive Performans Testi Tablo Boyutları

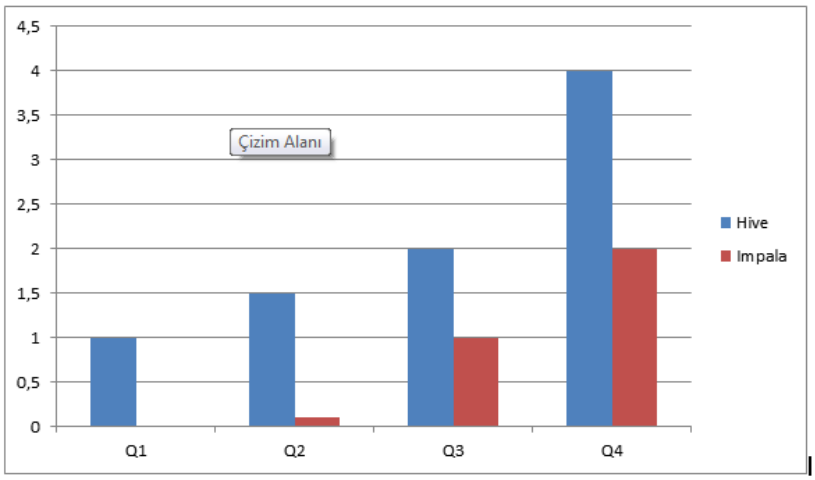
Tablo Özellikleri;

Her bir tablonun boyutu	Log tablosu (Milyon satır)	Log tablosu2 (Milyon satır)	Transactions tablosu(Milyon satır)
16GB	249	210	334
32GB	497	419	659

Yukarıdaki tabloda testlerin yapıldığı farklı boyutlardaki tablolara ait özellikleri göstermektedir.



Şekil 24. Impala ve Hive 16GB Tablo boyutu ile



Şekil 25. Impala ve Hive 32GB Tablo boyutu ile

Şekil 24 ve şekil 25 da 16GB ve 32 tablo boyutunda Hive ve Impala için Tablo2 deki özelliklere sahip bir bilgisayarda 4 farklı sorgu için performans değerlendirmesi yapılmıştır. Bu değerlendirmeye göre; Impala da çalıştırılan sorgular Q1 ve Q2 için hafıza yetersiz hatası alınmıştır. Diğer sorgular için Impala daha hızlı cevap vermiştir. Burada y ekseni zamanı gösterirken kırmızı renkli sütunlar Impala, mavi sütunlar Hive sonuçlarını temsil etmektedir.

Benzer sorgular ile yapılan farklı örneklere bakmak için [44] adresine bakılabilir.

Hive ve Impala birbirlerinin yerine kullanılabilir gibi görünse de gerçek dünyada ikisi aynı anda kullanılmaktadır. Hive çok daha büyük verileri işlemek için uzun süren sorgularda kullanılabilirken, Impala gerçek zamanlı sorgularda kullanılmaktadır.

Performanslar açısından Hive ve Impala değerlendirmesi;

1- Impala performanslar ve zaman bakımından daha kazançlıdır.
2- Join yapan sorgularda ve MapReduce kullanmamasından dolayı Impala performanslar olarak önce çıkmaktadır.
3- Basit sorgularda Impala daha fazla kaynak tükettiği için öne çıkmaktadır.
4- Kaynak kullanımı konusunda Hive daha kullanışlıdır.

5.2 Hive, Impala, İlişkisel Veri Tabanı Performans Testi

Aşağıda ilişkisel veri tabanları üzerinde ve hadoop sistemi üzerinde çalıştırılan aynı sorguların perfomans değerlendirmelerine değinilmiştir. Aynı sorgu aynı boyuttaki veri üzerinde çalıştırıldığı zaman hive, impala ve pl/sql için şekil 26 daki sürelerde cevap vermektedir.

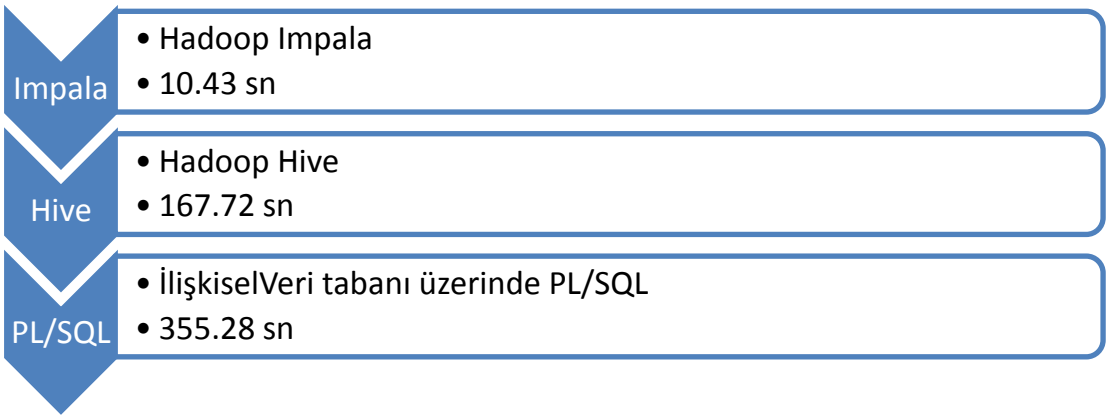
Sorgu: SELECT count(*) FROM sis_bos.sis_ws_log;

Veri: 16 GB

Pl-sql sorgu süresi: 355.28 sn

Hive sorgu süresi: 167.72 sn

Impala sorgu süresi: 10.43 sn



Şekil 26. Hive, Impala,PL/SQL performans değerlendirme

Yine benzer bir performas değerlendirmesini [33] kaynağından baktığımızda Mysql üzerindeki 60 milyon satır üzerinde mysql sorgusu, Hive sorgusu ve Impala sorgusu çalıştırılmıştır. Bu teste ait veriler aşağıdaki gibidir.

Sorgu: SELECT SAYI(*) FROM comments;

Veri: 8 GB

Mysql sorgu süresi: 138.82 sn

Hive sorgu süresi: 62.87 sn

Impala sorgu süresi: 3.74 sn

Yukarıdaki iki işlem sonucu değerlendirildiğinde:

- Impala, Hive'e göre daha hızlı cevap vermektedir.
- Hive klasik veritabanları üzerinde çalışan sorgulara göre daha çabuk cevap vermektedir.
- Hive ve Impala arasındaki oran veri miktarı değiştiğinde de yaklaşık olarak aynı kalmaktadır.

5.3 K-means Paralleştirme Performanslar Değerlendirmesi

Alsabti, Ranka ve Singh'ın kapsamlı bir şekilde belirttiği üzere [45] K-means algoritmasında aranan küme sayısını ifade eden k önceden bilinen ve kümeleme

işlemi bitene kadar değeri değişmeyen bir sabit olmalıdır [46]. Kümeleme işlemi başlangıcında küme merkezlerini temsilen k adet rastgele nokta seçilir. Bu noktaların her biri prototip olarak adlandırılmaktadır. Kümeleme başlangıcında k adet prototip (w1,w2,w3 .. wk) ve her bir küme ya da örüntü (i1,i1,i3 .. in) olmak üzere

$$w_i = i_l, j \in \{1,\ldots,k\}, l \in \{1,\ldots,n\} \qquad (1)$$

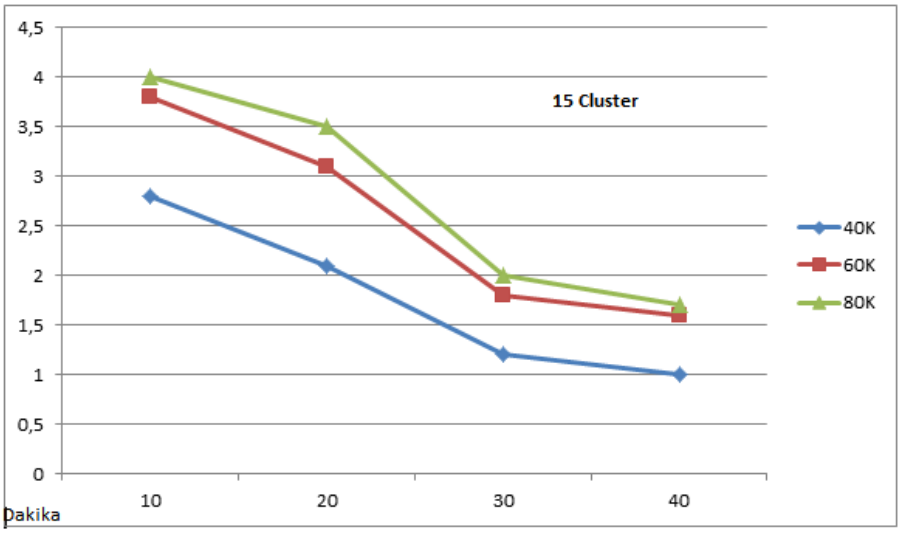
durumundadır.

K-means algoritmasının matematiksel yorumlanışına ilişkin aşağıda verilen açıklamalarda, Cj ifadesi j. elemanı temsil etmek üzere, kümeleme işleminin kalitesi Denklem 2'deki hata fonksiyonu ile ifade edilir [47]:

$$E = \sum_{j=1}^{k} \sum_{i_l \in C_j} \left| i_l - w_j \right|^2 \qquad (2)$$

Büyük ölçekli ve orta ölçekli işlerde veri madenciliği tekniklerinin paralelleştirilmesi ile daha iyi performanslar ortaya konulabilmektedir. Sadece paralel olarak bu teknikleri geliştirmek bir çözüm olmasa da yüksek performanslar elde edebilmek için bu dönemin veri madenciliği temel ihtiyaç olacaktır. K-means algoritmasının paralelleştirilmesi sonucunda hemen hemen paralel çalıştırmada kullanılan bilgisayar sayısı kadar performanslar kazanımı elde edilmiştir [48].

Şekil 27 de görülen deney sonuçları algoritmanın hız düğümlerine dağıtılmış veri kümesi boyutu olarak doğrusal değildir. Düğümler sayısı arttıkça ancak bazı durumlarda iki katına olduğunu göstermektir. Bu yaklaşım sürenin kısalması, kaynağın daha az kullanılması gibi performanslar kazanımları sağlamamıza yardımcı olacaktır. Bu da yaklaşımın başarısını göstermektedir. Bu konuda farklı örnekler için kaynağına bakılabilir.[49]



Şekil 27. K-means 15 cluster ile paralelleştirme

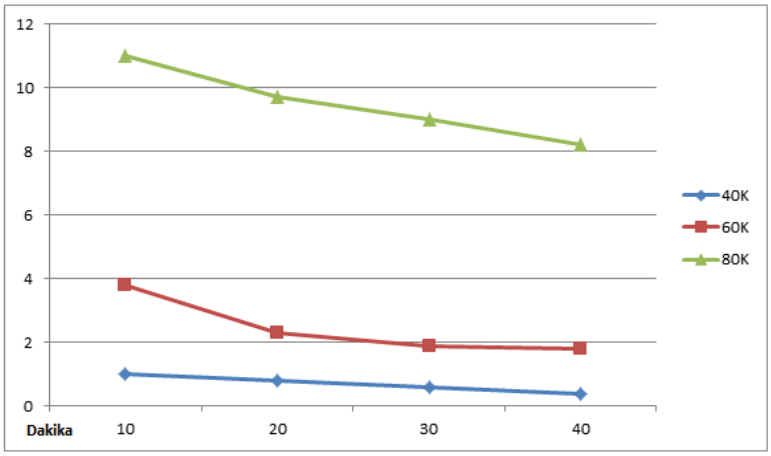
5.4 Mahout ile K-means Algoritmasının Çalıştırılması

K-means Algoritmasının Mahout üzerinde çalıştırılması ile büyük veriler üzerinde de K-means algoritmasını uyguluyor olacağız. Bu algoritmaların çalıştırılabilmesi için bazı komutları bilmek gerekmektedir. Aşağıda bu komutlardan bazıları hakkında bilgi verilmiştir. Detaylı bilgi için [50] adresine bakılabilir.

1- **input:** Giriş verileri içerin bir dosya dizini. (-i)
2- **Clusters:** Kmeans kümeleri ve Canopy için dosya dizini. (-c)
3- **Output:** Algoritma tüm sonuçlarının yazılacağı dosya dizini.(-o)
4- **distanceMeasure:** Kümelemede kullanılacak değişkendir.(-dm)
5- **convergenceDelta:** (-cd)
6- **maxIter:** Maksimum numaradır çalışma yenilemek için.(-x)
7- **runClustering:** Boolen değer alır eğer true ise K-means input verilerini kullnacaktır.(-cl)

5.5 DBSACAN Algoritması Paralelleştirme Performans Değerlendirmesi

DBSCAN algoritmasının paralelleştirmesine yönelik yapılan çalışmanın sonucu şekil 29 da gösterilmiştir.



Şekil 28. DBSCAN paralelleştirme performansları

Bu çalışma göstermektedir ki data sayısı arttıkça kümeleme süresinde artmaktadır. Hunain Durrani'nin 2013 yılında yaptığı DBSCAN algoritmasının paralelleştirmesi çalışması [49] da benzer sonuçları içermektedir. Aynı boyuttaki veri seti için düğüm sayısı arttıkça kümeleme sürenin kısaldığı gözlemlenmektedir. Düğüm sayısı artırılarak kümeleme daha hızlı bir hale getirilebilir. Büyük veriler için yapılacak çalışmalarda düğüm sayısı daha büyük tutulmalıdır.

SONUÇLAR VE ÖNERİLER

Teknolojinin hızla gelişmesi bilgisayar yazılımlarının sayısını ve kullanım oranını artırmakla birlikte, yazılımın pek çok sektöre girmesine zemin hazırlamıştır. Giderek gelişen bu yapı sonucu büyüyen veriler, hayatımıza "büyük veri" kavramının girmesine neden olmuş ve beraberinde iş süreçlerini olumsuz etkileyebilecek problemler getirmiştir. Nitekim yazlımın yayılması veri miktarını artırarak verilerin yönetilmesini zorlaştırmıştır ve bu sürecin yönetiminde veri madenciliğinde kümeleme, sınıflandırma gibi özel yöntemlerin kullanımını gerekli kılmıştır.

Veri madenciliği verileri işleyerek geleceğe yönelik tahminler yapılmasını sağlayan büyük veriler içerisinden bilgisayar programları vasıtası ile anlamlı verilere ulaşmamızı sağlayan işlemler bütünüdür. Bu kavram ışığında büyük veriler yönetimsel açıdan sınırlanabilir hale getirilmiş ve anlamlı veriye ulaşmamız kolaylaşmıştır.

Büyük veri problemine sunulan no-sql veri tabanları, hadoop benzeri teknolojiler ve paralelleştirme gibi pek çok çözüm bulunmaktadır. Bu çözümler veri boyutunun büyüklüğü ile alakalı olduğu kadar kullanılabilecek kaynaklarla da değişebilir. Ancak etkinlik bakımından değerlendirildiğinde veri madenciliği algoritmalarının paralelleştirmesi maliyetinin düşüklüğü ve uygulanmasının kolay olması noktalarında diğer çözümlerin önüne geçmektedir.

Veri madenciliği algoritmalarının paralelleştirilerek kullanılması sürecinde kullanılacak düğüm sayısı uygulama büyüklüğüne göre belirlenmelidir. Bu işleyiş uygulayıcıya maliyet yeterliliği ve performans kazanımı sağlamaktadır. Örnek olarak, veri madenciliği kümeleme algoritmalarından olan K-means ve DBSCAN algoritmasının paralelleştirmesinin büyük veriler üzerindeki etkileri incelenmiş ve paralelleştirmenin performans ve hız kriterlerini olumlu etkilediği görülmüştür. Buradan hareketle veri madenciliği kümeleme algoritmalarından K-means ve DBSCAN algoritmasını paralelleştirerek büyük verileri işlemede daha iyi performansların elde edilebileceğini söylemek mümkündür.

K-means ve DBSCAN algoritmalarının değişik düğmlerde paralelleştirmesi incelendiğinde düğüm sayısının artması ile kümeleme hızının da arttığı görülmüştür. Düğüm sayısı ile kümeleme hızı paralel bir şekilde artış göstermektedir. Orta ve küçük ölçekli projelerde paralelleştirme yöntemi ile veri madenciliği algoritmalarını paralelleştirip kullanabileceğimiz sonucuna buradan erişilebilmektedir.

Büyük veri problemine sunulan bir diğer çözüm yöntemi ise Hadoop teknolojisinden yararlanmaktır. Çalışma kapsamında yapılan uygulamalar neticesinde veri madenciliği algoritmaları ile Hadoop Mahout kullanarak büyük veriler üzerinde daha iyi performanslar elde edilebileceği görülmüştür. Hadoop üzerinde sorgu çalıştırmak için kullanılan Hive,bImpala Ve Pig teknolojileri de mevcut olup Hadoop'ta Hive ve Impala performanslar değerlendirmeleri ayrıca irdelenmiştir.

Hive, kaynak kullanımın kısıtlı olacağı, belirli bir zaman kısıtının olmadığı, sitemi fazla yormasını istemediğimiz sorgularımız için kullanılmaktadır. Impala ise kaynak kullanımın kısıtlı olmadığı yüksek performans ve hız beklenen, kısa sürede cevap alınması istenen sorgular için kullanıma uygundur. Tüm bunlara ek olarak Hive ve Impala sql sorguları ile çalıştırabilecek teknolojiler olduğu gibi Pig, veri ambarı üzerinde çalışan sorgularda kullanılmaktadır.

Uygulama aşamasında yapılan performans değerlendirmelerinde Hadoop kullanılmadan çalıştırılan pl-sql sorgusu 355.28 sn sürerken Hive sorgusunun 167.72 sn sürdüğü görülmüştür. Bu süreler göz önünde bulundurulduğunda Hive ile sorgu süresinin %50'nin üstünde bir azalma gösterdiği gözlemlenmiştir. Çalışma süresinin önemli olduğu sektörlerde ilişkisel veri tabanları yerine Hadoop veri tabanı olan hbase kullanarak Hive sorgusu çalıştırmak çok daha yerinde olacaktır. Ayrıca Hadoop ile HDFS dosya sitemi üzerindeki dosyalarda da sorgu çalıştırmanın mümkün olduğu unutulmamalıdır.

Yapılan bir diğer performans değerlendirme sorgusunda Hive ile 167.72 sn süren sorgunun Impala ile 10.43 sn sürdüğü gözlemlenmiştir. Hive ile Impala'da çalıştırılan sorgu cevap sürelerinde büyük farklılık olduğu gözlemlense de Impala

Hive'e göre kaynak tüketimi fazla olduğu için kaynaklarının fazla olduğu projelerde kullanılabilir.

Bilişim dünyasında veri boyutunun sürekli artması problemine veri madenciliği algoritmalarının paralelleştirilerek kullanılması ile bir çözüm getirilebilir. Ayrıca bu probleme hadoop üzerinde veri madenciliği algoritmalarının kullanılıyor olması da bir çözüm getirmektedir.

KAYNAKLAR

[1] (2015, Nisan.) Visual Networking Index (VNI). [Online]. http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] İlginç Demir, "Hadoop Tabanlı Büyük Ölçekli Görüntü işleme altyapısı," in *Yüksek Lisans Tezi*. Kocaeli Üniversitesi, 2012, p. 11.

[3] Harun Reşit Er, "Gezgin Satıcı Probleminin Hadoop üzerinde çalışan paralel genetik algoritma ile çözümü," in *Yüksek Lisans Tezi*. Kocaeli Üniversitesi, 2013, p. 7.

[4] Qiu J., Zheng Q., Zhong X., Li J., Li Y. Dong B., "A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop: A Case Study by PowerPoint Files," , 65-72, 2010.," in *Services Computing (SCC), 2010 IEEE International Conference*, 2010.

[5] P. Jacobs, "Data Mining: What General Managers Need to Know," in *Harvard Management Update*., 1999, pp. Cilt 4, No 10, 8.

[6] Ş., ve Türkoğlu,İ., Doğan, "Hypothyroidi and Hyperthyroidi Detection from Thyroid Hormone Parameters by Using Decision Trees ," *Doğu Anadolu Bölgesi Araştırmaları Dergisi*, pp. Cilt 5, No 2, 163-169., 2007.

[7] D.J., Hand, "Data Mining: Statistics and More?," in *The American Statistician*., 1998, pp. Cilt 52, 112-118.

[8] Kitler R. ve Wang W., "The Emerging Role of Data Mining," in *Solid State Technology*., 1998, pp. Cilt 42, No 11, 45.

[9] Nurettin TOPALOĞLU, Mithat YILMAZ Serkan SAVAŞ, "VERİ MADENCİLİĞİ VE TÜRKİYE'DEKİ UYGULAMA ÖRNEKLERİ," *21*, pp. 1-23, 2012.

[10] P. ve Zantinge, D. Adriaans, "Data Mining," in *MA, USA Addison Wesley Longman Publishing.* Boston, 1997.

[11] O. İnan, "Veri Madenciliği, Yüksek Lisans Tezi," in *Selçuk Üniversitesi, Fen Bilimleri Enstitüsü.*, 2003.

[12] M. Albayrak, "EEG Sinyallerindeki Epileptiform Aktivitenin Veri Madenciliği Süreci ile Tespiti," in *Doktora Tezi, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü.*, 2008.

[13] Ö. ve Çakır, F. Akgöbek, "Veri Madenciliğinde Bir Uzman Sistem Tasarımı," in *Akademik Bilişim 09, 11-13 Şubat Harran Üniversitesi*. Şanlıurfa, 2009, pp. 801-806.

[14] C. Shearer, "The Crisp-DM Model: The New Blueprint for Data Mining," *Journal of Data Warehousing*, pp. Cilt 5 No 4, 13-23., 2000.

[15] S. Özekes, "Data Mining Models and Application Areas," in *İstanbul Commerce University Journal of Science*., 2003, pp. No.3, 65-82.

[16] Nurdan Bayraktar,. İstanbul, 2013.

[17] U., Piatetsky-Shapiro G. and Symth, P. Fayyad, "From data mining to knowledge discovery in databases," in *AI Magazine*., 1996, pp. 37-54.

[18] J. Han and M. Kamber, "Data Mining Concepts and Techniques," in *Morgan Kaufmann Publishers Inc.*, 2001.

[19] Gökhan Silahtaroğlu, "Veri Madenciliği Kavramları," in *Veri Madenciliği Kavram ve Algoritmaları*., 2013, ch. 2, p. 21.

[20] Fatih Çil, *Banka Yatırım Fonu Müşteri Hareketlerinin Belirlenmesine Yönelik Bir Veri Madenciliği Uygulaması*, Gazi Üniversitesi, Ed. Ankara: Yüksek Lisans Tezi, 2010.

[21] Aslı Çalış, *VERİ MADENCİLİĞİ YAKLAŞIMI İLE BİREYSEL MÜŞTERİLERİN KREDİ ÖDEME PERFORMANSLARININ DEĞERLENDİRİLMESİ*, Kocaeli Üniversitesi, Ed. Kocaeli, 2013.

[22] Bilen H., *Bankacılık sektöründe personel seçimi ve performans değerlendirilmesine ilişkin veri madenciliği uygulaması*. Ankara: Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, 2009.

[23] Esra Dinçer, *Veri Madenciliğinde K-means Algoritması ve Tıp Alanında uygulanması*, Kocaeli Üniversitesi, Ed. Kocaeli, 2006.

[24] M., Kriegel, H. P., Sander, J., Xu, X., Ester, "A density based algorithm for discovering clusters in large spatial databases," in *Int. Conference of Knowledge Discovery and Data Mining (KDD'96)*. Portland, USA, 1996, pp. 226-231.

[25] Yılmaz ÇAMURCU Turgay Tugay BİLGİN, "DBSCAN, OPTICS ve K-Means Kümeleme Algoritmalarının Uygulamalı Karşılaştırılması," *Politeknik Dergisi*, pp. 139-145, 2005.

[26] [Online]. http://en.wikipedia.org/wiki/Big_data

[27] Cüneyt Göksu. [Online]. http://datawarehouse.gen.tr/big-data-nedir-geleneksel-veri-yonetimine-etkisi-ne-olur/

[28] (2015, Mart) wikipedia. [Online]. http://tr.wikipedia.org/wiki/Bulut_bili%C5%9Fim

[29] C. Lam,. USA, 2010, ch. 9781935182191.

[30] (2015, Mayıs) wiki. [Online]. http://wiki.apache.org/hadoop/PoweredBy

[31] (2015, Mart.) devveri. [Online]. http://devveri.com/hadoop/apache-pig-domuzcugun-hikayesi

[32] Melih Koca, "The costs and benefits of turning data into information using bigdata system," in *Yüksek Lisans Tezi, Ozyegin University*., 2014, ch. s.16.

[33] Hakan İlter, "Pig ve Hive ile Veri Analizi," *Türkiye Hadoop group*, pp. 38-39, Jan. 2015.

[34] (2014, Kasım.) cloudera. [Online]. http://cloudera.github.io/hue/docs-2.1.0/manual.html

[35] Gobioff H., Leung S. Ghemawat S., "The Google File System," in *Proc. of the 19th ACM Symp. on Operating System Principle*., 2003, pp. 29–43.

[36] Dean J. and Ghemawat S.,., 2008, ch. 51, pp. 107–113.

[37] Güven Fidan, "Büyük Veri de Türkiye'den Uygulama Örnekleri," in *ARGEDOR Bilişim Teknolojileri*., pp. 10-12.

[38] (2015, Mayıs) mahout.apache. [Online]. https://mahout.apache.org/users/basics/algorithms.html

[39] (2015, Ocak.) hadoop. [Online]. https://hadoop.apache.org/

[40] T. White,. Inc. USA., 2009 , ch. 978-0-596-52197-4.

[41] Özal Serkan,., 2013, p. 5.

[42] R. H., Goldberg, D. E., Llora, X., Verma, A. Campbell,. Pisa, Italy, 2009, 30 Kasım – 2 Aralık.

[43] (2015., Nisan.) amazon. [Online]. http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/impala-optimization.html,

[44] (2015, Nisan.) amazon. [Online]. http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/impala-optimization.html

[45] Sanjay R., Vineet S. Khaled A., "An Efficient K-Means Clustering Algorithm," in *IPPS: 11th International Parallel Processing Symposium*, p. 1998.

[46] L., Rosseeauw, P.J. Kaufman,. New York, USA, 1990.

[47] Pavel Berkhin, "Survey of Clustering Data Mining Techniques," in *Accrue Software Inc.* San Jose, California, USA, 2002.

[48] Erhan Sülün, *IMPROVEMENTS IN K-MEANS ALGORITHM TO*, İzmir Enstitü, Ed. İzmir, 2004.

[49] Hunain Durrani,. Yüksek Lisans Tezi, Middle East Tecnical Univesity, 2003, p. s.38.

[50] [Online]. https://mahout.apache.org/users/clustering/k-means-clustering.html

[51] (2015, Nisan.) j2eedev. [Online]. http://j2eedev.org/ecosystem-hadoop-animal-zoo/

[52] C. Lam,. USA, 2010, ch. 9781935182191.

[53] (2014, Kasım.) docs.oracle. [Online]. http://docs.oracle.com/cd/E24628_01/install.121/e22624/preinstall_req_cygwin_ssh.htm#EMBSC152

[54] (2014, Kasım.) java. [Online]. https://java.com/tr/download/help/windows_offline_download.xml

[55] (2015 Şubat) [Online]. https://github.com/prabaprakash/Hadoop-2.3-Config/archive/master.zip

[56] (2015 Şubat) [Online]. https://app.box.com/s/11fwozokqmc1ohttt117

[57] codeproject. [Online]. http://www.codeproject.com/Articles/757934/Apache-Hadoop-for-Windows-Platform

[58] (2015 Mayıs)[Online]http://guyharrison.squarespace.com/blog/tag/hive

[59] (2015 Mayıs)[Online] http://www.rabidgremlin.com/data20/#(3)

EKLER

Şekil 29. Hadoop Ekosistemi **[51]**

ÖZGEÇMİŞ

Furkan Kayım, 1988 yılında Yozgat'ta doğdu. Yozgat Anadolu Lisesi Sayısal Bölümü'nde lise eğitimini tamamladıktan sonra 2007 yılında Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümü'nde lisans eğitimine başladı. Lisans süresince bilgisayar ile ilgili uygulama ve eğitim bilimleri konularında eğitim aldı. 2012 yılında mezun olarak bilişim sektöründe özel bir firmada kariyerine devam etti. "Yazılım Uzmanı" olarak sektörde üç yıldır çalışmaya devam etmekte olup 2013 yılında başladığı Beykent Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği anabilim dalı yüksek lisans eğitimini sürdürmektedir.

Furkan KAYIM