BÜYÜK VERİ ANALİZİNDE DAĞITIK MAKİNE ÖĞRENMESİ ALGORİTMALARININ KULLANILMASI
YÜKSEK LİSANS TEZİ

İbrahim Rıza HALLAÇ

Anabilim Dalı: Bilgisayar Mühendisliği
Programı: Kuramsal Temeller
Danışman: Yrd. Doç. Dr. Galip AYDIN

Temmuz-2014

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK VERİ ANALİZİNDE DAĞITIK MAKİNE ÖĞRENMESİ ALGORİTMALARININ KULLANILMASI

YÜKSEK LİSANS TEZİ

İbrahim Rıza HALLAÇ

111129109

Anabilim Dalı: Bilgisayar Mühendisliği
Programı: Kuramsal Temeller

Danışman: Yrd. Doç. Dr. Galip AYDIN

Tezin Enstitüye Verildiği Tarih: 15 Temmuz 2014

Temmuz-2014

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK VERİ ANALİZİNDE DAĞITIK MAKİNE ÖĞRENMESİ ALGORİTMALARININ KULLANILMASI

YÜKSEK LİSANS TEZİ

İbrahim Rıza HALLAÇ

111129109

Tezin Enstitüye Verildiği Tarih: 15.07.2014
Tezin Savunulduğu Tarih: 01.08.2014

Tez Danışmanı : **Yrd. Doç. Dr. Galip AYDIN**

Diğer Jüri Üyeleri : **Prof. Dr. Mehmet KAYA**
Prof. Dr. İbrahim TÜRKOĞLU

TEMMUZ-2014

ÖNSÖZ

İnternet ağındaki bilgisayar sayısının 2020 yılı itibariyle 24 milyara ulaşacağı ve bu bilgisayarların büyük çoğunluğunun birbirleriyle bulut sistemi üzerinden haberleşeceği tahmin edilmektedir. YouTube video paylaşım sitesine geçen her dakikada 48 saat uzunluğunda video yükleniyor, Büyük Hadron Çarpıştırıcısı deneyi yılda 15 petabayt büyüklüğünde veri üretiyor, Boeing jet motorları 30 dakikalık bir uçuş ile 10 terabayt büyüklüğünde veri üretiyor ve dünyada her biri yaklaşık 10 petabayt büyüklüğündeki 40 milyar adet web sitesine erişiliyor olması gibi gelişmeler büyük veri problemini oldukça önemli bir noktaya taşımıştır.

Bu tez çalışmasında bulut bilişim konusu ele alınmış, büyük verilerin yönetimi ve büyük veriden otomatik olarak faydalı bilgilerin çıkartımı konusundaki yeni teknoloji ve yöntemler incelenmiştir. Büyük verilerden otomatik olarak faydalı bilgi çıkarımında kullanılan makine öğrenmesi teknikleri ve bu yöntemlerin dağıtık bir sistem üzerinde uygulanışı yine uygulamalar geliştirilerek incelenmiştir.

Bu tez çalışması boyunca ilgi ve yardımlarını esirgemeyen danışman hocam, Sayın **Yrd. Doç. Dr. Galip AYDIN**'a teşekkür ve şükranlarımı sunarım. Ayrıca hayatım boyunca bana destek olan aileme çok teşekkür ederim.

İbrahim Rıza HALLAÇ
ELAZIĞ - 2014

İÇİNDEKİLER

ÖZET

Farklı kaynaklarca üretilen verinin türünde, miktarında ve oluşturulma hızındaki artış bu verilerin yönetimi ve onlardan yararlanılması amacıyla yeni teknoloji ve yöntemlerin doğmasına sebep olmuştur. Büyük ölçekteki verilerin standart bilgisayarlar ile saklanması veya işlenmesi mümkün değildir. Büyük miktarda veri üzerinde çalışmak ve bu veriler üzerinde analiz yapabilmek için yüksek hesaplama gücüne ihtiyaç duyulmaktadır. Bu sebepten dolayı artık geleneksel hesaplama yaklaşımları yerine bilgisayar kümeleri, dağıtık dosya sistemleri (HDFS, RDD); geleneksel programlar ve programlama dilleri yerine ise Hadoop, Spark gibi platformlar yaygınlaşmaktadır. Verinin saklanmasında ve işlenmesinde kullanılmaya başlanan bu teknolojiler verilerin analizinde kullanılan makine öğrenmesi yöntemlerini de etkilemiştir. Duygu analizi, doğruluk tespiti, tavsiye sistemleri, sosyal ağ keşifleri, tıbbi keşifler, web içeriklerinin keşfi ve sınıflandırılması gibi birçok amaçla makine öğrenmesi algoritmaları kullanılmaktadır. Bu tez çalışmasında büyük veri ve bulut bilişim teknolojileri, büyük veri üzerinde paralel algoritmaların çalıştırılması ve dağıtık makine öğrenmesi algoritmalarının büyük veriye uygulanması incelenmiştir.

Anahtar kelimeler: Büyük veri, dağıtık makine öğrenmesi

SUMMARY

Using Distributed Machine Learning Algorithms on Big Data Analysis

The increase in the variety, volume and the velocity of the data produced by different sources caused new technologies and approaches to be born. It is not possible for the big data to be stored and processed by standard computers. We need high computational powers to study and analyze big data. For these reasons computer clusters and distributed file systems instead of traditional computing approaches and frameworks such as Hadoop and Spark instead of traditional programs and programming languages are becoming more common. These technologies which are used to store and process data have also affected the machine learning approaches used for data analysis. Machine learning algorithms are used for many areas such as sentiment analysis, accuracy detection, recommendation systems, social network discovery, discovery and classification of web content etc. In this thesis big data and cloud computing technologies, running parallel algorithms on big data and the application of distributed machine learning algorithms on big data are investigated.

Keywords: Big data, distributed machine learning

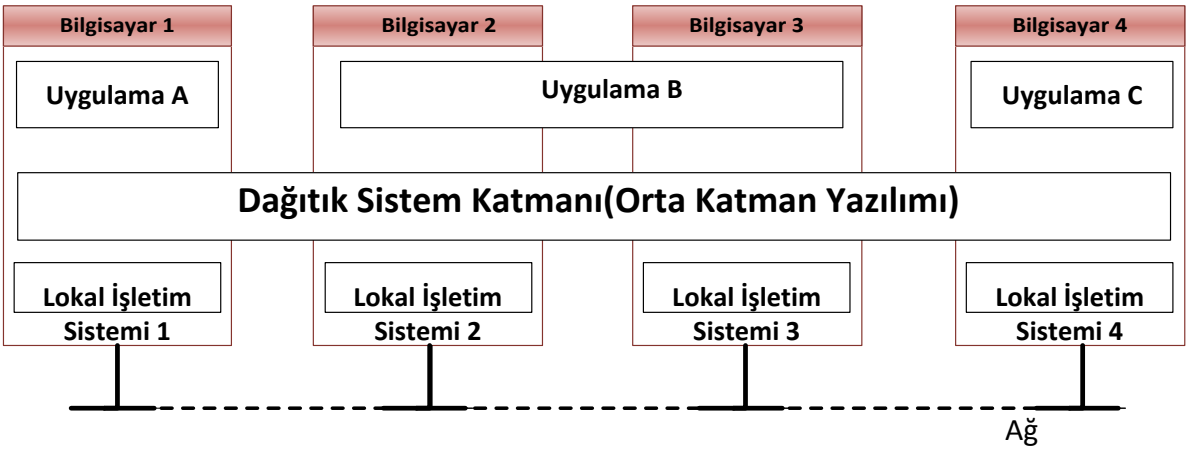
ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

1. GİRİŞ

Literatürde dağıtık sistemlerin birbirinden farklı çeşitli tanımları bulunmaktadır. En genel tanımıyla bir dağıtık sistem; kullanıcısına tek bir bilgisayarmış gibi görünen ve bağımsız bilgisayarlardan meydana gelen sisteme denilir [1]. Bu tanımda dikkat edilmesi gereken bazı önemli noktalar vardır. Bunlar bir dağıtık sistemin bağımsız bileşenlerden (örneğin bilgisayarlardan) oluşuyor olması, kullanıcıların (insan ya da programlar) tek bir sistemle çalıştıklarını düşünmesi ve bu bağımsız bileşenlerin nasıl tek bir cihaz gibi çalışacağının gerçekleştirileceğidir. Bir dağıtık sistemde bilgisayarlar arasındaki farklılar ve iletişim katmanıyla ilgili detaylar mümkün olduğunca kullanıcıdan gizlenmeye çalışılır.

Tek bir sistem görünümünde olacak olan heterojen bilgisayarların ve ağların desteklenebilmesi için dağıtık sistemler genellikle bir yazılım ara katmanıyla tasarlanırlar. Bu katman mantıksal olarak kullanıcıların ve uygulamaların üzerinde ve işletim sistemi ve temel iletişim yeteneklerinin altında bir katmanda yer alır.



Şekil 1.1. Dağıtık sistem mimarisi [1]

Şekil 1.1'de bir 4 bilgisayarın ağ üzerinden birbirine bağlı olduğu bir dağıtık sistem örneklenmiştir. Bu dağıtık sistemde üç tane uygulama çalıştırılmaktadır. Bu uygulamalardan B uygulaması 2. ve 3. Bilgisayarlar üzerinde dağıtılarak çalıştırılmaktadır.

Moore kanunu, bir işlemcide bulunan transistör sayısının katlanarak artacağını öngörmüştür [2]. Bu kanunda öngörülen transistör artışı sayısında güncellemeler olmasına rağmen günümüzdeki son kullanıcı bilgisayarlarının hesaplama güçlerinin üssel (exponansiyel) olarak arttığına şahit olmaya devam etmekteyiz. Fakat fiziksel dünyanın sınırları göz önüne alındığında bu artışın sonsuza kadar devam edemeyeceği de aşikârdır [3].

Bir işi oluşturan işlemlerin birden fazla işlemcinin hesaplama gücü kullanılarak yürütülmesine paralel hesaplama denir. Paralel hesaplama dağıtık hesaplamanın bir dalı olarak kabul görmüştür. Paralel hesaplama, karmaşık işlemlerin yürütülme sürelerinin azaltılması amacıyla kullanılmaktadır. Bu yönden bakıldığında hızlı sonuç elde etmek amacıyla veya daha büyük problemlerin çözümünde paralel hesaplamaya ihtiyaç duyulduğu görülür.

Literatürde farklı yönler üzerinden çeşitlere ayrılmış olan paralel hesaplama [4] verilerin paralelleştirilmesi ve işlerin paralelleştirilmesi olarak düşünülebilir [5]. Belirli bir veri üzerinde yapılacak işlemler işlemcilere verilerin dağıtılması ve her işlemcinin aynı işlemleri gerçekleştirmesi yoluyla paralel hesaplama yapıldığı gibi yapılacak işlemlerin de işlemciler arasında paylaştırılması yoluyla paralel hesaplama yapılabilir.

Kullanıcıya tek bir bilgisayar gibi görünen dağıtık sistemlerde sistemi oluşturan birimlerin kullanıcıdan gizlenmesi esası söz konusudur. Sistemin bu özelliğine saydamlık (transparency) denilir. Dağıtık sistemler farklı saydamlaştırma formları barındırmaktadırlar. Bunlar Tablo 1.1'de gösterilmiştir.

Tablo 1.1. Dağıtık bir sistemdeki farklı saydamlık formları [1]

Saydamlık	**Tanımı (Description)**
Erişim	Verilerin gösterimindeki farklılıkların ve kaynağa nasıl erişildiğinin gizlenmesi
Konum	Kaynağın konumunun gizlenmesi
Göçerme	Bir kaynağın farklı bir konuma taşınmasının gizlenmesi
Yer değiştirme	Bir kaynağın kullanıldığı sırada farklı bir konuma taşınmasının gizlenmesi
Yineleme	Bir kaynağın birden fazla kopyasının olmasının gizlenmesi
Paralel erişim	Bir kaynağın birden fazla kullanıcı tarafından paylaşılıyor olmasının gizlenmesi
Hata	Hataların ve kaynağın kurtarılmasının gizlenmesi

1.1. Grid Mimarisi

Grid hesaplama; dağıtık ve büyük ölçekli bilgisayar kaynaklarından oluşan bir paralel işleme altyapısı ve araçları olarak tanımlanabilir [6]. Kaynaklar (bilgisayar, bellek, özel cihazlar, servisler) grid sistemine dinamik olarak dahil olabilir ve sistemden ayrılabilir. Geniş alan ağlarıyla birbirine bağlı olan kaynaklar genellikle coğrafi olarak farklı konumlarda olurlar. Belirli kullanıcılar bu sistemlere isteğe bağlı olarak erişebilmektedir.

1.2.Servis Odaklı Mimari

Servis odaklı mimari (SOA) farklı kaynak ve hizmetlerin bir servis olarak sunulduğu yapıdır. SOA mimarisi senkron veya asenkron bir şekilde istek / cevap prensibiyle çalışır. Birçok heterojen sistemin bir arada kullanılmasına olanak sağlayan bu yapı dağıtık sistemler için çok önemli bir yaklaşımdır. İşlemci, hafıza, depolama alanı, işletim sistemi gibi bir çok farklı kaynak veya hizmet SOA ile sunulabilmektedir.

1.3. Bulut Bilişim

Grid hesaplamadaki bazı sorunların giderilmesi yoluyla geliştirilen bulut bilişim ve grid kavramlarını birbirinden ayırmak oldukça zordur. Grid yapısında coğrafi olarak birbirinden farklı konumlarda bulunan kaynakların veya kaynak kümelerinin, bu kaynakları yöneten kullanıcılar tarafından hesaplama kümesine dahil edilip edilmeyeceği zamanlara karar vermesi şeklinde bir çalışma söz konusudur [7] . Grid yapısı coğrafi olarak farklı mekanlarda bulunan donanımlar üzerinde kurulduğu gibi gridi kullanan kişiler de konumdan bağımsızdırlar. Farklı grid havuzları birleştirilerek süper bilgisayar kapasitesindeki sistemler oluşturulup bu sistemin tek bir bilgisayar gibi davranması sağlanabilir.

Grid sistemi üzerinde çoğu zaman birden fazla iş paralel olarak yürütülür. Bu işlerin yürütülmesinin zamanlanması ve kaynak kullanımının dağılımının belirlenmesinin dinamik olarak gerçekleştirilmesi oldukça karmaşıktır. Bir diğer sorun da her ağ tabanlı sistemde olduğu gibi grid için de söz konusu olan güvenlik konusudur.

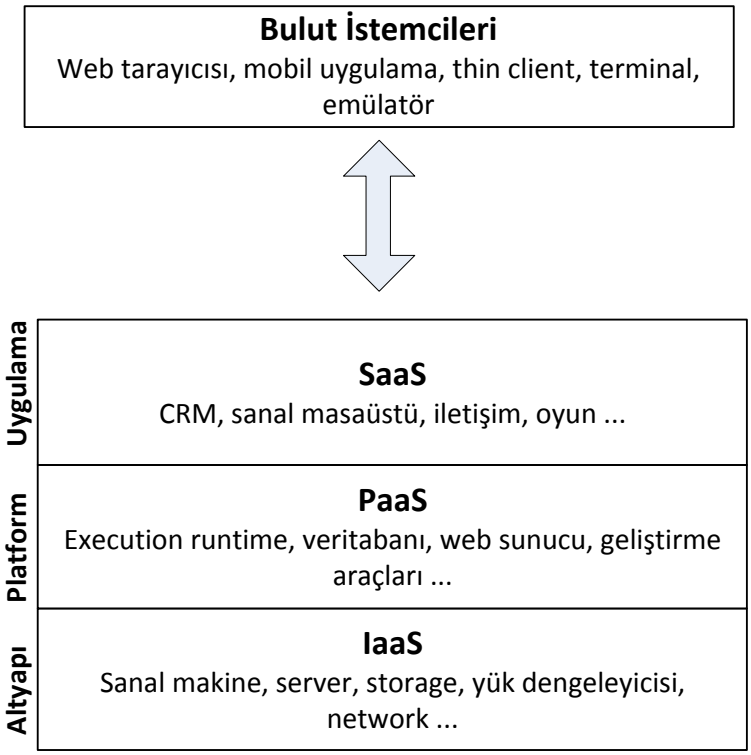
İnternetin bulutlardan oluşan gökyüzüne benzemesinden yola çıkarak bulut ismini alan bulut bilişim için oldukça çeşitli tanımlar yapılmıştır. Bulut internet demektir [8] . Ağ diyagramlarında interneti göstermek için bulut resmi kullanılır. İnternet üzerinden birbirine bağlanmış, birlikte çalışan, dinamik olarak yönetilebilen, izlenebilen ve bakımı yapılabilen sanal sunucuların oluşturduğu sistemin adıdır. Kullanıcılar bulut üzerinde kendi sanal imajlarını oluşturup kullanabilir ya da bulut üzerinde var olan kendi ihtiyaçlarına uygun bir imajı sanal makine olarak çalıştırabilir [9]. Bir sanal makine, üzerinde bulunduğu kaynaktan bağımsız bir ortam oluşturabilir. Kaynak üzerinde oluşturulan sanal makineler aynı zamanda birbirinden de bağımsızdır. Bu bağımsızlık hem uygulama seviyesinde hem de kaynakların kullanımı seviyesinde geçerlidir. Örneğin bir kaynak üzerinde oluşturulan sanal makineler farklı işlemci kapasitesine, hafızaya, disk kapasitesine veya bant genişliğine sahip olabilirler. Kaynakların bu şekilde yönetilebilmesi bulut bilişimin en önemli avantajlarından birisidir. Bu şekilde kullanıcılara ya da uygulamalara gereken miktarda kaynak ayrılması sağlanır ve bu da tıpkı evlere gelen elektrik enerjisinin elektrik faturasına yansıması hadisesindeki faydalanma esasına göre maliyet oluşturmayı sağlar.

Kaynak paylaşımı ne kadar esnek olursa kullanımın ölçeklendirilmesi o kadar fazla olur. Bulut sisteminde kaynak ve alt yapının aşağıdaki katmanlardan oluştuğu düşünülebilir.

1. **Uygulama**: Bulut üzerinde çalıştırılan yazılımın kendisidir.
2. **Veri**: Uygulamanın kullandığı veriler.
3. **Çalışma anı**: Uygulama kodlarının yürütülme anı.
4. **Ara katman yazılımı**: İşletim sistemi üzerinde uygulamanın çalıştırılabilmesi için gerekli ara birim.
5. **İşletim sistemi**: Sanal makinenin işletim sistemidir.
6. **Sanallaştırma**: Sanal makine imajlarının oluşturulması ve başlatılması işlemleridir.
7. **Sunucu:** Kaynakları barındırır.
8. **Depolama**: Sunucunun sabit disk kapasitesini harici olarak arttırmayı sağlayan birim.
9. **Ağ**: İnternet ağı.

1.4. Bulut Servisleri

Bulut bilişim sağlayıcıları farklı birkaç temel model üzerinden servis sunmaktadır [10, 11]. Bu modeller sunulan hizmete İngilizcede "servis olarak" anlamına gelen "as a service" ön eki getirilerek adlandırılmışlardır. Uygulanan üç temel model Servis Olarak Altyapı (IaaS), Servis Olarak Platform (PaaS) ve Servis Olarak Yazılım (SaaS)'dır.



Şekil 1.2. Bulut hizmet modelleri [12]

1.4.1. Servis Olarak Altyapı (IaaS)

Bulut sistemindeki ağ, depolama, sunucu ve sanallaştırma işlemlerinin bir bütün olarak sunulduğu hizmet modelidir. Bu modelde hizmeti alan taraf donanımsal alt yapıdan tamamen soyutlanır. Hizmet alan kullanıcı bu alt yapı üzerinde sanal makinelerini (sunucularını) başlatır, bu sanal makineler üzerinde işletim sistemi dâhil olmak üzere ihtiyaç duyduğu yazılımların kurulumunu gerçekleştirir. Bu sayede ihtiyaç duyulan fiziksel alt yapı bir servis olarak sunulur. Bu alt yapıların sıfırdan oluşturulması, servis olarak kiralanmasına göre çok daha pahalıdır. Satın alınan

servisler ihtiyaç duyulan süre kadar kullanılır ve kullanım miktarına göre ödeme esasına dayanır.

Amazon EC2 servis olarak altyapı sunan kar amaçlı bir web servistir. Bu alanda en bilinenlerden olan Amazon EC2'nun dışında Azure, Rackspace, HP Cloud, Oracle Servis Olarak Altyapı gibi çok sayıda IaaS (Servis Odaklı Altyapı) bulunmaktadır.

Tablo 1.2. Amazon bulut hizmeti kaynak çeşitleri [8]

Type	Arch.	CPU	Cores	Memory	Network	Storage	Price
m1.small	32-bit	2.0-2.6 GHz Opteron	1/2	1.7 GB	1-Gbps Ethernet	Local disk	$0.10/hr
m1.large	64-bit	2.0-2.6 GHz Opteron	2	7.5 GB	1-Gbps Ethernet	Local disk	$0.40/hr
m1.xlarge	64-bit	2.0-2.6 GHz Opteron	4	15 GB	1-Gbps Ethernet	Local disk	$0.80/hr
c1.medium	32-bit	2.33-2.66 GHz Xeon	2	1.7 GB	1-Gbps Ethernet	Local disk	$0.20/hr
c1.xlarge	64-bit	2.33-2.66 GHz Xeon	8	7.5 GB	1-Gbps Ethernet	Local disk	$0.80/hr
abe.local	64-bit	2.33 GHz Xeon	8	8 GB	10-Gbps InfiniBand	Local disk	N/A
abe.lustre	64-bit	2.33 GHz Xeon	8	8 GB	10-Gbps InfiniBand	Lustre	N/A

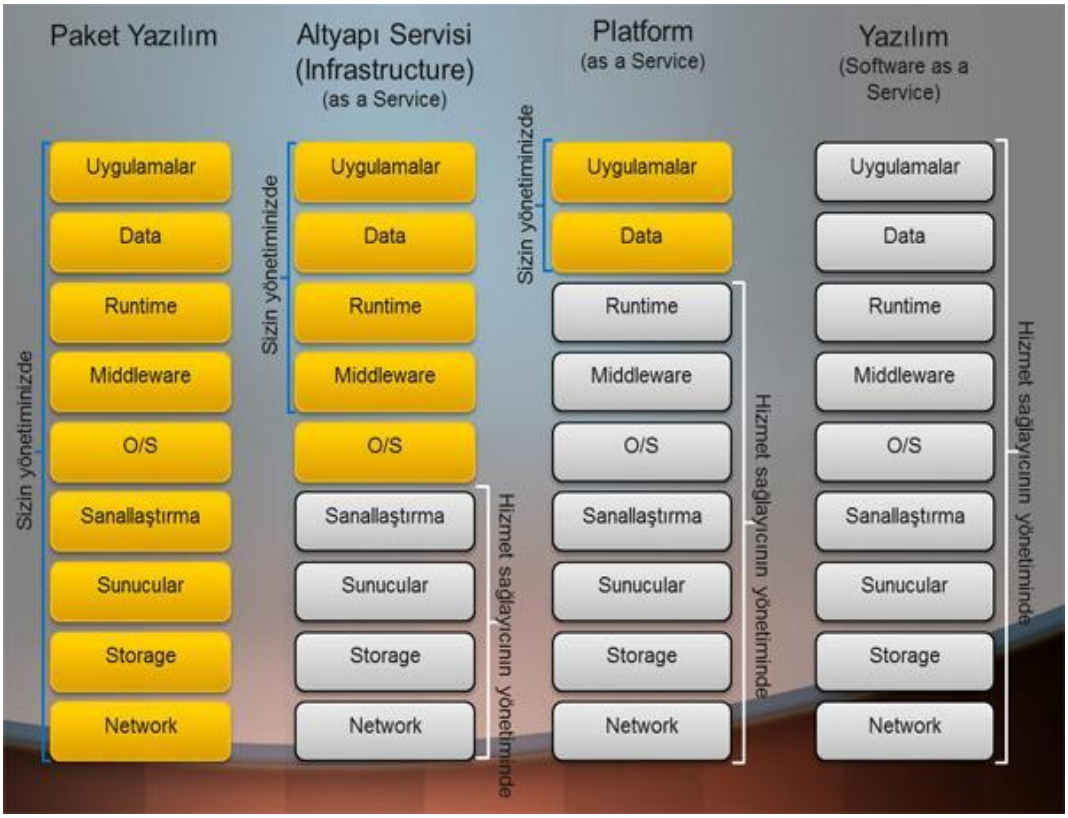
Tablo 1.2'de Amazon EC2 'da sunulan 5 kaynak çeşidi karşılaştırmalı olarak gösterilmiştir. Bu seçeneklerden istenilen adette sanal makine kiralanıp bu makineler üzerinde tercih edilen bulut işletim sistemi imajı çalıştırılır.

1.4.2.Servis Olarak Platform (PaaS)

Bir diğer bulut modeli olan Servis Olarak Platform (PaaS) hizmetinde uygulamalarımızı çalıştıracağımız donanımlarla birlikte gerekli işletim sistemi (Windows Server, Ubuntu Server) ve platform yazılımları (.Net, JDK, SQL) hazır olarak sunulur.

1.4.3.Servis Olarak Yazılım (SaaS)

Servis Olarak Yazılım (Saas) bulut modelinde kullanılan hizmetin tüm bileşenleri sağlayıcı tarafından sunulur. Microsoft firmasının Ofis ürününü satın almak yerine aylık bir ücret ödeyerek kiralama yoluyla kullanmak SaaS kullanımına bir örnektir.



Şekil 1.3. IAAS, PAAS, SAAS ve Windows Azure [12]

Bulut teknolojisi teknoloji devi firmalar tarafından çeşitli konularda kullanılmaktadır [9]. Online veri saklama, sosyal ağ, e-ticaret, web tabanlı e-mail servisleri gibi birçok uygulamalar için Google, IBM[13] Amazon[14] Yahoo, Microsoft gibi şirketler bulut bilişimi kullanmakta ve bu alandaki çalışmalarını arttırmaktadır [15].

1.5. Bilim Bulutları

Dünyanın önde gelen ticari şirketleri bulut altyapılarını hızla geliştirmektedir. Amazon EC2, RockSpace alt yapı servisi (Infrastucture as Service), Google App Engine, Microsoft Azure platform servisi (Platform as a Service) ve Salesforce.com, Google Docs, Application Portals gibi bulut sistemi üzerinde çalışan uygulama servisleri (Software as a Service) verilebilecek örneklerdir. Bu teknoloji şirketleri bulut hizmetlerini kendi ihtiyaçlarını karşılamak amacıyla kullandıkları gibi sahip oldukları alt yapıyı kiralama usulüyle müşterileriyle de paylaşmaktadırlar[16].

Alt yapı ve platform hizmetleri kullandığın kadar öde (utilization) mantığıyla müşteriye sunulduğundan dolayı çok uygun fiyatlara hizmet sunulabilmektedir [17]. Eski gelenekselleşmiş sistemde ne kadar hesaplama gücü, hafıza alanı ve band genişliğine ihtiyaç duyuluyorsa o kadar kaynak kiralanır fakat bu kaynaklar kullanılmadığı zamanlar için de rezerve edildiği için yüksek maliyetli olurdu. Bulut sisteminde ise bilgisayar kaynakları adeta bir elektrik kaynağı gibi kullanım kadar faturaya yansıyan bir hizmet haline gelmiş oldu [18].

Ticari bulut servisleri aynı zamanda yüksek hesaplama kapasitesi, yüksek disk alanı veya yüksek band genişliğine ihtiyaç duyan bilimsel çalışmalar [8] ve araştırmalar için de bir çözüm teşkil etmektedir[16, 19].

Ticari olan alt yapı ve platform servislerinden yola çıkarak bilimsel çalışmalar için yeni bulut bilişim çözümleri üzerinde çalışmalar yapılmaktadır [10]. Paralel uygulamaları sanallaştırılmış ortamlar üzerinde çalıştırma esasına dayanan bulut teknolojisi alanında açık kaynaklı platformlar geliştirilmiştir. OpenStack[20], OpenNebula[21], Nimbus[22], Eucalyptus[23], Nirvanix [15].

Bilim bulutu alt yapı (donanımsal kaynaklar) , platform (işletim sistemi) ve yazılım ihtiyacını gerekli performans ölçütlerine göre kolayca ölçeklenebilecek bir şekilde karşılar. Basit olarak anlatmak gerekirse dağıtık yapıdaki donanım üzerinde sanal makineler çalıştırılarak bir alt yapı oluşturulur. Sanal makineler eldeki kaynaklarla sınırlı olmak üzere ihtiyaç duyulduğu kadar arttırılabilir ya da azaltılabilir [19].

1.6. Büyük Veri

2020 yılında 24 milyar bilgisayarın internet ağında olacağı ve bu bilgisayarların çoğunun bulut üzerinden gönderdikleri bilgilerle birbiriyle haberleşen cihazlar olacağı tahmin edilmektedir [24]. Bu şekilde günlük yaşamda kullanılandan daha fazla bilgi sayısallaştırılacak ve yine günlük hayatımıza faydalı olacak bir şekilde geri dönmesi için teknolojiler geliştirilecektir.

Günümüzdeki bazı veri büyüklüklerinden bahsedersek ;

- ✓ Her biri yaklaşık 300 kilobayt olan 40X10^9 web sayfası 10 petabayt büyüklüğündedir.

- ✓ LHC (büyük hadron çarpıştırıcısı) yılda 15 petabayt veri üretmektedir.
- ✓ Radyoloji alanında yılda 69 petabayt veri üretilmektedir.
- ✓ Square Kilometer Array (Kilometre Kare Dizgesi-SKA) isimli teleskobun saniyede 100 terabit veri elde etmesi beklenmektedir.
- ✓ Boing jet motorları 30 dakikalık uçuş için 10 terabayt veri üretebilmektedir.
- ✓ YouTube'a her dakika 48 saat uzunluğunda video yüklenmektedir (yılda 2.5 petabayt).

Ayrıca insanlar her an sosyal medyalarında paylaştıkları metin, fotoğraf ve videolarla yükledikleri veri miktarını arttırmaktadır. Örneğin 2014 rakamlarına göre 645 milyondan fazla kayıtlı Twitter kullanıcısı bulunmakta ve bu kullanıcılar günde ortalama 58 milyon Tweet yazmaktadır. Facebook kullanıcılarının sayısı ise 1.2 milyar civarındadır. Bu da Facebook için her gün 500 terabaytlık veri üzerinde işlem yapmak demektir.

Görüldüğü üzere bir büyük veri patlaması söz konusudur. Dijital ortamdaki veriler çok çeşitlidir ve miktarı üssel (exponansiyel) olarak artmaktadır. Bu durum verilerin analizi için yeni yöntemler ve çözümler geliştirilmesini kaçınılmaz kılmaktadır.

Bilgisayarların hesaplama güçlerinin arttırılması, süper bilgisayarların üretilmesi bilimsel problemlerin çözümü için oldukça olumlu gelişmelerdir. Çünkü veri miktarı ve hesaplama gücü ihtiyacı her geçen gün artmakta, bilim insanları da yeni problemleri çözmek istemektedir. Bilimsel alanlar dışında, kar amaçlı şirketler de yine yüksek miktarda veri analizi için yeni çözümlere başvurmaktadır.

Bu şartlar göz önüne alındığında her problem için bir süper bilgisayar tahsis etmek verimlilik açısından iyi bir yaklaşım değildir. Ayrıca her iş için belirli kaynaklar tahsis edilmesi mümkün olsa bile kaynakların belirli geçici süreler boyunca kullanılmayacağı durumlar olacağından dolayı yine verimlilik problemi söz konusu olacaktır. Maliyetleri çok yüksek olan süper bilgisayarlar yerine günümüz teknolojisinde genel kullanım amaçlı bilgisayarın (örneğin ~2.5 GHz, 2 çekirdekli veya 4 çekirdekli, 500 GB - 1 TB disk kapasitesine sahip olan) kaynaklarını paralel bir şekilde kullanmak, ağ üzerinden bir araya getirilmiş bu kaynakları tek bir noktadan

kontrol etmek ve meşgul olmayan kaynakları diğer işler için paralel bir şekilde kullanılabilmek bulut bilişimin en önemli avantajlarındandır.

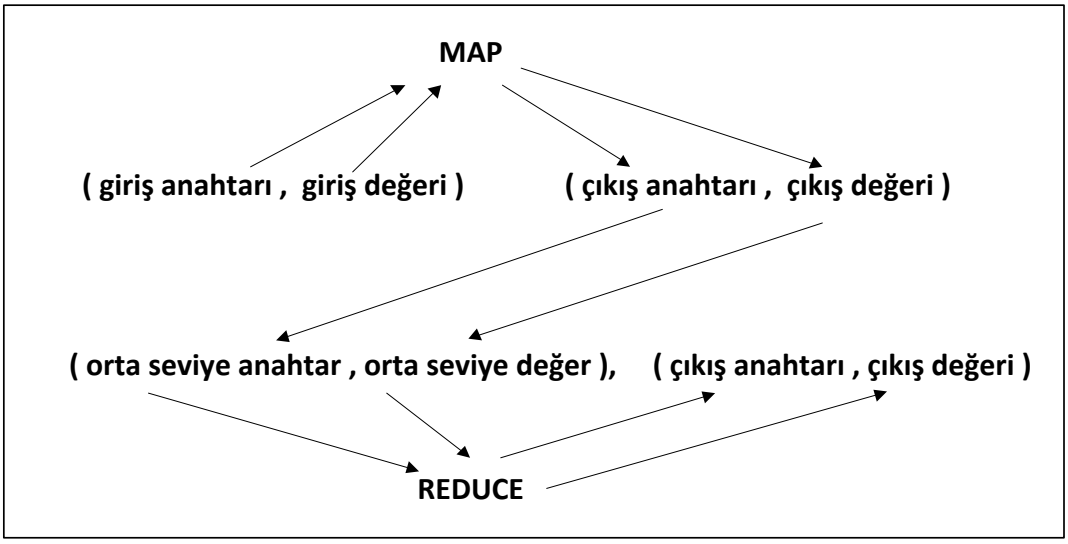
1.7. MapReduce

MapReduce Google şirketinin dağıtık bilgisayar kümeleri üzerinde büyük veriler üzerinde analiz yapabilmek için geliştirdiği programlama modelidir [25]. Bu modelde veriler üzerinde Map ve Ruduce adı verilen 2 aşamalı bir işlem gerçekleştirilir. Mapperlar dağıtık dosya sistemi üzerine bloklar halinde dağıtılmış verileri (anahtar, değer) şeklinde listeler :

Map (k1,v1) → list (k2,v2).

Ara anahtar/değer listesi adı verilen veriler oluşturulduktan sonra Reducerlar anahtar/değer ikilileri üzerinde gruplama işlemini gerçekleştirir. Bu iki temel aşama Şekil 1.4'te gösterilmiştir.

Reduce (k2, list (v2)) → list (v3).



Şekil 1.4. MapReduce adımları

MapReduce modelini bir örnekle anlatırsak, elimizde 5 yılın Amerikan Doları - Türk Lirası kurunun tutulduğu bir veri seti olsun. Bu veri setindeki satıların aşağıdaki gibi olduğunu düşünelim.

```
2012 , 1 ABD = 1.8037 TL

2012 , 1 ABD = 1.8014 TL

2011 , 1 ABD = 1.7936 TL

2010 , 1 ABD = 1.7988 TL

2010 , 1 ABD = 1.8004 TL

2009 , 1 ABD = 1.7995 TL

2008 , 1 ABD = 1.6702 TL
```

Bu veri setindeki değerler gerçek olmayıp, MapReduce program mantığını göstermek için rastgele verilmiştir. Her yıl için Amerikan Doları'nın Türk Lirası karşısında en çok değer kazandığı değerleri bulan MapReduce programının çalışma adımları şu şekilde olur :

Map fonksiyonu yıl değerini anahtar, TL değerini ise değer olarak alıp aşağıdaki gösterildiği şekilde ara anahtar/değer (yıl, TL) girişlerini oluşturur.

```
(2012, 1.8037)

(2012, 1.8014)

(2011, 1.7936)

(2010, 1.7988)

(2010, 1.8004)

(2009, 1.7995)

(2008, 1.6702) ...
```

Map fonksiyonu çıktısı reduce fonksiyonu tarafından giriş olarak alınmadan önce MapReduce sistemi (anahtar, değer) verileri üzerinde sıralama ve gruplama işlemi gerçekleştirir. Bu şekilde aşağıdaki gibi bir liste elde edilir.

```
(2012, [1.8037, 1.8014]), (2011, [1.7936]),

(2010, [1.8004,     1.7988]), (2009, [1.7995]),

(2008, [1.6702])...
```

Son aşamada reduce fonksiyonu sıralanmış ve gruplanmış Dolar-TL çiftlerinden en yüksek değerdeki ikilileri alıp çıkış değerlerini aşağıdaki gibi oluşturur.

```
(2012, 1.8037), (2011, 1.7936), (2010, 1.8004),

(2009, 1.2995), (2008, 1.6702)
```

2. MAKİNE ÖĞRENMESİ

Makine öğrenmesi; bilgisayarın var olan verilerden yola çıkarak (öğrenerek) yeni gelen veriler üzerinde tahminde bulunması veya onları farklı sınıflara ayırmasıyla ilgilenen yapay zekânın bir koludur. Makine öğrenmesi teknikleri kullanılarak sınıflandırma ve tahmin yapılabilir. Yani var olan veriler sınıflara ayrılarak yeni gelen veriler hakkında çıkartım yapılır. Bazı örnek makine öğrenmesi problemleri şunlardır; yüz tanıma, optik karakter tanıma, spam emaillerin tespiti, konu tespiti, müşteri segmentasyonu, dolandırıcılık tespiti, hava durumu tahmini [26]. Makine öğrenmesi yöntemleri temelde iki farklı gruba ayrılabilir. Bu gruplar verilerin eğitilmesi aşamasında kullanılan verinin etiketli (labelled) olup olmamasına göre danışmanlı (supervised) veya danışmansız (unsupervised) öğrenme isimlerini alır. Danışmanlı öğrenme yönteminde muhtemel sonuçların ne olduğu önceden bilinir, yeni gelen verinin bu bilinen sonuçlardan hangisine sahip olacağı hesaplanır. Danışmansız öğrenmede ise veri içerisindeki sınıfların veya veri içerisinde olan bir bilginin ortaya çıkarılması söz konusudur.

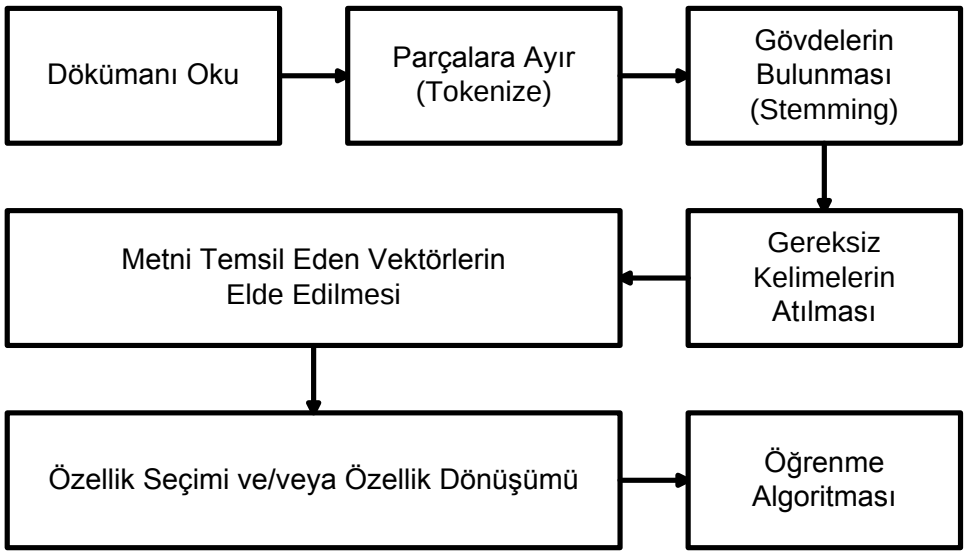
Bu tez çalışmasında büyük verilerin dağıtık makine öğrenmesi algoritmalarıyla analiz edilmesi amaçlanmıştır. Bu bölümde literatürdeki makine öğrenmesi algoritmaları incelenerek algoritmaların genel özelliklerine değinilmekle birlikte daha çok metin tabanlı veriler üzerinde kullanıldığı senaryolar göz önünde bulundurulmuştur. Makine öğrenmesi tekniklerinin dağıtık bir şekilde doküman sınıflandırma, log analizi vb. konularında uygulanması için gerekli bilgiler özetlenmiştir.

2.1.Metin Sınıflandırma Adımları

Dijital dokümanların sayılarının gün geçtikçe çok hızlı bir şekilde artmasıyla birlikte bu dokümanların otomatik olarak önceden belirlenmiş sınıflara ayrılması son yıllarda oldukça önem kazanan bir konu haline gelmiştir [27]. Doküman sınıflandırması yöntemleri kullanılarak dokümanlar kategorilere, başlıklara ayrılabilir, spam (önemsiz) olup olmadığı, hangi dilde yazıldığı, hangi duygu durumunu gösterdiği tespit edilebilir.

Metin sınıflandırması; d_i tüm dokümanları temsil eden D'ye ait bir dokuman ve $\{s_1, s_2, \dots, s_n\}$ kategoriler olmak üzere d_i'nin ait olduğu s_j kategorisinin bulunması

işlemidir [28]. Dokümanların türlerine (ekonomi, spor, siyaset vb.) göre sınıflandırılmasında internet aracılığıyla dergi, gazete, bülten tahtaları (bulletin boards) gibi farklı yerlerden doküman elde edilebilir. Dokümanların toplanması (information retrieval) sınıflandırmanın ilk basamağıdır.



Şekil 2.1. Makine öğrenmesiyle metin sınıflandırma aşamaları[28]

Dokümanlar toplandıktan sonra öğrenme algoritmasının uygulamasına kadar bazı işlemlerden geçmesi gerekir. İkonomakis ve arkadaşları bu aşamaları çalışmalarında çok iyi bir şekilde göstermiştir [28]. Şekil 2.1'de gösterildiği gibi toplanan dokümanlara uygulanacak ilk işlem okunmalarıdır. Algoritmaların paralel olarak yürütüleceği durumda bu aşama öncesinde dağıtık dosya sistemi üzerine verilerin yazılması ve kaynaklar arasında paylaştırılması gerekir.

Veriler okunduktan sonra parçalara ayrılırlar. Bu parçalar bir yazıyı oluşturan yapısal unsurlardır. Örneğin cümleler veya kelimelere bölünmesi işlemi bu adımda yapılır. Bu aşamadan sonra kelime gövdeleri bulunur. Örnek olarak "Brezilya, Almanya'ya 7-1'lik tarihi bir skorla mağlup olarak ev sahipliği yaptığı Dünya Kupası'na yarı finalde veda etti." cümlesi bu aşamadan sonra "Brezilya, Almanya 7-1 tarih bir skor mağlup olmak ev sahip yapmak dünya kupa yarı final veda etmek" olarak değiştirilir. Bunun amacı dokümanların temsil edildiği kelime uzayında aynı kelimenin farklı ek ve çekimlerden dolayı farklı kelimeler olarak hesaplanmasının istenilmemesidir.

Dokümanların kelime sayısı zaten oldukça fazladır ve bu sayıya bir kelimenin birden fazla sayıda eklenilmesi problemi daha da karmaşıklaştırmaktadır.

Bu aşamadan sonra metinlerdeki "ve", "ile", "de", "da" gibi tek başına bir anlamı olmayan kelimeler atılır. Gövdelerin bulunması ve gereksiz kelimelerin bulunması aşamalarının otomatik bir şekilde yapılırken işlem yapılan dil için bir **doğal dil işleme kütüphanesi** gereklidir. Bu kütüphane kullanılarak kelime kök ve gövdeleri, edatlar, bağlaçlar gibi dil bilgisi gerektiren işlemler otomatik bir şekilde yapılır. Bu tez çalışmasında Zemberek [29] isimli doğal dil işleme kütüphanesi kullanılmıştır.

Dokümanlar bu ön işlemlerden sonra hesaplama ortamına aktarılmaya hazır hale gelmiş olur. Buradaki en önemli soru kelimelerden oluşan bir metnin sayısal dünyada neye karşılık geleceğidir. Dokümanlar kelimelerden oluşan birer dizi olarak düşünülebilir. Metin dokümanları genellikle bir sözlükteki kelimelerin bulunması durumuna ağırlıklandırılmış vektörler olarak temsil edilirler. Buna **vektör uzayı modeli** denir. Bu gösterimde kelimelerin bulunduğu konumlar yani hangi sırada geçtikleri önemsizdir [30]. Kelime vektörlerinin ağırlıklandırılmasında farklı yöntemler kullanılmaktadır. Örneğin kelimelerin sadece dokümanda bulunup bulunmamasına göre 1 veya 0 değerini alabileceği gibi dokümanda kaç kez geçtiği yani frekansı da vektörün ağırlık değerine katılabilir. Bu gösterim ile (doküman sayısı * farklı kelime sayısı) boyutlu bir matris elde edilir. En yaygın kullanılan ağırlıklandırma yöntemlerinden birisi **TF-IDF** (term frequency - inverse document frequency)'dir. Bu yöntem şu şekilde formülüze edilir [31]:

$$M_{ij} = tf_{ij} \tag{1}$$

$$M_{ij} = \log(tf_{ij} + 0.5) * \log(\frac{\mathrm{D}}{df_i}) \tag{2}$$

Burada tf_{ij}, i kelimesinin j dokümanında geçme sayısını, D toplam doküman sayısını, df_i, i kelimesi içeren dokuman sayını göstermektedir. Görüldüğü üzere bir dokumanın **tf-idf** ağırlığı hesaplanırken kelimenin o dokumanda geçme sıklığıyla doğru orantılı bir şekilde önemi artmakta fakat diğer dokümanlarda da sık geçmesiyle birlikte azalmaktadır. Bu durum şöyle açıklanabilir; eğer bir kelime tüm dokümanlarda çok

fazla geçiyorsa bu o kelimenin belirleyici olmaması gerektiğini fakat sadece bazı dokümanlarda fazla geçiyorsa o dokümanların ait olduğu sınıf için belirleyici kelimeler arasında olması gerektiğini gösterir.

Tf-idf yöntemi ve verilerin istatiksel olarak gösterildiği diğer yöntemlerde, vektörlerin nasıl oluşturulduğunu göstermek amacıyla aşağıdaki cümleler örnek olarak kullanılacaktır.

1. Yangında tekneler zarar gördü.
2. Tekneler kıyasıya yarıştı.
3. Yarışmacılar zarar görmedi.

Her cümlenin farklı bir dokümanın içeriği olduğu düşünülürse, toplam üç adet doküman (d_1, d_2, d_3) ve altı farklı kelime ("yangın", "tekne", "zarar", "görmek", "kıya", "yarışmak") bulunmaktadır. Bu 3x6 boyutunda bir matris elde edileceğini gösterir.

	Yangın	**Tekne**	**Zarar**	**Görmek**	**Kıya**	**Yarışmak**
$\boldsymbol{d_1}$	1	1	1	1	0	0
$\boldsymbol{d_2}$	0	1	0	0	1	1
$\boldsymbol{d_3}$	0	0	1	1	0	1

$d_1 = \{1,1,1,1,0,0\}$

$d_2 = \{0,1,0,0,1,1\}$

$d_3 = \{0,0,1,1,0,1\}$

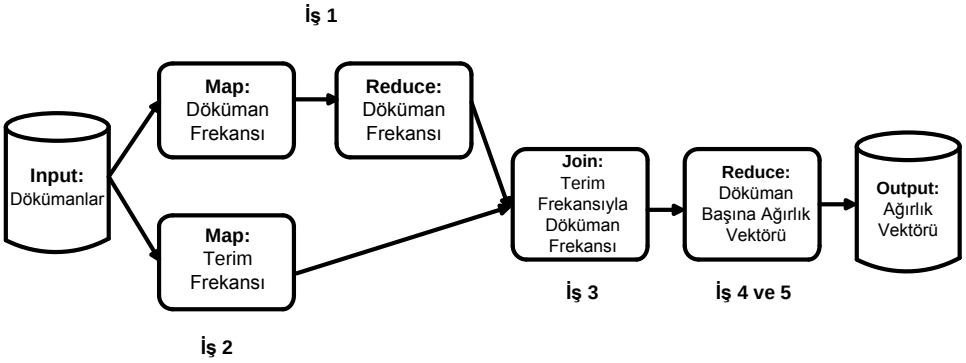
d_1 dokümanında tekne (ikinci kelime) kelimesinin tf-idf ağırlığı (M_{12}) şu şekilde hesaplanır:

D=3, df_i = 2

$$M_{12} = tf_{12} = 1$$

$$M_{ij} = \log(tf_{ij} + 0.5) * \log(\frac{\mathrm{D}}{df_i}) = \log(1 + 0.5) + \log(\frac{3}{2}) = 0{,}26$$

Elde edilen tf-idf değerleri frekans değerlerinin yerine yazılarak dokümanları temsil eden d_1, d_2, d_3 vektörleri elde edilir. Bu şekilde sözel bilgilerden oluşan dokümanlar sayılar karşılıklarıyla makine öğrenme algoritmasında hesaplanmaya hazır hale getirilmiş olur.



Şekil 2.2. Dağıtık Tf-idf akış şeması

Tf-idf değerlerininin bulunması dağıtık bir şekilde yapılmak istenildiğinde izlenilmesi gereken yol Şekil 2.2'de gösterilmiştir. Görüldüğü üzere tek adımda bu hesabın yapılması mümkün olmamakta ve beş tane işten oluşmaktadır. Bu işlerden birinci ve ikincisinin paralel yürütülmesi uygundur.

2.2.Makine Öğrenmesi Algoritmaları

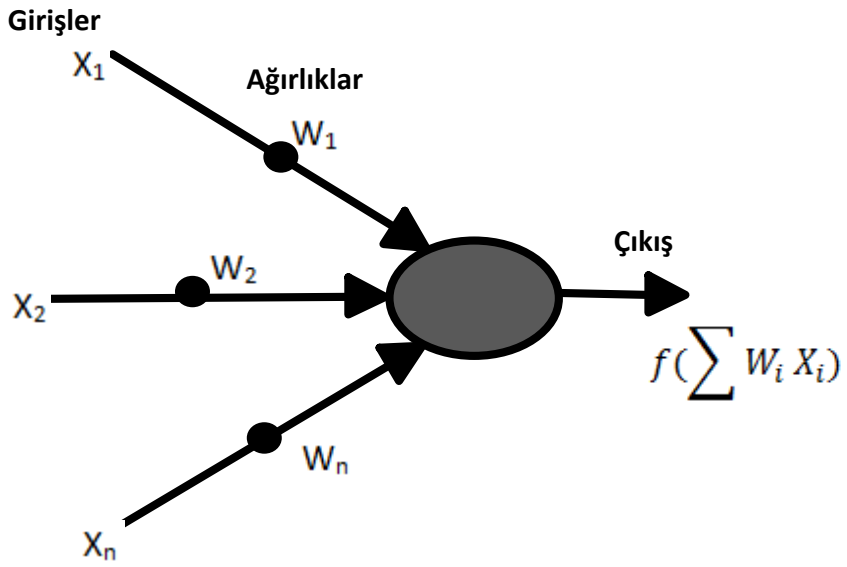
Literatürde metin tabanlı dokümanların sınıflandırması konusunda birçok yöntemin önerildiği görülmektedir. Bu tez çalışmasında başarı oranı yüksek olan bazı popüler yöntemler incelenmiştir. Bunlar yapay sinir ağları, destek vektör makinaları (DVM), k en Yakın Komşu (kNN) yaklaşımı, naif Bayes sınıflandırıcı ve karar ağaçlarıdır.

a. Yapay Sinir Ağları

Yapay sinir ağı, yapı olarak sinir hücresine benzeyen f aktivasyon fonksiyonuna bağlı giriş (x), girişlerin ağırlıkları (w) ve çıkış (y) arasındaki bağlantıdan oluşan öğrenme yöntemidir. Sinir ağları genellikle çok katmanlıdır ve her katmanın farklı bir fonksiyonu vardır. Metin sınıflandırmada yapay sinir ağlarında, genellikle gizli bir katman ve çıkış katmanı bulunur. Aktivasyon fonksiyonu (f) lineer, hiperbolik tanjant

veya sigmoid olabilir. Girişler genellikle katman olarak düşünülmez ve nöronlar arasındaki bağlantılar ağırlıklandırılarak belirli elementler arasındaki bağlılık gücü bulunur [30].

$$y = f(\sum_{i=1}^{N} w_i + x_i)$$



Şekil 2.3. Yapay sinir ağı

Yapay sinir ağlarıyla metin sınıflandırma, çok fazla özellik olan durumlarda iyi sonuç verebilmesi ve gürültü olan durumlarda doğru sınıflandırma yapabilmesi açısından kullanışlıdır [30]. Fakat hesaplama maliyeti (computational cost) yüksek olduğu için de dezavantajlıdır.

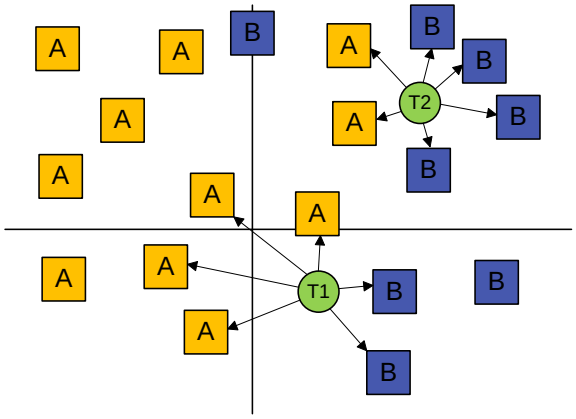
Güven ve arkadaşları [32] yapay sinir ağı yöntemini kullanarak "spor", "ekonomi" ve "kültür" konu başlıklarında göre web sayfalarını sınıflandırmıştır. Önerdikleri yöntemde 60 adet giriş ve 3 adet çıkış bulunmaktadır. Ayrıca giriş ve çıkış arasında 20 nörondan oluşan gizli bir katman dâhil edilmiştir. Transfer fonksiyonu olarak sigmoid kullanılmıştır. Girişler, kelimenin o dokümanda geçme frekansının tüm kelimelerin toplam tekrarlanma sayısına bölümünden ortaya çıkan değerle eğitilmiştir. Önerilen yöntem %99 oranında başarılı sınıflandırma yapabilmektedir.

b. k-En Yakın Komşu Algoritması

Bu yöntemde de giriş verisi vektör uzay modeliyle temsil edilen dokümanlardır. Daha önce bahsedilen tf-idf ile ağırlıklandırılmış vektör modeli bu algoritmada en çok kullanılan temsil yöntemidir. Bir dokümanının hangi sınıfa ait olduğu hesaplanırken yakınındaki k-adet dokümana olan uzaklığı hesaplanır ve hangi dokümanın bulunduğu sınıfa daha yakınsa o sınıfa ait olduğu bulunur.

İlk adım olan eğitim verilerinin benzerliklerinin (yakınlıklarının hesaplanması) r-koordinat uzayında düşünülerek kosinüs benzerliği ile yapılır. Vektörlerin özellik sayısı burada r'yi belirler. Al-Shalabi ve arkadaşları çalışmalarında [33] sırasıyla j ve k dokümanlarını temsil eden A ve B vektörlerinin benzerliklerinin aşağıdaki formülle hesaplandığını göstermiştir.

$$\text{sim}(A, B) = \frac{\sum_{i=1}^{r} w_{ij} * w_{ik}}{\sqrt{\sum_{i=1}^{r} w_{ij}^2} * \sqrt{\sum_{i=1}^{r} w_{ik}^2}}$$

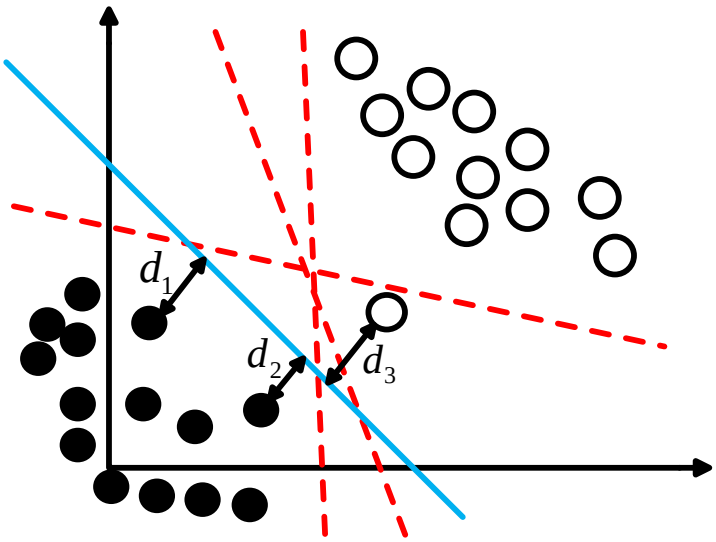


Şekil 2.4. k-NN algoritmasının çalışma şekli [30]

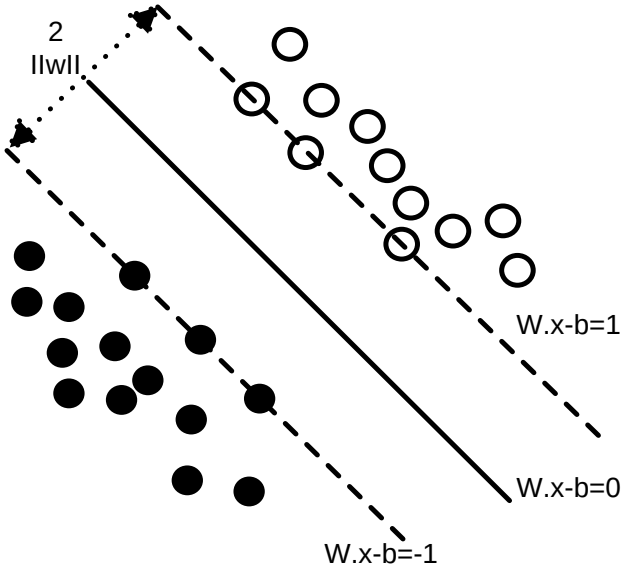
Şekil 2.4 k-NN algoritmasının çalışma şeklini göstermektedir [30]. A harfi ile gösterilen sınıfa dâhil olan dokümanlar ve B harfi ile gösterilen sınıfa dâhil olan görülmektedir. T1 ve T2'nin hangi sınıfa ait olacağı k tane en yakın komşusuyla benzerliklerinin hesaplanmasıyla bulunur.

c. Destek Vektör Makinaları (DVM)

İlk olarak Vapnik'in [34] önermiş olduğu bu yöntem iki farklı sınıfa ayırma temeline dayanır. Amaç verileri bir hiper düzlemin böldüğü, birbirinden mümkün olduğunca uzak iki farklı sınıfa yerleştirmektir. DVM'ler hızlı çalışması ve yüksek boyutlu giriş vektörleriyle iyi sonuçlar verebilmesi özelliğinden dolayı metin sınıflandırılması konusunda popüler bir yöntemdir [35].



Şekil 2.5. DVM karar yüzeyi



Şekil 2.6. DVM hiper düzlemi

Şekilde 2.5'te DVM karar yüzeyi gösterilmiştir [30]. Noktalar her bir dokümanın hiper düzleme göre olabileceği en uzak noktada olacak şekilde eğitilir. Şekil 2.6'da ise sınıfların hiper düzlemle birbirinden ayrılması işlemi gösterilmiştir.

d. Naif Bayes Sınıflandırıcı

Naif Bayes, basit bir istatiksel algoritma yöntemidir ve Bayes teoremine dayanır. Bir d dokümanının s sınıfına ait olma olasılığı P şu şekilde gösterilir:

$$P(s \mid d) = \frac{P(d \mid s)\, P(s)}{P(d)}$$

Naif Bayes sınıflandırıcının önemli avantajlarından birisi az miktarda eğitim verisi olduğu durumlarda bile iyi sonuç verebilmesidir. Bu yöntem Bölüm 3.4'te uygulama ile verilmiştir.

3. UYGULAMALAR

Makine öğrenmesi algoritmalarının büyük miktardaki veriler üzerinde uygulanmasındaki en önemli faktörlerden birisi performans kısıtlarıdır. Süper bilgisayar adı verilen yüksek hesaplama kapasitesine sahip bilgisayarlar ile bu engel aşılabilmektedir fakat her kişinin her problem için bu bilgisayarlara erişimi kolay ve ekonomik olmamaktadır. Bulut bilişim ve yeni dağıtık veri işleme teknolojileri ile standart bilgisayarların paralel olarak çalışacak şekilde kullanıma sunularak hızlı, esnek, ölçeklenebilir ve aynı zamanda güvenli bir ortam ortaya çıkmıştır.

Makine öğrenmesi algoritmaları ile büyük miktarda veri üzerinde analiz yapılırken problemlerin birçoğu için ortak ve aynı zamanda paraleleştirilebilir bazı süreçler vardır. Tezin bu bölümünde öğrenme algoritmaları öncesi yapılan bazı işlerin Hadoop, Spark gibi güncel teknolojilerle nasıl yapıldığı, ne kadar başarılı olduğu, birbirine göre ve paralel olmayan yöntemlere göre üstün yönleri açıklanmıştır.

3.1.Uygulama Ortamı

Bilgisayar kümelerinin ve dağıtık hesaplama altyapılarının kurulması aşamaları önemli bir süreç olduğu için tez çalışmasında kullanılan yazılım ve donanım platformları ve bazı kurulum bilgileriyle ilgili deneyimler paylaşılmıştır. Çalışmanın tamamı bulut sistemi üzerinde yapıldığı için öncelikle bulut sistemi kurulumu ve kullanımı hakkında bilgiler verilmiştir. Devamında ise Hadoop ve Spark kümelerinin oluşturulmasıyla ilgili bilgiler verilmiştir. Bölüm 3.2. Paralel log analizinde, Bölüm 3.3. Dağıtık doğal dil işleme uygulamasında ve Bölüm 3.4. Naive Bayes ile paralel doküman sınıflandırma çalışmasında yapılan testlerde burada anlatılan altyapı kullanılmıştır ve uygulamaların anlatıldığın bölümde ayrıca platformlar hakkında bilgi verilmemiştir.

3.1.1.FutureGrid

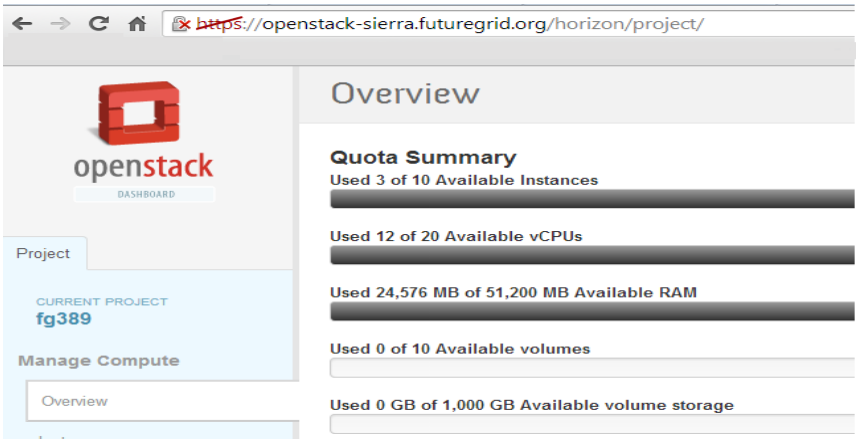
Bu çalışmada OpenStack servisi kullanılmış olan FutureGrid, araştırmacıların karmaşık kurulum, kullanım ve güvenlik detaylarından soyutlanarak grid ve bulut

ortamından yararlanmasını sağlayan Amerika Birleşik Devletleri'nde Indiana Üniversitesinde gerçekleştirilen ve halen devam eden bir projedir [36]. Kapsamlı bir bulut sisteminde halledilmesi gereken birçok konu bulunmaktadır. Bunlardan bazıları; erişim, yetkilendirme, planlama, ara katman yazılımı tasarımı, arayüz tasarımı ve siber güvenliktir. FutureGrid ile astronomi, kimya, biyoloji, mühendislik, epidemiyoloji gibi farklı alanlarda çalışan bilim insanlarının kullanımına uygun bir hesaplama ortamı sunulmuştur.

Tablo 3.1. FutureGrid donanım kaynakları [36]

System type	# CPUs	# Cores	TFLOPS	RAM (GB)	Storage (TB)	Site
IBM iDataPlex	256	1024	11	3072	335	IU
Dell PowerEdge	192	1152	12	1152	15	TACC
IBM iDataPlex	168	672	7	2016	120	UC
IBM iDataPlex	168	672	7	2688	72	UCSD
Cray XT5m	168	672	6	1344	335	IU
Shared mem. system	40	480	4	640	335	IU
IBM iDataPlex	64	256	2	768	5	UF
Total	1337	5600	58	10560	552	

FutureGrid coğrafi olarak farklı yerlere dağılmış heterojen bilgisayar sistemi kaynaklarından oluşmaktadır. Tablo 3.1'de FutureGrid'i oluşturan donanım kaynakları gösterilmiştir. Sahip olduğu veri tabanı yönetim sistemi ile yardımcı verileri (metadata) ve bulut bilişim için gerekli olan yazılım imajlarını saklamaktadır. Test ortamı (test-bed) sanal makine tabanlı bir ortam sağladığı gibi, sanallaştırılmamış standart işletim sistemi de sağlamaktadır.

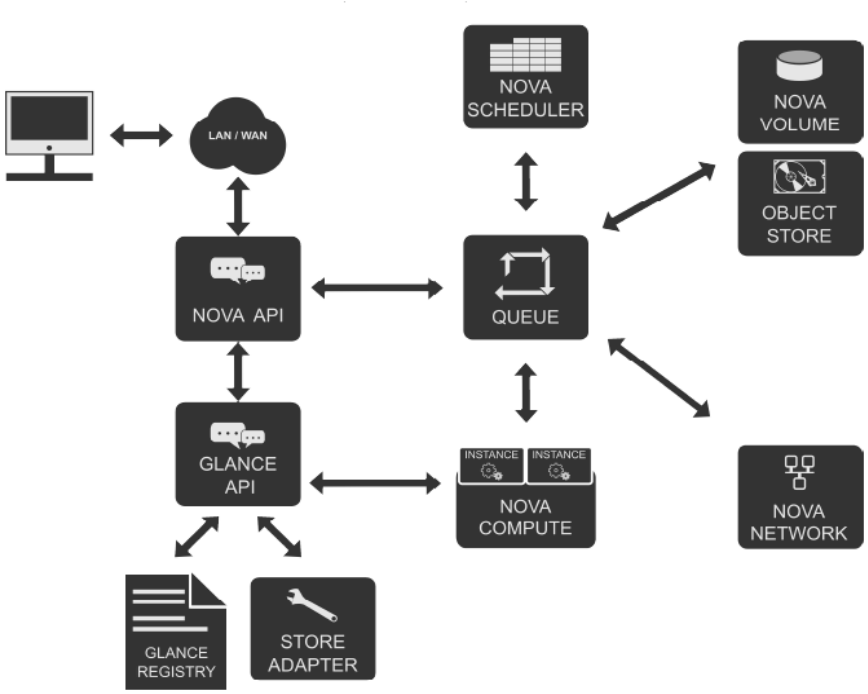


Şekil 3.1. OpenStack Sierra kullanıcı arayüzü

Bu çalışmada FutureGrid'in sadece OpenStack servisi kullanılmıştır. Şekil 3.1 tez çalışmasında kullanılan platformun web arayüzü ekran görüntüsüdür. Test ortamı olarak FutureGrid'in kullanılması oldukça elverişli olmuştur fakat bulut yazılımı olan OpenStack'ın daha kapsamlı incelenmesi için ayrıca kurulum yapılıp üzerinde testler gerçekleştirilmiştir.

3.1.2.OpenStack ve Kurulum adımları

Uygulamalarda açık kaynaklı bulut yazılım ve yönetim platformu olan OpenStack kullanılmıştır. OpenStack bir IaaS (Servis olarak altyapı)'dir. Rackspace Cloud ve NASA tarafından geliştirilmiş olan ve daha sonra bir IBM, Intel, Yahoo gibi şirketlerce de desteklenmiş açık kaynaklı bulut altyapısı projesidir. Bu çalışmada bulut ortamı olarak hem FutureGrid kaynakları hem de kendi kurulumunu yaptığımız OpenStack kullanılmıştır.



Şekil 3.2. OpenStack yazılım mimarisi [37]

OpenStack kurulumuyla birlikte 5 farklı servis sağlanmış olunur:

a) Nova – Hesaplama Servisi

Nova, OpenStack bulut sisteminin temelini oluşturur. Bulut sistemindeki kaynaklar, ağ yapıları, yetkilendirmeler ve ölçeklenebilirlikle ilgili tüm işler Nova tarafından yönetilir. Nova'nın kendine ait bir sanallaştırma yapısı bulunmamaktadır fakat libvirt API'sini kullanarak desteklediği misafir sistem ara katmanlarıyla (hypervisor) etkileşebilir. Nova, sahip olduğu tüm yeteneklerini Amazon Web Servisleri API'si olan EC2 ile uyumlu olan bir Web Servisi API'siyle de sunabilmektedir.

Nova işlev ve özellikleri:

- ✓ Makine (instance) yaşam döngüsü
- ✓ Bilgisayar kaynaklarının yönetimi
- ✓ Ağ yönetimi ve yetkilendirme
- ✓ REST tabanlı API
- ✓ Nihai tutarlı (asynchronous eventually consistent) iletişim
- ✓ Misafir sistem ara katmanı (hypervisor) agnostik : Xen, XenServer/XCP, KVM, UML, VMware vSphere and Hyper-V desteği.

b) Swift – Depolama Servisi

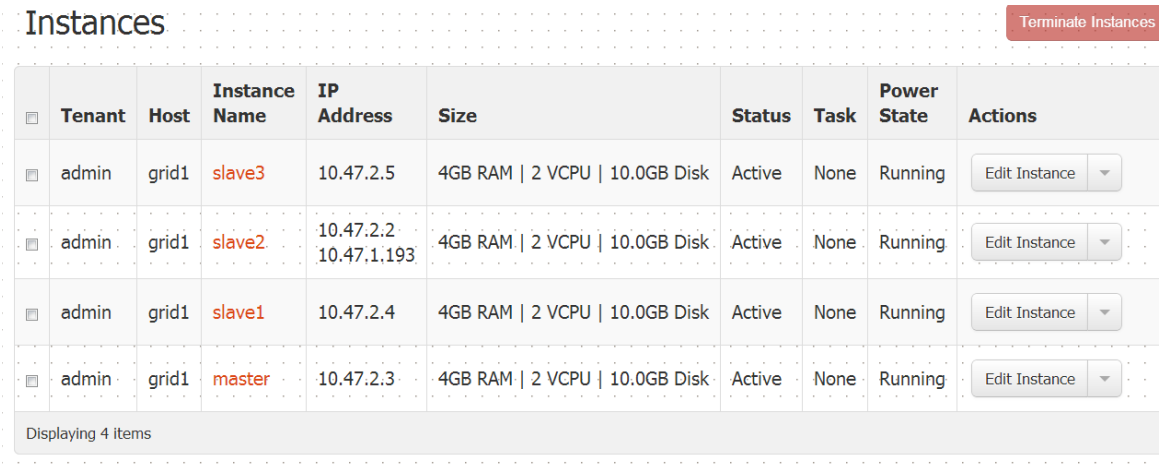
Swift, OpenStack'ın dağıtık ve nihai tutarlı sanal nesne depolama birimidir. Amazon Web Services - Simple Storage Service'e benzer. Swift, milyarlarca nesnenin dağıtık sistem üzerindeki bilgisayarlarda saklanmasını sağlayabilmektedir. Büyüklük (birkaç petabayt) ve sayı (nesne sayısı) bakımından oldukça ölçeklenebilir bir yapısı vardır.

Swift İşlev ve Özellikleri:

- ✓ Büyük boyutlu nesnelerin depolanması
- ✓ Veri fazlalığı
- ✓ Arşivsel yetenekler ve büyük veri setleriyle çalışma imkanı
- ✓ Sanal makineler ve bulut uygulamaları için veri saklama birimi
- ✓ Veri akışı ortamı yetenekleri
- ✓ Nesnelerin güvenli bir şekilde saklanması
- ✓ Yedekleme ve arşivleme
- ✓ Yüksek ölçeklenebilirlik.

c) Glance – İmaj Servisi

Glance, OpenStack'ın sanal makine imajı alma, düzenleme, silme ve yönetilmesiyle ilgili yetenekleri gerçekleştiren birimidir.



Şekil 3.3. Glance

Şekil 3.3'te kurulan sistemin Glance ekranı verilmiştir. Glance aşağıdaki depolama sunucuları kullanılacak şekilde konfigüre edilebilir:

- ✓ Yerel dosya sistemi (varsayılan)
- ✓ OpenStack - imaj saklamak için nesne deposu

- ✓ Direkt S3 depolama
- ✓ Nesne deposuyla birlikte ara erişim kaynağı
- ✓ HTTP (salt-okunur)

d) Keystone – Kimlik Servisi

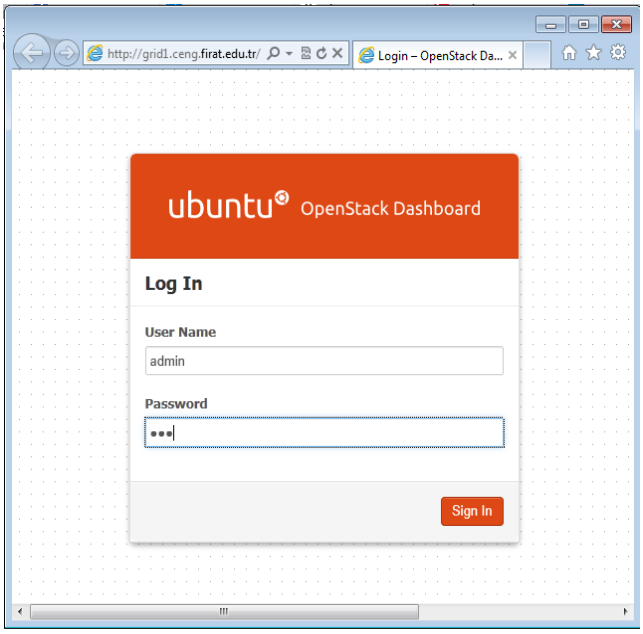
Keystone, OpenStack servislerine erişimi sağlayan birimidir. Swift, Glance, Nova bileşenlerinin de dâhil olduğu OpenStack bileşenlerine erişim yapılırken gerekli kimlik denetimi ve yetkilendirmeden sorumludur.

Keystone iki türlü erişim yöntemi kullanmaktadır. Bunların birisi kullanıcı adı ve şifre ile bağlanmaktır. Diğeri ise jeton tabanlı erişimdir.

e) Horizon – Kullanıcı Ara Yüzü Servisi

OpenStack'ın yönetici web arayüzüdür. OpenStack servislerinin, kullanıcı adı ve şifreyle girilen bir web arayüzüyle yönetilmesini sağlar. Bu işlemler özetle aşağıda sıralanmıştır:

- **Makine yönetimi:** Sanal makinelerin oluşturulması, sonlandırılması, konsol loglarının görüntülenmesi gibi işlemlerdir.
- **Erişim ve güvenlik yönetimi:** Güvenlik grupları oluşturma, keypair oluşturma, makinelere IP adresi atama gibi işlemlerdir.
- **Flavor yönetimi:** Oluşturulabilecek makinelerin donanımsal özellikleri açısından alternatifleri anlamına gelen flavor'ların ve taslakların yönetilmesiyle ilgili işlemlerdir.
- **İmaj yönetimi:** İmajların oluşturulması ve silinmesi işlemleridir.
- **Kullanıcı yönetimi:** Kullanıcıların oluşturulması, yönetilmesi ve projelere erişim yetkilerinin ayarlanmasıyla ilgili işlemlerdir.
- **Volume yönetimi:** Volume ve snapshot oluşturması işlemleridir.



Şekil 3.4. OpenStack Horizon arayüzü

Fırat Üniversitesinde kurulan bulut sisteminin horizon arayüzü Şekil 3.4'te gösterilmiştir.

Bu bölümde verilen kodlar kendi bulut sistemimiz üzerinde çalıştırılmıştır. İşlem adımları [29]'dan takip edilmiştir.

Kurulum için sistem gereksinimleri minimum 8GB RAM, 4 işlemci çekirdeği, (2) hard disk, (1-2) Ethernet kartı olmasıdır. Kurulum yaptığımız sistemin özellikleri şöyledir:

16 GB RAM, 2 x Quad-Core Intel Xeon E5440 işlemci, 440 GB hard disk ve 2 Ethernet kartı.

OpenStack kurulumu için en az 2 disk gerektiğinden disk bölme yoluyla yeni bir mantıksal birim elde edilmiştir. Bu disk bölümünün sistem tarafından ikinci bir disk olarak algılanması için işletim sistemi kurulumu sırasında bir bölüm ayrılır ve bu bölüm kurulumdan sonra "partprobe /dev/sda" komutu ile yeni bir bölüm olarak gösterilir.

```
ibrahim@grid1:~/openstackgeek/essex$ sudo fdisk -l
```

[sudo] password for ibrahim:

Disk /dev/sda: 440.1 GB, 440076861440 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	2048	499711	248832	83	Linux
/dev/sda2		501758	859523071	429510657	5	Extended
/dev/sda3		499712	501757	1023	83	Linux
/dev/sda5		501760	859523071	429510656	8e	Linux LVM

a) Kurulum kodlarının indirilmesi

Github'dan kurulum kodları indirilir.

```
git clone git://github.com/StackGeek/openstackgeek.git
cd openstackgeek/essex
```

b) Ana kurulum dosyalarının çalıştırılması

Kurulumun ilk adımında ./openstack_base_1.sh çalıştırılır.

```
./openstack_base_1.sh
```

./openstack_base_1.sh'ın çalışması bittikten sonra kurulum yapılan sunucunun ağ konfigürasyonu yapılır. Aşağıdaki komut ağ ayarlarının düzenleneceği dosyayı düzenleme modunda açar.

```
sudo nano /etc/network/interfaces
```

Bizim ağ ayarlarımız şu şekilde :

The loopback network interface

auto lo

iface lo inet loopback

The primary network interface

auto eth0

```
iface eth0 inet static
```

```
address 10.47.1.201
netmask 255.255.0.0
network 10.47.1.0
broadcast 10.47.1.255
gateway 10.47.1.1
```

Ağ ayarları değiştirildikten sonra yeni ayarların etkin olabilmesi için ağ adaptörü yeniden başlatılır. Bundan sonra 2. kod olan ./openstack_base_2.sh çalıştırılır.

```
/etc/init.d/networking restart
./openstack_base_2.sh
```

İşletim sistemi kurulumundan önce 2. disk olarak kullanmak üzere ayırdığımız disk alanı kullanılarak yeni bir mantıksal birim oluşturulur. Buradaki cihaz isimlerinin değişebileceği unutulmamalıdır.

```
openstackgeek# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xb39fe7af.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-62914559, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-62914559, default
62914559):
```

```
Using default value 62914559
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
openstackgeek# pvcreate -ff /dev/sda1
Physical volume "/dev/sdb1" successfully created
openstackgeek# vgcreate nova-volumes /dev/sda1
Volume group "nova-volumes" successfully created
```

c) OpenStack İçin MySql Kurulumu

İndirilen kurulum dosyaları içerisinde MySql kurulumunu başlatacak çalıştırılabilir bir dosya olan openstack_mysql.sh bulunmaktadır. Bu dosyanın çalıştırılması ve kullanıcı adı, şifre gibi sorulara verilen girişler aşağıdaki gibidir :

```
./openstack_mysql.sh
Enter a password to be used for the OpenStack services to talk to
MySQL (users nova, glance, keystone): 123
mysql start/running, process 8796
#
Creating OpenStack databases and users.  Use your database password
when prompted.
Run './openstack_keystone.sh' when the script exits.
#
Enter password:
```

MySql kurulumundan sonra aşağıda gösterildiği şekilde OpenStack ve root kullanıcıları veritabanına bağlanabiliyor olmalıdır.

```
mysql -u root -123
mysql -u nova -123 nova
mysql -u keystone -123 keystone
mysql -u glance -123 glance
```

d) KeyStone Kurulumu

Keystone'nun kurulum dosyası çalıştırılır.

```
./openstack_keystone.sh
```

Mysql kullanıcıları için kurulum sırasında verilmiş olan şifre (biz "123" vermiştik) kullanılır.

```
# Enter a token for the OpenStack services to auth wth keystone:
r4th3rb3t0k3n
# Enter the password you used for the MySQL users (nova, glance,
keystone): 123
# Enter the email address for service accounts (nova, glance,
keystone): user@foobar.com
```

Keystone kullanıcı listesinin görüntülenebilmesi için önce stackrc çalıştırılmalıdır.

```
. ./stackrc
   keystone user-list
```

```
ibrahim@grid1:~/openstackgeek$ cd essex/
ibrahim @grid1:~/openstackgeek/essex$ . ./stackrc
ibrahim @grid1:~/openstackgeek/essex$ keystone user-list
```

```
+----------------------------------------+---------+--------------------+--------+
|                id                      | enabled |       email        |  name  |
+----------------------------------------+---------+--------------------+--------+
| 3d6f404efcfe4a3faf9c63bded566f48       |  True   | user@foobar.com    |  nova  |
| 9dc0da8612fe4e11af69dcb7a3219662       |  True   | user@foobar.com    | glance |
| cb3ac9ca40c047e78fa8a2ab4c2aa290       |  True   | user@foobar.com    | admin  |
| dd4417159c5746c5a81ac03dbbb3180e       |  True   | user@foobar.com    |  demo  |
+----------------------------------+---------+----------------+--------+
```

e) Glance Kurulumu

Glance'nin kurulum dosyası çalıştırılır.

```
./openstack_glance.sh
```

Bu dosya çalıştırıldığında bilgisayara StackGeek'in S3 kaynağından Ubuntu 12.04 LTS bulut imajını indirecektir.

İşlem sonlandığında aşağıdaki komut çalıştırılarak sistemde 1 tane imaj dosyası olduğu görülebilir.

```
glance index
+----------------------------------+---------+----------------+--------+
|                id                | enabled |      email     |  name  |
+----------------------------------+---------+----------------+--------+
| 3d6f404efcfe4a3faf9c63bded566f48 | True    | user@foobar.com | nova   |
| 9dc0da8612fe4e11af69dcb7a3219662 | True    | user@foobar.com | glance |
| cb3ac9ca40c047e78fa8a2ab4c2aa290 | True    | user@foobar.com | admin  |
| dd4417159c5746c5a81ac03dbbb3180e | True    | user@foobar.com | demo   |
+----------------------------------+---------+----------------+--------+
```

Bu yöntemin dışında da bulut imajları oluşturulup glance imaj listesine eklenebilir. Bu çalışma için Ubuntu 12.04 LTS uygun bir işletim sistemi olduğu için biz farklı bir imaj yükleme işlemi gerçekleştirmedik.

f) Nova Kurulumu

Bu aşamada OpenStack bulut platformunun tüm bilgisayar ve ağ yönetimini gerçekleştiren, imajların oluşturulması, görüntülerinin alınması ve silinmesi gibi tüm işlemlerden sorumlu olan yazılımı Nova'nın kurulumu gösterilecektir. Diğer bileşenlerde olduğu gibi Nova kurulumunda da ilk olarak ilgili çalıştırılabilir dosya yürütülür.

```
./openstack_nova.sh
```

Bu dosya kurulumu gerçekleştirirken ağ konfigürasyonunu gerçekleştirebilmesi için ağ ortamımızla ilgili bilgileri girmemizi isteyecektir. Bizim sistemimiz "10.47.1.0" ağında bulunmaktadır ve konfigürasyonu şu şekilde yapılmıştır.

```
###
The IP address for eth0 is probably 10.0.1.35. Keep in mind you
need an eth1 for this to work.
###
Enter the primary ethernet interface IP: 10.47.1.201
Enter the fixed network (eg. 10.0.2.32/27): 10.47.2.1/27
Enter the fixed starting IP (eg. 10.0.2.33): 10.47.2.2
###
The floating range can be a subset of your current network.
Configure your DHCP server
to block out the range before you choose it here.  An example would
be 10.0.1.224-255
###
Enter the floating network (eg. 10.0.1.224/27): 10.47.1.221/27
Enter the floating netowrk size (eg. 32): 32
```

Bu ayarlara göre başlatacağımız sanal makineler "10.47.2.2", "10.47.2.3" … şeklinde ip adresi alacaktır.

Nova kurulumu bittikten sonra artık Nova üzerinden Glance'a imaj listesi sorgulanabilir:

ibrahim@grid1:**~/openstackgeek/essex$ nova image-list**

```
+--------------------------------------------+----------------------+--------+--------+
|                 ID                         |        Name          | Status | Server |
+--------------------------------------------+----------------------+--------+--------+
| 25f55f13-2f41-4000-87dd-7bd7b94f2117 | Ubuntu 12.04 LTS | ACTIVE |        |
+--------------------------------------------+----------------------+--------+-------+
```

g) Horizon Kurulumu

Son olarak OpenStack'ın web tabanlı yönetim arayüzünün kurulumu gerçekleştirilir.

```
./openstack_horizon.sh
```

Horizon kurulumu tamamlandığında konsola web arayüzünün URL'si yazılır. Bu bağlantı adresi üzerinden "admin" kullanıcı adı ve kurulum sırasında verilmiş olan veri tabanı şifresiyle giriş yapılır. Bu ekran Şekil 3.4'te gösterilmiştir.

h) OpenStack İle Sanal Makinelerin Yönetimi

OpenStack'ın Horizon bileşeni sayesinde herhangi bir komut yazmadan sadece web arayüzü kullanılarak sanal makineler başlatılabilir ve konfigüre edilebilir. Bu çalışmada sanal makinelerle ilgili işlemler daha kolay olması nedeniyle web ara yüzü kullanılarak gerçekleştirilmiştir. OpenStack dökümantasyonunda komutların kullanımıyla da bu işlemlerin nasıl gerçekleştirilebileceği anlatılmaktadır. Sırasıyla web arayüzü (Horizon)→Project Images & Snapshots→Ubuntu 12.04 LTS→Launch adımları takip edilerek yeni bir makine başlatılır. Şekil 3.3'te gösterildiği gibi 4 adet yeni makine başlatılmıştır. Bu makinelerin her biri 2 çekirdekli işlemci, 4 GB RAM ve 10 GB sabit disk kapasitesine sahiptir. Hadoop ve Spark kurulumları bu makineler üzerine yapılarak küme oluşturulmuştur.

3.1.3.Hadoop ve Kurulum Adımları

Bölüm 1.7'de anlatıldığı üzere MapReduce bilgisayar kümeleri üzerinde algoritmaların dağıtık ve paralel bir şekilde çalıştırılması amacıyla geliştirilmiş bir programlama modelidir.

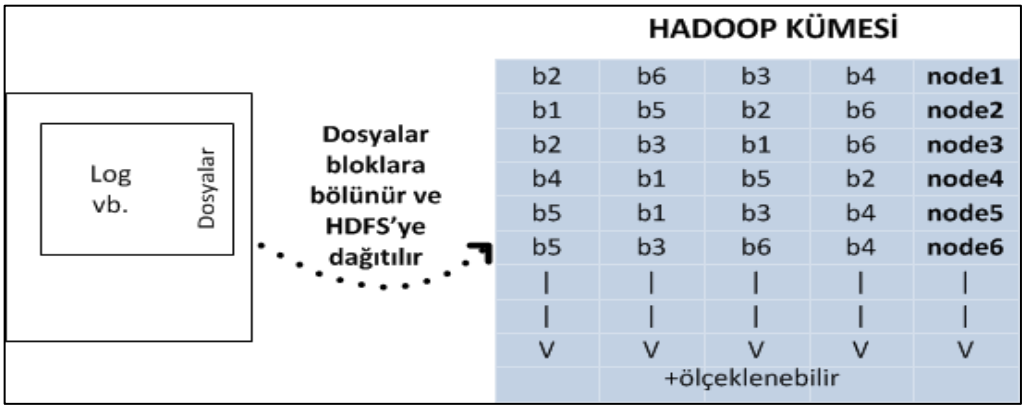
MapReduce modeline göre yazılan programların çalıştırılabilmesi için öncelikle bu model temel alınarak yazılmış sınıfların olduğu bir birim gereklidir. Bu çalışmada Google'ın MapReduce modelini anlattığı makalesindeki[25] bilgiler ile geliştirilmiş açık kaynaklı Apache projesi olan Hadoop kütüphanesi kullanılmıştır. Kullanılan Hadoop versiyonu 1.0.3 olup bu çalışmada Hadoop ile ilgili verilen bilgilerin bu versiyon üzerindeki sürümler için farklı olabileceği göz önünde bulundurulmalıdır.

MapReduce programının çalıştırılması için önemli diğer bir bileşen de bir dağıtık dosya sistemidir. Verilerin bilgisayarlar üzerinde dağıtılması, işlenmesi ve sonuçların yazılması için gerekli olan dağıtık dosya sistemini oluşturmak için birden fazla bilgisayardan oluşan, birbiriyle ağ bağlantısı yoluyla konuşabilen bilgisayarlar gerekmektedir. Hadoop, Google Dosya Sistemi (GFS) baz alınarak geliştirilmiş olan Hadoop Dağıtık Dosya Sistemine (HDFS) sahiptir [38]. HDFS, GFS'nin açık kaynaklı

geliştirilmiş versiyonudur [30]. Gerçek bir dosya sistemi olmayan HDFS, işletim sisteminin sahip olduğu Ext3, Ext4 gibi dosya sistemlerini kullanan soyut bir dosya sistemidir. HDFS ile petabaytlar seviyesindeki verilerin dağıtık kümeler üzerinde analiz edilebilmesi hedeflenmiştir.

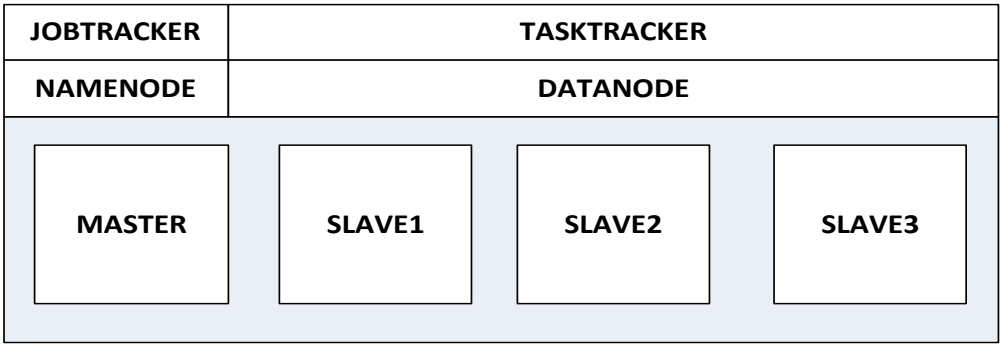
Hadoop sisteminde veriler bloklara bölünür. Her bir bloğun büyüklüğü varsayılan olarak 64 MB olarak ayarlanmıştır. Farklı blok büyüklüğü *dfs.block.size* parametresiyle ayarlanabilir. Verilerin bloklara ayrılırması 64 MB'lık veri büyüklüğü elde edien satırdan sonra yeni bir blok oluşturarak gerçekleştirilir. Şekil 3.5'te giriş verisinin b1,b2,b3… bloklarına ayrılması ve Hadoop kümesi üzerindeki node1, node2, node3… bilgisayarlarına dağıtılması gösterilmiştir. Bu bloklar Hadoop kümesi üzerindeki bilgisayarlara rastgele bir şekilde dağıtılır. Hadoop'un konfigürasyon dosyalarından olan hdfs-site.xml dosyasındaki parametreler değiştirilerek replikasyon faktörü belirlenebilir. Şekil 3.5'te de görüldüğü üzere bir blok replikasyon sayısına bağlı olmak üzere birden fazla bilgisayarda tutulmaktadır. Bu sayede her verinin bloklar seviyesinde yedekleri olur ve verinin üzerinde bulunduğu bilgisayarda herhangi bir problem olması durumunda programın akışına bir zarar gelmez, veri kaybı yaşanmaz [39].

Hadoop sisteminde verileri saklayan bilgisayarlara Data Node denir. Name Node ise bu verilerin küme üzerinde nasıl dağıtıldığını yani bir nevi haritasını tutan meta verilerin organizasyonundan sorumludur. Bir Hadoop kümesinde bir tane Name Node bulunur, bunun dışındaki bilgisayarlar Data Node'dur. Bir bilgisayar Name Node olarak çalışırken aynı zamanda Data Node olarak da çalıştırılabilir [39]



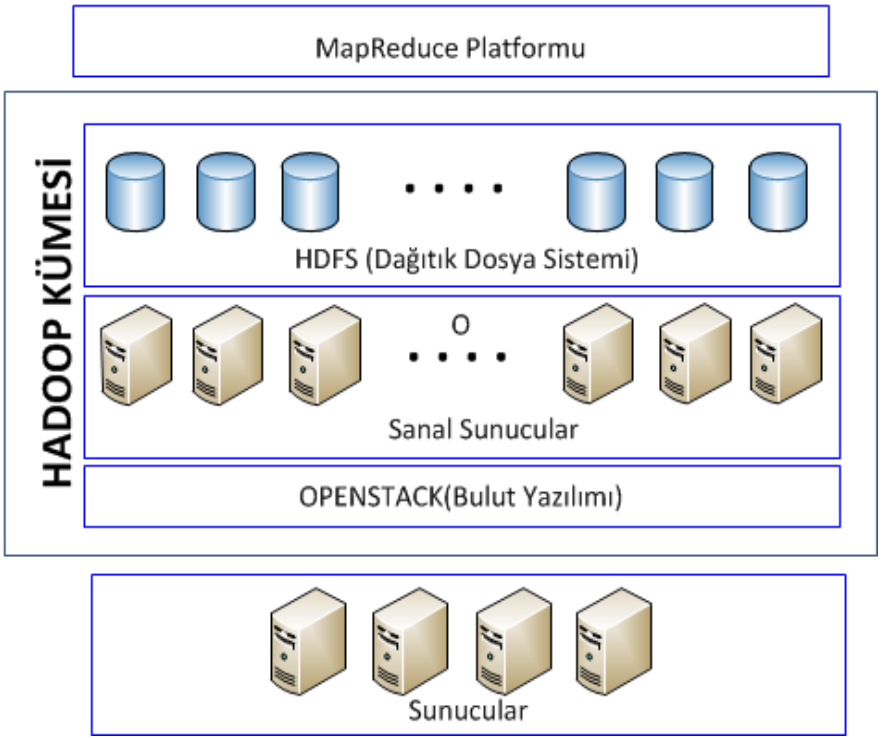
Şekil 3.5. HDFS Blok Yapısı

Oluşturacağımız kümenin bilgisayarlarını önceki bölümde (OpenStack kurulumu) kurulumu gösterilen OpenStack bulut sistemin üzerinde başlattığımız 4 adet sanal makine oluşturmaktadır. Bu makinelere master, slave1, slave2, slave3 isimleri verilmiştir. Hadoop kümesinde Master bilgisayar NameNode ve JobTracker olarak görev yaparken Slave bilgisayarlar DataNode ve TaskTracker olarak görev yapar. Hadoop kümesi şekil 3.6'da gösterilmiştir.



Şekil 3.6. Hadoop kümesi

Bu sanal makineler Grid1 adını verdiğimiz bu bulut sistemi üzerinde bulunduğu için aynı bilgisayar ağında bulunmaktadır. Platformun genel şeması Şekil 3.7'de verilmiştir.



Şekil 3.7. Bulut uygulama mimarisi

Kurulum adımları ise şu şekildedir:

a) Makinelerin kuruluma hazırlanması

Açık kaynaklı Hadoop sistemi Java tabanlı olarak yazılmıştır. Dolayısıyla çalışacağı platformda bir JDK (Java Development Kit) kurulu olmalıdır. Bu çalışmada Hadoop kümesindeki bilgisayarlara Oracle JDK versiyon 7u21 kurulmuştur. Oracle'ın web sitesinden işletim sistemi için uygun olan JDK kurulum dosyaları indirildikten sonraki kurulum adımları aşağıda verilmiştir.

```
tar xvf jdk-7u21-linux-x64.gz
sudo mkdir -p /usr/lib/jvm/jdk1.7.0
sudo mv jdk1.7.0_21/* /usr/lib/jvm/jdk1.7.0/
sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/lib/jvm/jdk1.7.0/bin/java" 1
sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/lib/jvm/jdk1.7.0/bin/javac" 1
sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"/usr/lib/jvm/jdk1.7.0/bin/javaws" 1
```

Kurulumun sağlıklı bir şekilde gerçekleşip gerçekleşmediği aşağıdaki komutla test edilebilir.

```
java -version
```

Bu komutun aşağıdaki gibi bir çıkış vermesi kurulumun gerçekleştiğini gösterir. Versiyon özellikleri değişebilir.

```
output : java version "1.7.0_21"
Java(TM) SE Runtime Environment  (build 1.7.0_21-b11)
Java HotSpot(TM) 64-Bit Server VM (build 23.21-b01, mixed mode)
```

JPS (Java Process Status) aracı yazılımının etkinleştirilmesi Hadoop için zorunlu olmamakla birlikte çalışmakta olan Hadoop işlemlerinin takip edilebilmesini sağlaması açısından yararlı olacaktır.

```
sudo update-alternatives --install /usr/bin/jps jps
/usr/lib/jvm/jdk1.7.0/bin/jps 1
```

b) Kurulumun yapılması

Hadoop kurulum dosyaları Apache Hadoop web sayfasından [29] indirilebilir. Hadoop kurulumu tek bir bilgisayar üzerinde yapılabilir. MapReduce programları yazılırken küçük boyutlu veriler üzerinde programın istenen şekilde çalıştığını test etmek için tek bilgisayarlı bir küme oluşturabilir. Eğer Hadoop büyük boyutlu verilerle uğraşılan durumlarda kullanılacaksa çok sayıda bilgisayardan oluşan kümeler oluşturmak gerekir. Örneğin Yahoo şirketinin 4500 bilgisayardan oluşan bir Hadoop kümesi bulunmaktadır [40]. Hadoop kümesi oluşturmak için bilgisayarlarda Hadoop'un kurulmuş olması gerekir. Bu bilgisayarlarda gerekli konfigürasyon parametleri değiştirilerek kümeye eklenirler. Çok sayıda bilgisayardan oluşan kümenin kurulumu için farklı otomatikleştirme yöntemleri kullanılabilir. Kendi kümemize özel araçlar programlayarak kümeye yeni bilgisayarın eklenmesi ve konfigürasyonu için harcanan süreyi kısaltabiliriz. Bu gereksinimler ve birçok MapReduce programlama modeliyle geliştirilmiş yazılım araçları büyük veri çözümleri sunan çeşitli firmalarca satılmaktadır. Bu firmalar örnek olarak Cloudera verilebilir. Bulut sistemi üzerinde oluşturulacak kümelerde bulut bilişim teknolojilerinin avantajlarından faydalanılabilir. Bulut üzerinde yeni makine imajı oluşturma, imaj görüntüsü alma ve bir makinenin yeni kopyalarının başlatılması işlemleri oldukça kolaydır.

Bu çalışmada 4 bilgisayardan oluşan bir Hadoop kümesi oluşturulmuştur. Bilgisayar sayısı çok fazla olmadığı için kümedeki bilgisayarlara önce Hadoop kurulumu yapılıp daha sonra bir master bilgisayar çatısı altında diğer 3 bilgisayar kümeye dâhil edilmiştir.

c) Kullanıcı işlemleri

Hadoop yüklenen bilgisayarda kullanıcı yönetimiyle ilgili işlevlerin yönetilebilirliğini ve izinlendirmelerin karmaşıklığının önüne geçmek için bir kullanıcı grubu ve bu grup içersinde bir kullanıcı oluşturmak yararlı olacaktır. Biz de Hadoop adında bir kullanıcı grubu ve kullanıcı oluşturduk.

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hadoop
```

//kurulum dizinini oluştur

```
sudo mkdir /usr/local/hadoop
```

//hadoop kullanıcısına kurulum dizininde yetki ver

```
sudo chown -R hadoop:hadoop /usr/local/hadoop
```

//hadoop kullanıcısına sudo komutu yetkisi verilmesi /etc/sudoers dosyasına aşağıdaki satır eklenir

```
hadoop ALL = (ALL) ALL
```

d) SSH bağlantı ayarları

Telnet benzeri bir program olan SSH bir linux makinanın güvenli bir şekilde uzaktan yönetilebilmesini sağlar. SSH secure shell kelimelerinin kısaltılmışıdır. MobaXterm, Cygwin, Putty gibi programlar kullanılarak Windows makineler de Linux bilgisayarları SSH bağlantısıyla yönetebilir.

Bağlanılacak bilgisayarın ip adresi ve bağlanılan kullanıcının şifresiyle SSH bağlantısı yapılabilir. Fakat Hadoop'daki master bilgisayarın bir çok kez slave bilgisayarlara bağlanması gerekecektir ve bilgisayarlara yapılan her bağlantıda bir kişinin manuel olarak şifre girmesi mümkün olmayacağı için SSH bağlantısının şifresiz olarak gerçekleştirilebilmesi gereklidir. Bu şekilde MapReduce programı çalışırken işler gönderilebilecek ve mesajlar alışverişi yapılabilecektir. Küme sadece bir tane bilgisayardan oluştuğu durumda ise localhost'a şifresiz SSH bağlantısı yapılabilmelidir.

Şifresiz SSH bağlantısı için yapılması gereken iş bağlanılacak bilgisayarın kabul ettiği bağlantıların bulunduğu dosyaya (authorized_keys dosyası) anahtarının gönderilmesidir. Bu şekilde bağlantı kurulmak istenildiğinde istekte bulunan bilgisayarın ip adresi ve önceden kaydedilmiş anahtarına bakılarak şifresiz bağlantı kurması sağlanır.

Slave makinelerden birine şifresiz bağlanmak için gerekli komut adımları şu şekilde gerçekleştirilmiştir:

- SSH anahtarları oluşturulur

```
hadoop@master:~$ ssh-keygen -t rsa
```

```
hadoop@slave1:~$ ssh-keygen -t rsa
```

-Master bilgisayarda oluşturulan anahtar slave makinelere gönderilir. Bu adımdaki işlemlerin hadoop kullanıcısı olarak yapılması önemlidir.

```
hadoop@master:~$ cat /home/hadoop/.ssh/id_rsa.pub >>
/home/hadoop/.ssh/authorized_keys
hadoop@master:~$ scp /home/hadoop/.ssh/id_rsa.pub
slave1:/home/hadoop/.ssh/master.pub
```

-Yukarıdaki kopyalama işlemi OpenStack'da .pem uzantılı anahtar kullanılarak yapılmalıdır.

```
cp /home/hadoop/.ssh/id_rsa.pub /home/ubuntu/id_rsa.pub
scp -i bulutkey.pem id_rsa.pub   slave1:/home/ubuntu/master.pub
```

-Alınan anahtar geçerli anahtarlar listesine yazılır.

```
cat /home/ubuntu/master.pub >> /home/hadoop/.ssh/authorized_keys
```

-Test

Master bilgisayardan slave bilgisayara aşağıdaki komut satırıyla bağlantı kurulur.

```
ssh slave
```

Bu komut adımları tüm slave bilgisayarlar için tekrar edilmelidir.

e) Kurulum dosyalarının konfigürasyonu

Bu aşamada Hadoop kurulum dosyaları önceden oluşturduğumuz kurulum dizininde açılır. Yapılaması gereken işlemler Ekler bölümünde ekran görüntüleriyle birlikte verilmiştir.

f) Hadoop kümesinde kelime sayfa örneğinin çalıştırılması

Programa dillerinin öğretilmesinde bir klasik olan "merhaba dünya" programının Hadoop için karşılığı wordcount, yani kelime sayma programıdır. Bu örnekte Hadoop kurulumuyla birlikte gelen hadoop-examples.jar wordcount örneği test dökümanları üzerinde çalıştırılmıştır. Metin tabanlı dosyalar halinde romanlar dosya sistemine yüklenip, bu romanlarda hangi kelimenin kaç kez geçtiği bulunmuştur.

WordCount programının kodları Ek 2'de verilmiştir.

3.1.4.Spark Ve Kurulum Adımları

Bir Spark kümesi oluşturmanın farklı yolları bulunmaktadır. Spark üç seçenekle birlikte gelir. Birincisi Apache'nin genel amaçlı bir küme yönetim sistemi olan ve farklı uygulamaların aynı ortamı paylaşmasını sağlayan Mesos kümesi üzerine kurulumdur. İkinci seçenek Hadoop 2'nin kaynak yönetim sistemi olan Hadoop Yarn üzerine kurulmasıdır. Bu çalışmada Spark'ın tek başına kurulumu (standalone) olan seçeneği kullanılmıştır. Kurulum manuel olarak önce makinelere Spark kurulumu yapılıp daha sonra master olan düşümden diğerlerine erişima ayarları ve konfigürasyonlar yapılarak Spark kümesi oluşturulmuştur.

Spark'ın yükleneceği sistemde Java, Scala, Maven, Python ve Git programlarının mevcut olmaları gerekmektedir. Öncelikle eğer yoksa bu programlar yüklenir. Tez çalışmasında yapılan kurulumlar şu şekildedir:

- Ubuntu 13.10
- Oracle Java 7
- Scala 2.9.3
- Maven 3.0.4
- Python 2.7.3
- Git

Eğer sistemde Java yüklü değilse önce Java yüklenir. Java'nın Ubuntu üzerinde kurulumu Hadoop kurulumu bölümünde detaylı olarak gösterilmiştir. Java kurulduktan sonra JAVA_HOME çevresel değişkeni mutlaka tanımlanmalıdır. Daha sonra Scala kurulur ve aynı şekilde SCALA_HOME tanımlanır. Maven, Python ve Git kurulumlarından sonra Spark'ın kurulumu şu şekilde yapılabilir:

//Spark'ı indirip, dosyaları çıkartıp, kurulum dizinine geç

```
wget http://d3kbcqa49mib13.cloudfront.net/spark-1.0.0.tgz
tar -zxf spark-1.0.0.tgz
cd spark-1.0.0
```

Spark indirilip bulunduğu dizine geçilir ve kurulumu (build) gerçekleştiren satır yazılır:

```
./sbt/sbt assembly
```

Bu adımdan sonra Spark gerekli dosyaları indirip kurar. İşlem tamamlandığında kurulumun başarılı bir şekilde tamamlandığını gösteren bir mesaj gösterir. Kurulumun sağlıklı bir şekilde yapıldığını tespit etmek için Spark'la birlikte gelen örnek programlar çalıştırılabilir. Bu programlar derlenmiş olarak "examples" dizininde bulunmaktadır.

Örnek olarak Pi sayısının değerini hesaplayan programın Spark-Shell üzerinde çalıştırılması ve sonucu şu şekildedir:

```
./run-example org.apache.spark.examples.SparkPi local
Pi is roughly 3.14634.
```

Spark üzerinde programlar çalıştırılırken kullanılan bazı önemli parametreler bulunmaktadır. Örnekte gösterilen Pi programı çalıştırılırken verilen "local" parametresi bu programın sadece o bilgisayarda çalışacağını gösterir. Bu parametreyle birlikte kaç çekirdeğin kullanılacağı da "local[n]" şeklinde n değerine işlemci çekirdek sayısı yazılarak belirlenebilir. Eğer program tek bir bilgisayarda değil de küme üzerinde çalıştırılacaksa "local" parametresi yerine "master=ip:port" şeklinde parametreyle master bilgisayarın ip adresi ve Spark için kullanacağı portu belirtilir ve bu şekilde program küme üzerinde çalıştırılır.

Spark'ın en kullanışlı özelliklerinden birisi de terminal üzerinden komutlarla kullanılması imkânıdır. Spark terminali (spark-shell) de yine "local" veya "master" parametreleriyle çalıştırılabilir. Spark terminalinin kullanımına örnek olması açısından aşağıda Spark'ın kurulum dizininde bulunan "README.md" dosyasında kaç tane kelime olduğunu bulan komutlar gösterilmiştir.

```
./bin/spark-shell
```

```
scala> val textFile = sc.textFile("README.md")
scala> textFile.count()
res0: Long = 126
```

Şimdi de bu dosyada Spark kelimesinin kaç tane satırda geçtiğini bulalım.

```
scala> val textFile = sc.textFile("README.md")
scala> textFile.filter(line => line.contains("Spark")).count()
res3: Long = 15
```

3.2.Paralel Log Analizi

Web sunucuları kullanıcıların web sitesiyle olan etkileşimlerini log dosyalarına kaydeder. Sunucu logları genellikle kullanıcı hareketlerinin istemci kimliği, zaman damgası, nesne kimliği, boyut, metot, durum, çeşit, sunucu gibi bilgilerle satırlar halinde metin dosyalarına yazılmasıyla oluşur.

Log dosyalarının analiziyle birçok yararlı bilgi elde edilebilir. Log dosyalarının analizi için özel yazılımlar bulunmaktadır. Örneğin; AWStats, Webanalizer. Bu uygulamada log dosyalarının çok büyük olduğu durumlarda etkili bir şekilde analiz edilmesine bir çözüm olması amacıyla paralel log analizi yapılmıştır. Log dosyalarının bilgisayar kümesi üzerinde dağıtılıp paralel bir şekilde işlenmesiyle çok hızlı bir şekilde analiz yapılabildiği görülmüştür. Hadoop ile web sunucusu log dosyaları üzerinde çeşitli analizler yapılıp performansı incelenmiştir. Verilerin analizini otomatik olarak gerçekleştiren dağıtık bir log analiz sistemi tasarlanmıştır. Ayrıca karşılaştırma amacıyla aynı iş hem Hadoop ve Spark kümeleri üzerinde yürütülüp performansları karşılaştırılmıştır.

3.2.1.Test Verileri

Analizlerde Microsoft Internet Information Server log dosyaları veri olarak kullanılmıştır. IIS log'u şu başlıklardan oluşur;

Date, Time, Client IP Address, User Name, Service Name and Instance Number, Server Name, Server IP Address, Server Port, Method, URI Stem, URI Query, HTTP Status,

Win32 Status, Bytes Sent, Bytes Received, Time Taken, Protocol Version, Host, User Agent, Cookie, Referrer, Protocol Substatus.

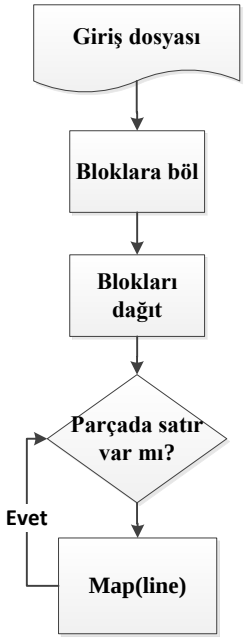
Kullanılan veri içerisindeki bir satır şu şekildedir;

```
2013-04-15 00:00:07 W3SVC1 10.1.1.5 GET
/ilahiyat/Tr/BilimselFaaliyetler/FakulteDergisi.htm - 80 -
66.249.78.66
Mozilla/5.0+(compatible;+Googlebot/2.1;++http://www.google.com/b
ot.html) - www.firat.edu.tr 404 0 3 1795
```

Satırlar belirli bir düzende olduğundan dolayı MapReduce program aşamalarında verilerin bulunduğu boşluk sırasına göre hangi bilgi olduğu anlaşılmaktadır. Örneğin ilk boşluk sonrası erişilen zamanı gösterir.

3.2.2.Hadoop İle Log Analizi

MapReduce programı dağıtık dosya sistemi üzerinde bulunan dosyaları giriş olarak kullanır. Öncelikle işlenecek veriler HDFS üzerinde 64 MB'lık bloklar olarak dağıtılır. Bu dosyalardaki her bir satır kümedeki bilgisayarlar tarafından satırlar halinde paralel olarak işlenir.

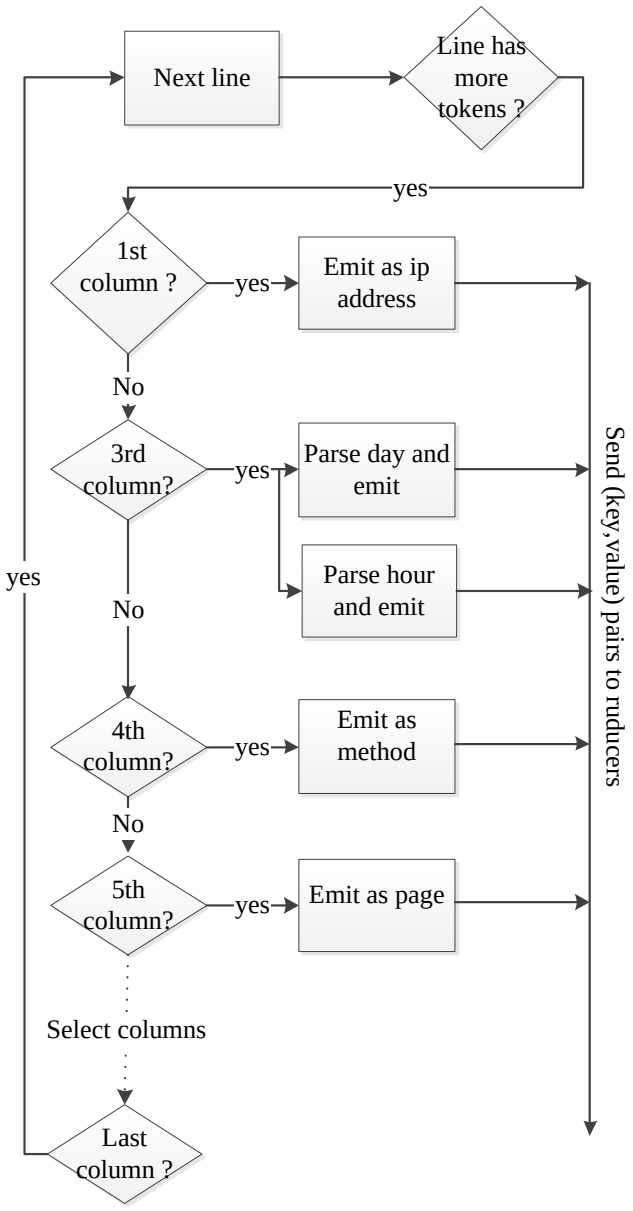


Şekil 3.8. Hadoop iş akış diyagramı

Şekil 3.8'de Hadoop ile yapılan analizlerin genel akış şeması verilmiştir. Hadoop ile yapılan analizler tek bir MapReduce çevriminde veya birbirine bağlı birden fazla MapReduce çevriminden oluşmasına göre ele alınmıştır.

a) Tek Çevrimli (Single) Çok Çıktılı İşler

Sunucu kayıtları içerisindeki bir kelimenin frekansı çeşitli önemli bilgiler verebilir. Örneğin belirli bir tarihte belirli kullanıcıların erişim oranları ya da en çok erişilen sayfa bu şekilde bulunabilir. Bunu yapabilmek için satırda bulunan her kelime bir anahtar olur ve 1 değeri atanarak map edilir. Map safhası bitip (anahtar, değer) çiftleri reduce aşamasına gönderilmeden önce anahtar değerlerine göre gruplanır ve reduce aşamasında her bir anahtar değerinin toplam kaç kez geçtiği değeri çıkışa yazılır.

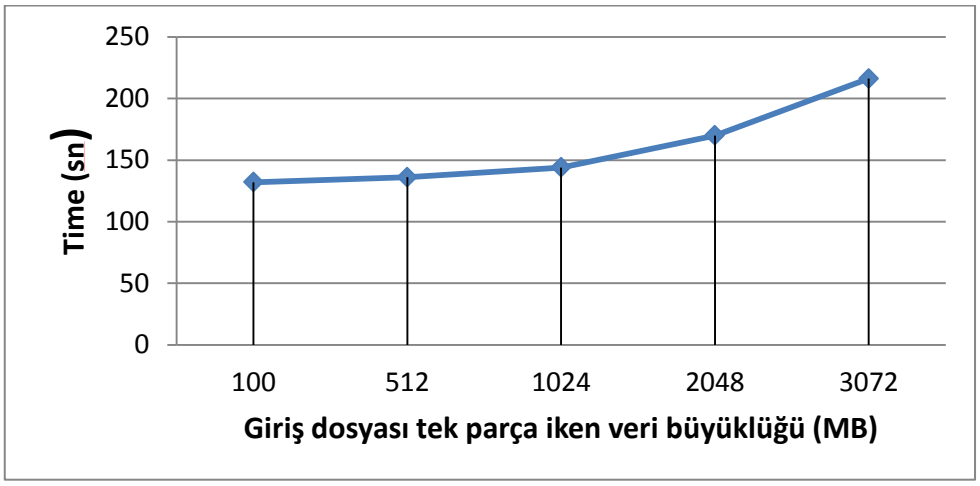


Şekil 3.9. Hadoop log analizi akış şeması

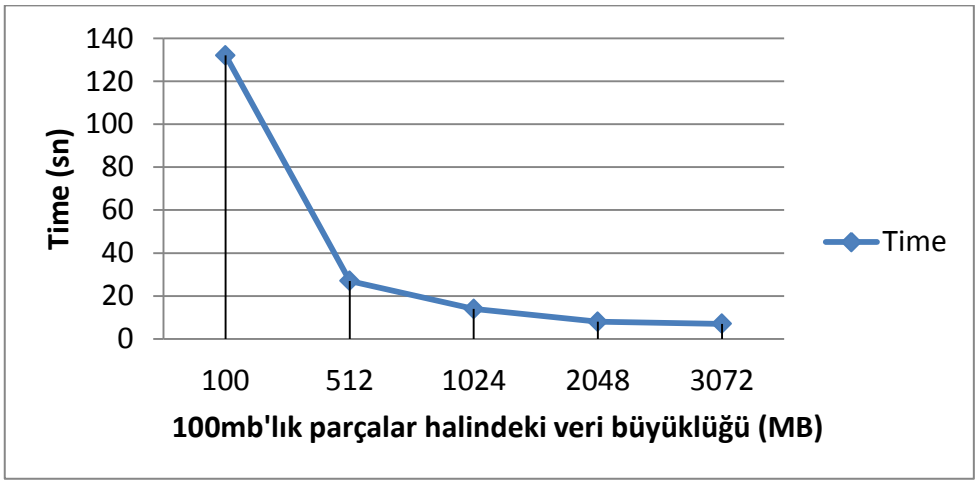
Kayıt dosyasındaki tüm kelimelerin map edilmesi gerekli değildir. Sadece analizi yapılmak istenen alanların değerleri map edilir ve diğerleri atlanır. Şekil 3.9'da ip, gün, saat, erişim metotu ve erişilen sayfalar için frekans değerlerinin nasıl bulunduğu gösterilmiştir.

Bu yaklaşımdaki problem ise tüm bilgilerin çıkış dosyasına karışık bir şekilde yazılıyor olmasıdır. Standart Hadoop bu şekilde çalışmaktadır. Yani çıktılar otomatik olarak tek bir dosyaya yazılır. Reduce aşaması biterken her bilginin kendisiyle ilgili bilgilerin olduğu ayrı bir dosya içerisine yazılmasını sağlayarak bu problemi çözmüş

olduk. Bunu gerçekleştirmek için Map aşamasında anahtar değerlerinin önüne birer ön ek getirildi. Bu ön ek reduce aşaması bittiğinde bu anahtara sahip verinin hangi dosyaya yazılacağını göstermektedir. Bunun için Hadoop API'sindeki FileOutputFormat sınıfından türetilmiş MultipleOutputFormat soyut sınıfı kullanılmıştır. Hadoop işinin konfigürasyon dosyasında setOutputFormat özelliği MultipleOutputFormat olarak atanmıştır. Böylelikle ip_part-00000, gün_part-00000, saat_part-00000, metot_part-00000, sayfa_part-00000 çıkış dosyalarında ip, gün, saat, erişim metotu ve erişilen sayfa bilgileri yazılmıştır. Bu yöntemin kullanılmadığı durumda, kayıt dosyasındaki her bir alan için ayrı bir Hadoop işi çalıştırılması gerekir ve bu hiç iyi bir yaklaşım değildir. Fakat yine de her zaman tek bir MapReduce işi ile istenilen analizi yapmak mümkün olmamaktadır. Çoğu zaman ancak birden fazla işin birbirine bağlı bir şekilde arka arkaya çalıştırılmasıyla istenen bilgi elde edilebilmektedir.



Şekil 3.10. Log analizi sure diyagramı



Şekil 3.11. Her 100MB için analiz süreleri diyagramı

Şekil 3.10'da giriş verisinin miktarına göre Hadoop üzerinde yapılan log analizi işlerinin çalışma süreleri gösterilmiştir. Şekil 3.11'de giriş verisinin 100 MB olduğu durumdaki analiz süresinin orantısı verilmiştir. Böylece (süre*100/size) şeklinde elde edilen değerler performans değeri olarak kabul edilerek 100 MB'lık veri miktarına düşen çalışma süreleri gösterilmiştir. Görüldüğü üzere veri miktarındaki artış performansı üssel olarak arttırmaktadır. Performans artışının daha büyük veriler üzerinden ve daha fazla makinenin parallel olarak çalıştırılması durumunda daha yüksek olacağını düşünmekteyiz.

b) Zincir (Chanined) MapReduce İşleri

Bir önceki bölümde de bahsedildiği gibi tek bir Hadoop işi yürütülmesi her zaman yeterli olmamaktadır. Karmaşık analizler yapılırken paralel programlamanın en önemli engellerinden biri olan paralelleştirilemeyen işlemler mutlaka karşımıza çıkacaktır. Log analizi yaparken de ne zaman, en çok, en az, hangisi gibi soruların dışında "sunucunun en yoğun olduğu gün en çok erişilen sayfası" gibi sorulara cevap aranmaktadır. Bu tür sorgulara SQL tipinde sorgu denilebilir. Örneğin, elimizde en çok erişim yapılan gün bilgisinin olduğu varsayılırsa, "sunucunun en yoğun olduğu gün en çok erişilen sayfası" bir veri tabanı sisteminde aşağıdaki gibi sorgulanabilir.

SELECT sayfa *FROM* kayıtlar

WHERE gün = *MAX*(gün)

Hadoop'un bu sorgu sonucunu tek bir iş yürüterek gerçekleştirmesi mümkün değildir. Bu problem için bazı çözüm yöntemleri geliştirilmiştir. Örneğin iteratif Hadoop işleri için HaLoop altyapısı (framework) geliştirilmiştir [41] . Rubao Lee ve arkadaşları da SQL tipi sorguların MapReduce işlerine dönüştürülmesi ve optimizasyonu için bir sistem[42] geliştirmiştir.

Bu örnek sorgu için gerekli MapReduce adımları Job1, Job2, Job3 olarak verilmiştir.

- 1 numaralı iş (Job1) Map1&Reduce1 adımlarıyla siteye hangi gün kaç adet ziyaret gerçekleştirildiğini bulur.
- 2 numaralı iş Map2&Reduce2 adımlarıyla output1'de elde edilen sonuçları sıralar ve en büyük değere sahip günü bulur. Bu iş Reduce adımındaki sıralama işlemini tetiklemek için yapılır.
- 3 numaralı iş Map3&Reduce3 adımlarıyla Job2'de elde edilen en yoğun gün içerisinde erişim yapılmış sayfaları bulur. Job3'den elde edilen output3 son değeri yani SQL tipindeki sorgunun sonucunu verir.

Buradaki map ve reduce adımlarını daha detayları bir şekilde açıklamak gerekirse: Öncelikle, 1 adet map (Map1) ve 1 adet Reduce (Reduce1) işleminden sonra günler ve karşılarında o günlerdeki erişim frekanslarını içeren output1 dosyası elde edilmiştir. Bu sorgu sonucunda sadece en çok erişim yapılan gün bulunur ve diğer veriler işin devamında kullanılmaz. Hadoop varsayılan olarak anahtarlar (key) üzerinde sıralama işlemi yapar. [25]'da gösterildiği üzere map sonrasında veriler reduce aşamasına geçirilmeden önce anahtarlar (key) artan bir şekilde otomatik olarak sıralanır. Hadoop'un bu özelliği günlerin frekanslarına göre sıralanmasında kullanıldı.

In	Emit	Out(çıkış dosyası)	
Map1(satır numarası, satır) LongWritable, Text	**Emit**(gün, one) Text, IntWritable	–	**Job1**
Reduce1(gün, [1,1,1…n]) Text, IntWritable	**Emit**(gün, $\sum_{i=0}^{i<n} value(i)$) Text, IntWritable	= output1 (gün, toplam erişim)	
Map2(satır numarası, satır) LongWritable, Text	**Emit**(token2, token1) IntWritable, Text	–	**Job2**
Reduce2(top. erişim, gün) IntWritable, Text	**Emit**(toplam erişim, gün) IntWritable, Text	= output2 (toplam erişim, gün)	
max_gün değişkenine en çok erişim yapılmış günün değerini atayıp konfigürasyon parametresi olarak belirle			
Map3(satır numarası, satır) LongWritable, Text	if(day=”max_gün”) **Emit**(sayfa, one) Text, IntWritable	–	**Job3**
Reduce3(sayfa, [1,1…n]) Text, IntWritable	**Emit**(page, $\sum_{i=0}^{i<n} value(i)$) Text, IntWritable	= output3 (sayfa, toplam erişim)	

Output1 (gün, toplam erişim) şeklinde ikili değerlerden oluşur ve token1=gün, token2=toplam erişim şeklinde ayrılarak map edilir. Sıralanmanın erişim sayısı üzerinden yapılmasını istediğimiz için (token1, token2) şeklinde olan ikililer (token2, token1) şeklinde emit edilir ve böylelikle anahtar konumuna alınan erişim saatleri

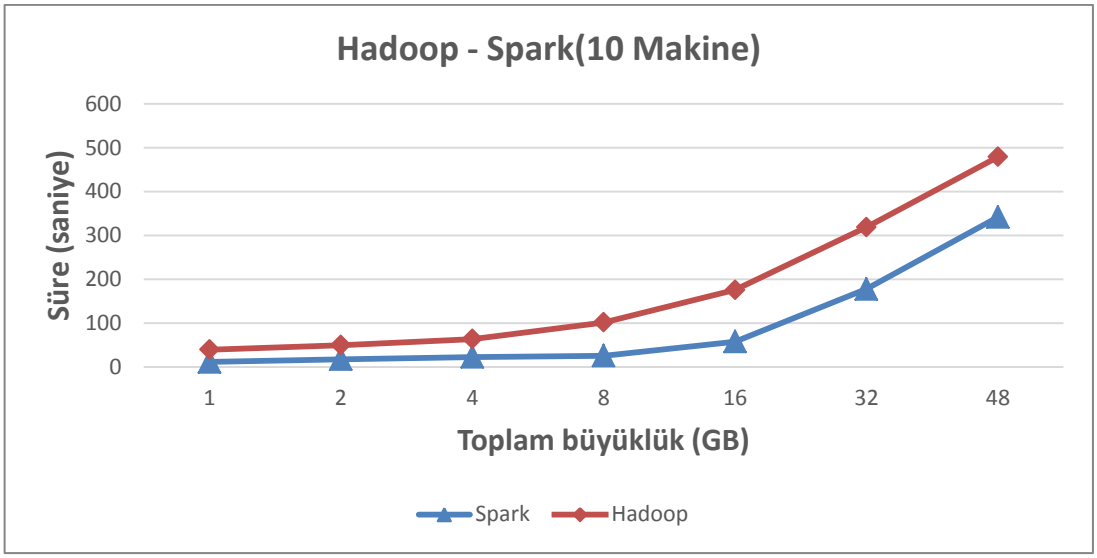
otomatik olarak sıralanır. Sıralama işlemi de tamamlandığı için artık bu Hadoop işi bitmiştir. Fakat sıralama işinin yapılması Reduce adımının bir ön adımı olduğu için reduce adımı da gerçekleştirilmelidir. İstenen değer zaten elde edilmiş olduğu için Reduce2 içerisinde bir işlem yapılmaz, gelen ikili değerler direk çıkışa aktarılır ve output2 elde edilir. Bu tip reduce adımına birim (identity) reduce denilir. Output2'nin son satırı sunucunun en yoğun olduğu günü gösteren max_gün değeri verir. Bu değer sonraki işin (Job3) konfigürasyon nesnesine aktarılır ve bu şekilde Job3'deki map ve reduce metotları bu değere erişebilir. Map3 max_gün değişkeni ile tutulan günde erişilen sayfaları anahtar (key), 1 sayısını da değer (value) olarak alır ve Reduce3'e gönderir. Reduce3'de erişim sayılarını yani birleri toplar ve output3 dosyasına yazar.

3.2.3. Spark ile Performans Karşılaştırması

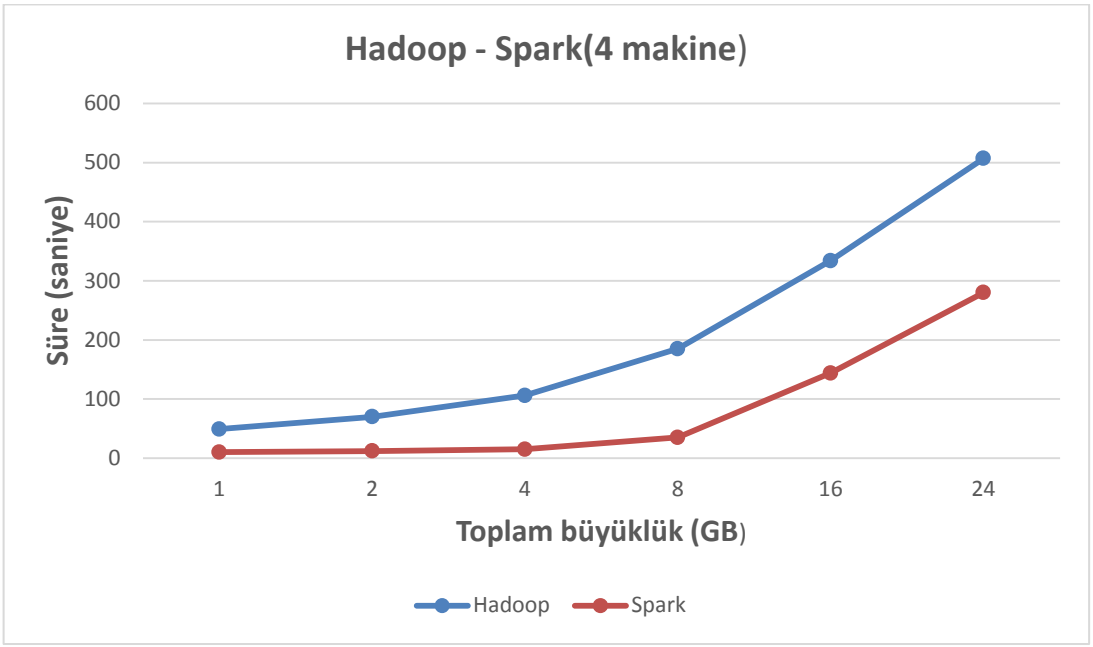
Spark, Apache'nin en yeni geniş ölçekli, büyük veri işleme platformlarından birisidir. Spark'ın Hadoop'a göre 10 kat daha hızlı çalıştığı öngörülmektedir [43]. Bu uygulamada Apache Hadoop ve Apache Spark'ın performanları dağıtık dosya üzerinde bir kelime sayma işlemi yapılarak karşılaştırılmıştır. Yapılan karşılaştırmalar sonucunda henüz ilk iterasyonda bile Spark'ın Hadoop'a göre daha hızlı sonuç verdiği görülmüştür.

Dosyalar üzerinde belirli bir kelimenin kaç kez geçtiğini bulan Java programları MapReduce ve Spark için ayrı ayrı yazılıp Hadoop ve Spark kümelerinde çalıştırılmıştır.

Karşılaştırmalar için 4 tane sanal makineden oluşan ve 10 tane sanal makineden oluşan Hadoop ve Spark kümeleri oluşturulmuştur.



Şekil 3.12. Hadoop ve Spark performans karşılaştırılması (10 makine)



Şekil 3.13. Hadoop ve Spark performans karşılaştırılması (4 makine)

Şekil 3.12 ve Şekil 3.13'te bir iterasyonluk kelime sayma işlerinin çalıştırılmasıyla elde edilen zamanlar gösterilmiştir. Karşılaştırmalar veri miktarının değişimine göre yapılmıştır. Ayrıca farklı sayıda makineden oluşan kümelerdeki performansları gözlenmiştir. Veri miktarı büyüdükçe Spark'ın Hadoop'a yakın bir performans gösterdiği görülmüştür. Çünkü Apache Spark in-memory yapıda

çalışmaktadır. Veri büyüklüğü hafıza kapasitesini aşınca Hadoop'da olduğu gibi disk giriş/çıkışı (I/O) şeklinde çalışmaktadır [43]. Bu durum bir iterasyonlu işlerde böyle olup, iterasyon sayısı arttığında veriler hafızadan direk kullanılacağı için değişmekte ve Spark 10 kata kadar daha hızlı çalışabilmektedir.

3.2.4. Log Analizi Sonuç

Bu uygulamada güncel ve popüler olan dağıtık veri işleme yöntemleriyle web sunucu log dosyalarının analizinin nasıl yapılabileceği ve performansları incelenmiştir. Geleneksel bilgisayar ve yazılımlarla işlenmesi çok fazla zaman aldığından dolayı işlenmesi mümkün olmayacak kadar büyük boyutlu verilerin Hadoop ve Spark gibi teknolojilerle standart özelliklerdeki bilgisayarlarla oluşturulan kümelerde paralel bir şekilde yüksek başarımlı olarak işlenmesi mümkündür.

Hadoop iyi bir yüksek başarımlı hesaplama alternatifidir ve etkisini gösterebilmesi için en az birkaç bilgisayardan binlerce bilgisayara kadar makineden oluşan kümeler gereklidir.

MapReduce tekniğinin gerçek gücü bir bilgisayar kümesi üzerinde verilerin dağıtık dosya sisteminde paylaştırılıp, veri ve iş yükünün paylaştırılması yani paralelleştirilmesiyle görülebilir. Bir kümenin oluşturulması, birden fazla makine üzerinde Hadoop'un çalıştırılması demektir. Sanallaştırma ile tek bir makine üzerinde veya birleştirilmiş tek bir makine gibi davranan makineler üzerinde sanal makineler oluşturulabilir, bu makinalar üzerinde işletim sistemi çalıştırılıp Hadoop düğümü (node) olarak kullanılabilir. Fakat çoğu sanallaştırma yazılımları yüksek lisans ücretleri ve kapsamlı profesyonel bilgi birikimi gerektirmektedir. Bu uygulamada açık kaynaklı bulut yazılımı olan OpenStack kullanılarak Hadoop düğümleri oluşturulmuştur. FutureGrid'in sağlamış olduğu kullanılmıştır. Ayrıca Fırat Üniversitesi 'nde oluşturduğumuz OpenStack bulut sistemi üzerinde de makineler oluşturulup testler yapılmıştır. Şekil 3.10 ve Şekil 3.11'deki çalışma süresi değerleri bu makineler kullanılarak elde edilmiştir. Hadoop kümelerinin manuel olarak kurulması işlemi çok fazla zaman aldığından dolayı etkili bir yaklaşım değildir. Cloudera, HortonWorks ve MapR gibi çözümler Hadoop gibi çok sayıda makineden oluşabilen kümelerin kurulumunu otomatikleştirmekte ve yönetimi için faydalı araçlar sağlamaktadır.

Bu uygulama ayrıca göstermiştir ki, eğer analiz dosyanın diğer alanlardan bağımsız bir alanı üzerinde yapılıyorsa Hadoop ile doğrudan yapılabilir. Eğer diğer alanlardaki değerlere bağımlı bir analiz gerçekleştiriliyorsa işin tamamı paralel olarak yapılamaz fakat yine de Hadoop ile kolay bir şekilde gerçekleştirilebilir. Öyle ki bazı durumlarda onlarca analiz (job) gerçekleştirilir ve bazılarının çıkışları diğerlerinin giriş verisi olarak kullanılır. Bu durumlarda iş zinciri (job chaining) kullanılabilir. Bu uygulamada SQL tipinde sorgular gibi karmaşık senaryolar için iş zinciri (job chaining) kullanılmıştır. HDFS üzerindeki çıkış dosyaları bir sonraki iş için giriş dosyası veya giriş parametresi olarak kullanılmıştır.

Ayrıca Apache Spark'ın interaktif terminali özelliğinin log analizi yapılırken arama, sayma vb. işlemler açısından oldukça kullanışlı olduğu gözlenmiştir. Spark terminalinin sunmuş olduğu fonksiyonel programlama ara yüzü sayesinde RDD (Resilient Distributed Datasets) veri tipinde giriş dosyaları kullanılarak baştan sona bir program yazmaya ihtiyaç duymadan paralel işlemler gerçekleştirilebilir. Spark'ın interaktif terminalinde kullanılan programlama dili Scala'dır. Terminalde örnek bir işlem Scala diliyle şu şekilde gerçekleştirilebilir:

logDosyası.filter(line.contains("error")).count()

Yukarıdaki bir satırlık giriş ile bir log dosyasında kaç tane satırda "error" kelimesinin geçtiği paralel bir şekilde hesaplanıp görüntülenelir. Bu özellik Spark'ı log analizi açısından oldukça kullanışlı yapmaktadır.

3.3. Dağıtık Doğal Dil İşleme Uygulaması

Makine öğrenmesi algoritmalarıyla sınıflandırma yapılırken dokümanlar vektör uzay modeliyle temsil edilirler. Dokümanların sayısal ortamda gösterimi ve özelliklerinin çıkarılması işlemlerinin nasıl yapıldığı makine öğrenmesi bölümünde gösterilmiştir. Metinlerin sınıflandırılmasında, sınıflandırma metodunun uygulanması öncesinde giriş verilerinin bazı aşamalardan geçirilmesi gerekir. Bu işlemler çoğunlukla kelime gövdelerinin bulunması ve önemsiz kelimelerin metinden çıkarılmasıdır. Paralelleştirilmesi çok kolay olan bu iş, büyük miktarda veri olduğu durumda çok fazla zaman almaktadır. Bu nedenle tez çalışmasında dağıtık doğal dil işleme uygulaması

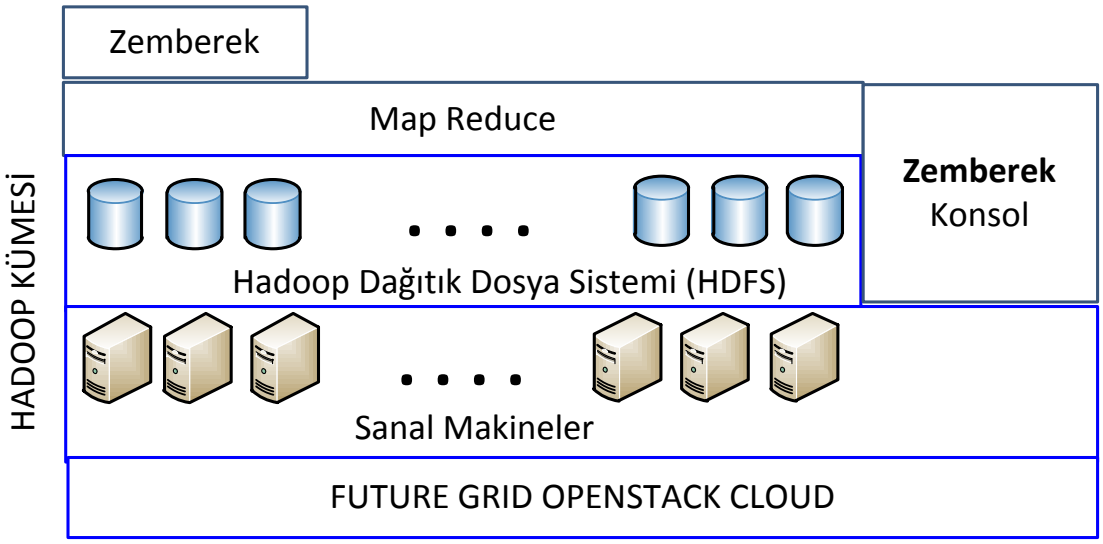
gerçekleştirilmiştir. Bu uygulama hangi yöntem kullanılırsa kullanılsın, sınıflandırma öncesi verilerin hazırlanması aşaması şeklinde kullanılabilir.

Bu çalışmada kullanılan bulut ortamı, bilgisayar kümesi, teknolojiler ve kurulumlar "log analizi" çalışmasıyla hemen hemen aynıdır. Yapılmak istenen iş aşağıda gösterilen birinci metinin ikinci metindeki hale dönüştürülmesidir.

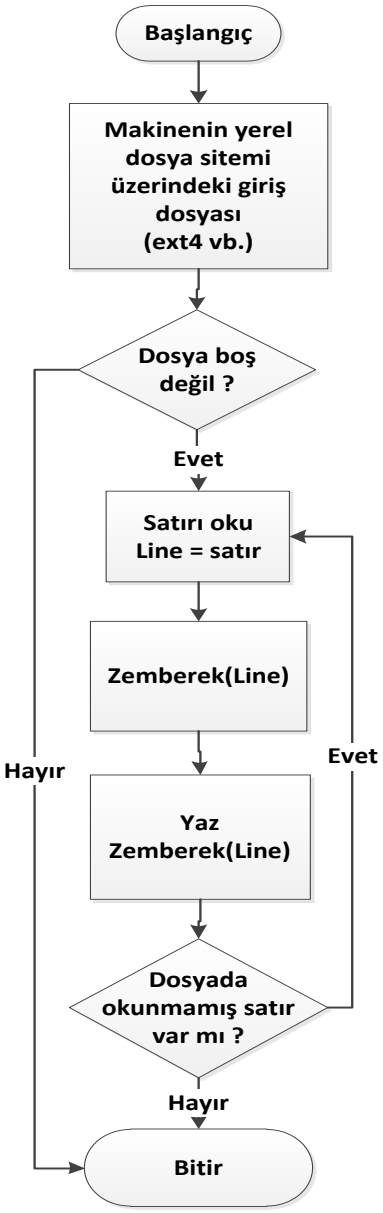
Feshe itiraz davası, işverence geçerli sebep gösterilmeden ya da kanunda öngörülen usule uyulmadan yapılan fesihlere karşı işçilerin başvurabileceği bir itiraz yolu olarak karşımıza çıkmaktadır.
Feshe itiraz dava işveren geçerli sebep gösterilmeden kanun öngör usul uyulmadan fesih işçi başvur itiraz yol olarak karşı

Bu işlem yapılırken kullanılan dil için bir doğal dil işleme metodu uygulanmalıdır. Bu aşama için Zemberek –ref- isimli doğal dil işleme kütüphanesi kullanılmıştır. Zemberek Java ile yazılmış açık kaynaklı bir projedir. Zemberek ile girişler kelimelerin gövdelerine dönüştürülmesiyle çıkışa aktarılmıştır.

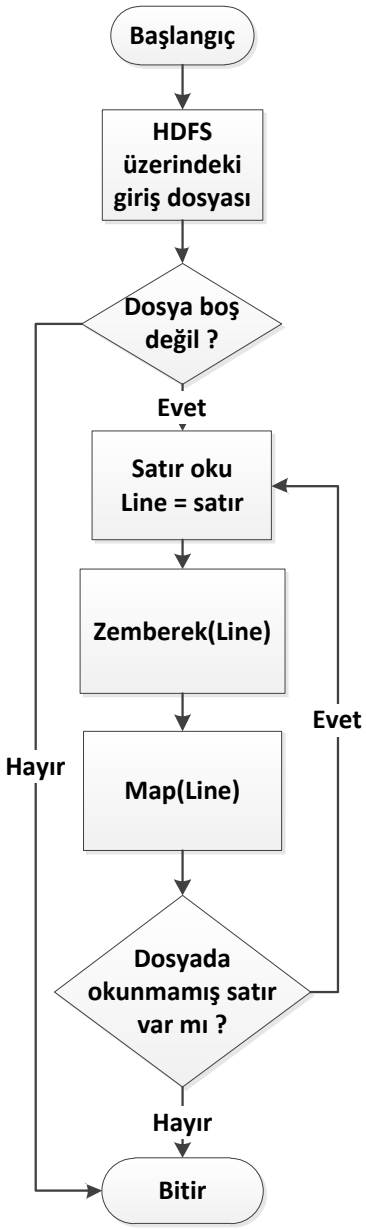
Bu işin paralel olarak yapılmasının performans olarak kazanımı ölçülmüştür. 2 çekirdekli, 4GB hafızaya sahip, 40 GB disk kapasiteli OpenStack üzerinde çalışan makinede ve bu makineyle aynı özelliğe sahip 7 adet makinede paralel olarak aynı iş yürütülerek programların çalışma süreleri ölçülmüştür. Paralel veri işleme Hadoop ile yapılmıştır. Test ortamı ve platformlar Şekil 3.14'de özetlenmiştir.



Şekil 3.14. Dağıtık doğal dil işlemesi mimarisi



Şekil 3.15. Konsol programı



Şekil 3.16. MapReduce programı

Giriş verisi olarak Türkçe dilinde yazılmış akademik makaleler kullanılmıştır. Pdf formatında toplanan makaleler PDF-Box Java kütüphanesi kullanılarak metin dosyalarına dönüştürülmüştür ve HDFS üzerinde metin formatıyla kaydedilmiştir. Karşılaştırma için farklı veri büyüklükleriyle testler gerçekleştirilmiştir. Giriş verileri 100MB, 1GB, 2GB, 4GB, 8GB ve 16GB olduğu durumlar için çalışma süreleri ayrı ayrı denemelerle hesaplanmıştır. Konsolda (Linux terminalinde) çalıştırılan standart Java

programının akış şeması Şekil 3.15'te ve Hadoop kümesi üzerinde çalıştırılan MapReduce programının akış şeması Şekil 3.16'da verilmiştir.

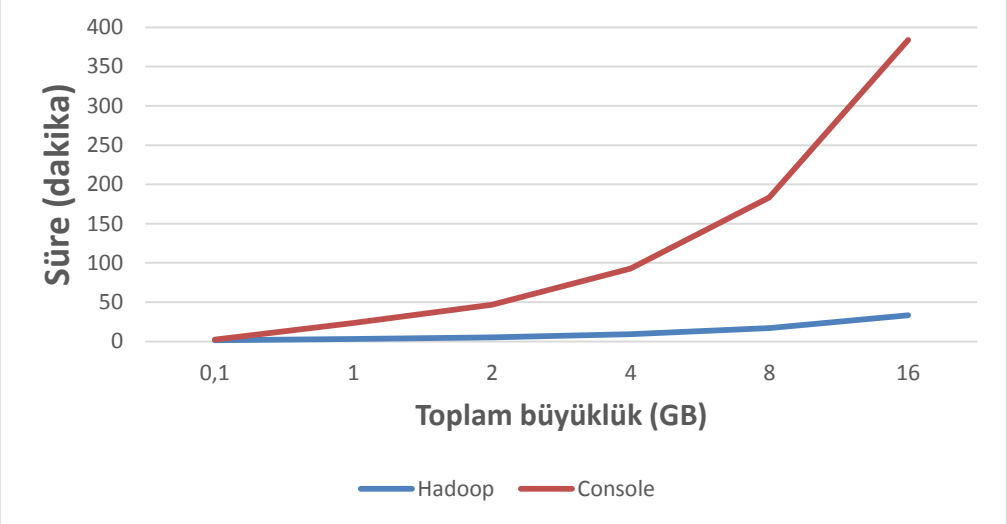
Çalışma süreleri elde edilirken, her bir test en az 10 kez yapılmıştır ve en büyük ve en küçük değerler ihmal edilip diğer sürelerin aritmetik ortalamaları alınmıştır. Bu yöntemle olabildiğince doğru süreler tespit edilmeye çalışılmıştır. Elde edilen çalışma süreleri Tablo 3.2 ve Tablo 3.3'te verilmiş ve Şekil 3.17'de grafik üzerinde gösterilmiştir.

Tablo 3.2. Konsol testi

Konsol testi (1 makine)	
Büyüklük (Gigabayt)	Çalışma süresi (dakika)
0,1	2,3
1	23,3
2	46,7
4	92,7
8	183,3
16	383,9

Tablo 3.3. Hadoop testi

Hadoop Testi (7 makineden oluşan küme)	
Büyüklük (Gigabayt)	Çalışma süresi (dakika)
0,1	1,33
1	3,35
2	5,26
4	9,1
8	17,16
16	33,3



Şekil 3.17. Hadoop ve Konsol performans karşılaştırması

Bu çalışmada Hadoop'un performansı giriş verisinin hangi büyüklükte parçalardan oluşuna göre de analiz edilmiştir. Her bir test önce tek bir dosyadan oluşan

giriş dosyasının HDFS'ye aktarılıp çalıştırılmıştır. Daha sonra aynı büyüklükteki giriş dosyası 100'er MB'lik parçalar halinde aktarılarak aynı iş için yeniden çalıştırılmıştır. Bu şekilde ve en az 10'ar kez tekrar edilerek tespit edilen süreler Tablo 3.4'te verilmiştir ve Hadoop'un giriş verisinin parçalardan oluşmasına göre gösterdiği performans Şekil 3.18'de verilmiştir. Görüldüğü üzere giriş verisinin tek parça yerine küçük parçalardan oluşması Hadoop'un çalışma süresini de arttırmaktadır. HIPI çalışması [44] da Hadoop'un çalışma süresinin giriş verisinin küçük parçalar halinde çok sayıda dosya olduğu durumda, tek bir büyük dosya olması durumuna göre çok daha fazla sürdüğü gösterilmiştir. Giriş dosyalarının Hadoop düğümlerine (node) dağıtılması işlemi zaman almaktadır ve bu zaman kaybı bazı durumlarda olması gerekenden 100 kat daha yavaş performans göstermesine neden olabilmektedir.

Tablo 3.4. Tek parçalı iş ile çok parçalı iş karşılaştırması

	Süre (Saniye)	
Veri boyutu	**Tek parça dosya**	**Çok parçalı dosya**
0,1	1,33	1,1
1	3,35	1,29
2	5,26	5,17
4	9,1	9,52
8	17,16	18,5
16	33,3	36,045

Şekil 3.18. Tek parçalı iş ile çok parçalı iş karşılaştırması

3.4. Naif Bayes İle Paralel Doküman Sınıflandırma

Naif Bayes sınıflandırıcı yöntemi kullanılarak farklı kategorilerdeki Türkçe haber metinlerinin sınıflandırılması yapılmıştır. Bu çalışma verilerin toplanması, ağırlık vektörlerinin oluşturulması, eğitim ve testler şeklinde gerçekleştirilmiştir. Belirtilen adımların tamamı otomatik olarak gerçekleştirilmiştir. Bu bölümdeki uygulamalar Apache Spark üzerinde Scala ve Java dillerinin birlikte kullanılmasıyla yapılmıştır.

3.4.1.Yöntem

a. Verilerin Toplanması

Farklı kategorilerden haberler web sitelerinden otomatik olarak indirilmiştir. Toplanan haberler başlık, kategori ve içerik olarak XML formatında kaydedilmiştir. Örnek bir haber dokümanı formatı aşağıda gösterilmiştir.

<item>
<date> tarih: saat **</date>**
<category> ekonomi **</category>**
<haber> "haber metni" **</haber>**
</item>

b. Ağırlık Vetörlerinin Oluşturulması

Bölüm 2.1. Metin sınıflandırma adımlarında vektör uzay modeli ve tf-idf ağırlıklandırma yöntemi açıklanmıştır. Öncelikle alınan bir dokümanın haber alanındaki metinde bulunan noktalama işaretleri, sayılar gibi anlam ifade etmeyen kelime öbekleri temizlenir. Kalan kelimelerin gövdeleri bulunarak değiştirilir. Bu aşamadan sonra tf-idf ağırlıkları hesaplanır ve dokümanları temsil eden vektörler oluşturularak veriler eğitim için hazır hale getirilir.

c. Verilerin Eğitilmesi

Dokümanların kategori bilgisi eğitim etiketi olarak kullanılır (<category> </category). Veriler tf-idf değerleri ve etiketten oluşan diziler haline getirilerek

istenilen sayıda dokümanın otomatik olarak eğitilmiştir. Farklı testlerle sınıflandırıcının başarısı ölçülmüştür. Farklı eğitim senaryoları gerçekleştirilmiştir. Eğitime katılan dokümanlarda en az yirmi adet kelime bulunmaktadır. Bu kritere sahip olmayan dokümanlar otomatik olarak silinmiştir.

3.4.2.Testler

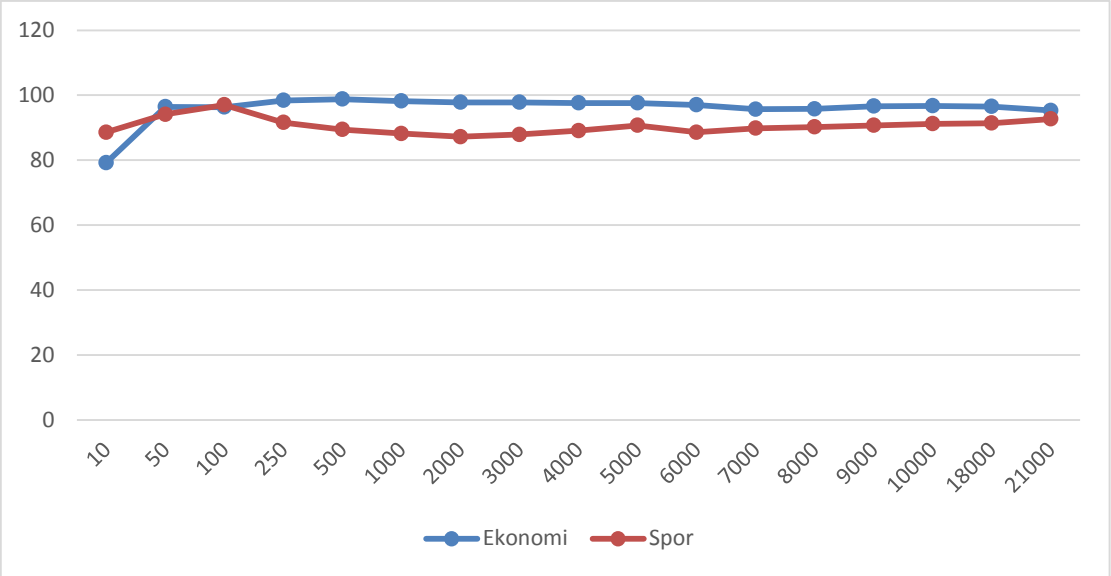
Ekonomi, spor, kültür, politika ve dünya kategorilerinden oluşan toplam beş farklı kategorideki veriler farklı senaryolarla eğitilip başarı ve performans testleri gerçekleştirilmiştir.

a. İki Kategori İle Sınıflandırma

Farklı sayılardaki ekonomi ve spor kategorilerinden haberler eğitilerek 1000'er adet ekonomi ve spor haberleriyle test edilmiştir. Grafikteki başarı oranları yüzde olarak verilmiş olup, bu başarı hesabı 1000 haberden yüzde kaçının doğru tespit edildiğini göstermektedir. Bu test yöntemi üç, dört ve beş kategori için yapılan testler için de geçerlidir.

Tablo 3.5. İki kategori ile sınıflandırma başarı tablosu

Kategori başına eğitim verisi sayısı	Ekonomi Başarı(%)	Spor Başarı(%)
10	79,2	88,6
50	96,4	94,1
100	96,3	97
250	98,4	91,6
500	98,8	89,4
1000	98,2	88,2
2000	97,8	87,2
3000	97,8	87,9
4000	97,6	89,1
5000	97,6	90,7
6000	97	88,6
7000	95,7	89,8
8000	95,8	90,2
9000	96,6	90,7
10000	96,7	91,2
18000	96,5	91,4
21000	95,3	92,7



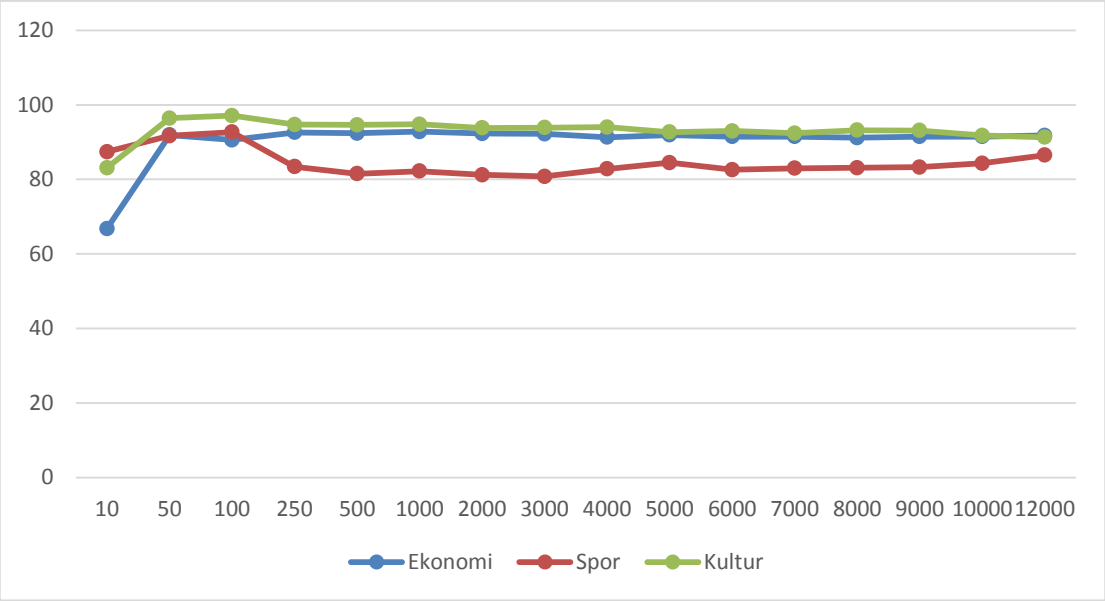
Şekil 3.19. İki kategori ile sınıflandırma başarı grafiği

b. Üç Kategori İle Sınıflandırma

Farklı sayılardaki ekonomi, spor ve kültür kategorilerinden haberler eğitilerek 1000'er adet ekonomi, spor ve kültür haberleriyle test edilmiştir.

Tablo 3.6. Üç kategori ile sınıflandırma başarı tablosu

Kategori başına eğitim verisi sayısı	Ekonomi Başarı(%)	Spor Başarı(%)	Kültür Başarı(%)
10	66,8	87,4	83,1
50	91,9	91,7	96,4
100	90,6	92,7	97,1
250	92,6	83,4	94,7
500	92,4	81,5	94,6
1000	92,8	82,2	94,8
2000	92,3	81,2	93,8
3000	92,2	80,8	93,9
4000	91,3	82,8	94
5000	91,9	84,5	92,7
6000	91,5	82,6	93
7000	91,5	83	92,4
8000	91,2	83,1	93,2
9000	91,5	83,3	93,1
10000	91,5	84,3	91,8
12000	91,8	86,5	91,3



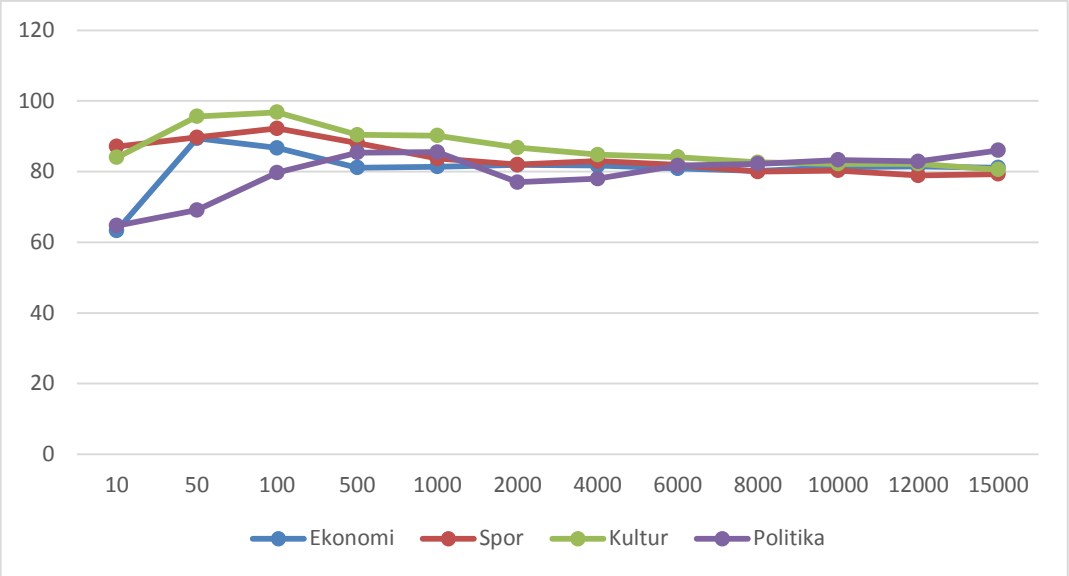
Şekil 3.20. Üç kategori ile sınıflandırma başarı grafiği

c. Dört Kategori İle Sınıflandırma

Farklı sayılarda ekonomi, spor, kültür ve politika haberleri eğitilerek 1000'er adet ekonomi, spor, kültür ve politika haberleriyle test edilmiştir.

Tablo 3.7. Dört kategori ile sınıflandırma başarı tablosu

Kategori başına eğitim verisi sayısı	Ekonomi Başarı(%)	Spor Başarı(%)	Kültür Başarı(%)	Politika Başarı(%)
10	63,3	87,1	84	64,7
50	89,4	89,7	95,6	69,1
100	86,7	92,2	96,8	79,7
500	81,1	88,1	90,4	85,4
1000	81,4	83,7	90,2	85,5
2000	81,9	82	86,8	77
4000	81,7	83	84,8	78
6000	80,9	81,9	84,1	81,7
8000	80,2	80	82,6	82,2
10000	81,3	80,3	82,2	83,3
12000	81,5	78,9	82,2	82,9
15000	81,1	79,3	80,6	86



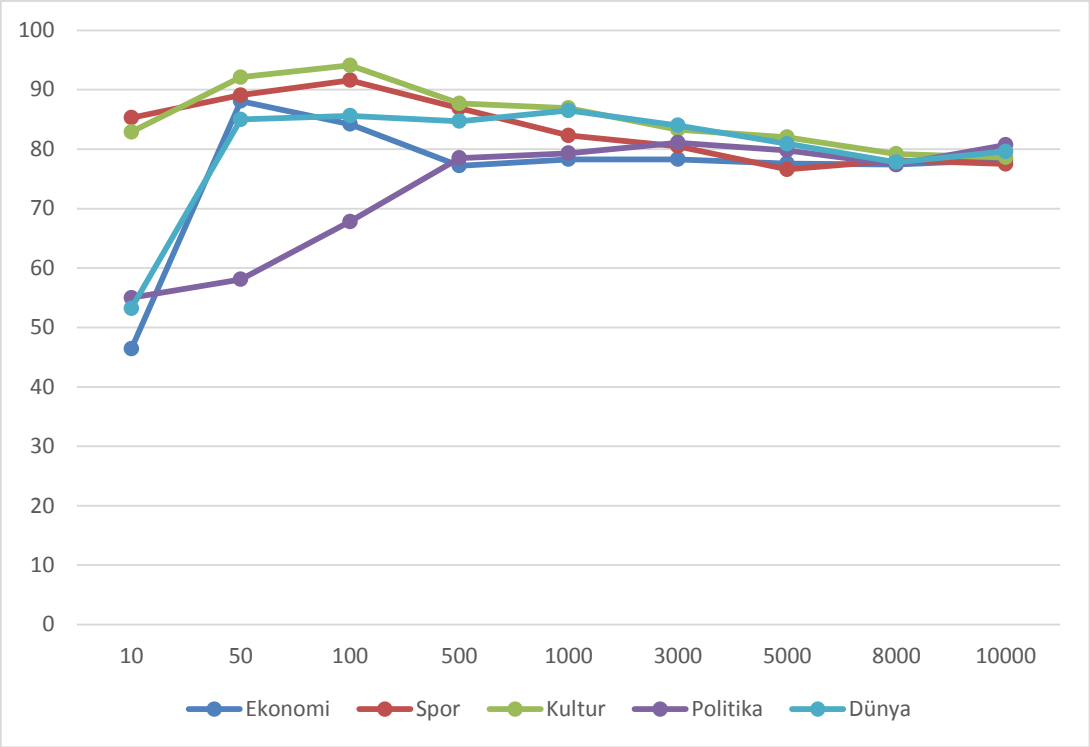
Şekil 3.21. Dört kategori ile sınıflandırma başarı grafiği

d. Beş Kategori İle Sınıflandırma

Farklı sayılarda ekonomi, spor, kültür, politika ve dünya haberleri eğitilerek 1000'er adet ekonomi, spor, kültür, politika ve dünya haberleriyle test edilmiştir.

Tablo 3.8. Beş kategori ile sınıflandırma başarı tablosu

Kategori başına eğitim verisi sayısı	Ekonomi Başarı(%)	Spor Başarı(%)	Kültür Başarı(%)	Politika Başarı(%)	Dünya Başarı(%)
10	46,4	85,3	82,9	55	53,2
50	88,1	89,1	92,1	58,1	85
100	84,2	91,6	94,1	67,8	85,6
500	77,2	86,9	87,7	78,5	84,7
1000	78,3	82,3	86,9	79,3	86,5
3000	78,3	80,5	83,3	81,1	84
5000	77,6	76,6	82	79,8	80,9
8000	77,4	78,1	79,2	77,4	77,8
10000	78,2	77,5	78,6	80,7	79,6



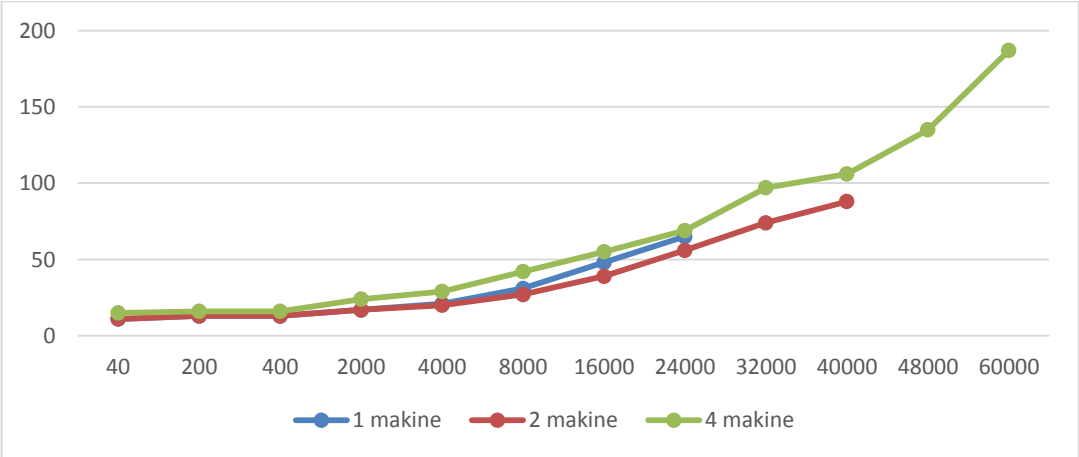
Şekil 3.22. Beş kategori ile sınıflandırma başarı grafiği

e. Performans Testleri

Ayrıca eğitim verisi miktarının ve kümedeki paralel iş yürüten bilgisayar sayısına göre performans testleri yapılmıştır. Tablo 3.9'da eğitime katılan toplam veri sayısı ve bu verilerin farklı sayıda makine ile eğitim süreleri verilmiştir. Veriler dört farklı kategoriden (ekonomi, spor, kültür, politika) eşit sayıda haberlerden oluşmaktadır. Belirli miktarın üzerindeki veri adetlerinde makine sayısının yetersiz olduğu ve Java'nın yetersiz hafıza hatası verdiği görülmüştür. Bu durumda makine sayısının arttırılması bir çözüm olabilmektedir. Diğer bir çözüm yolu ise dosyaların birleştirilip daha büyük boyutlu daha az sayıda giriş verisine dönüştürülmesidir.

Tablo 3.9. Farklı sayıda makine ile eğitim süreleri tablosu

Toplam eğitim verisi sayısı	1 makine Süre(sn)	2 makine Süre(sn)	4 makine Süre(sn)
40	11	11	15
200	13	13	16
400	13	13	16
2000	17	17	24
4000	21	20	29
8000	31	27	42
16000	48	39	55
24000	65	56	69
32000	*outOfMem*	74	97
40000	*outOfMem*	88	106
48000	*outOfMem*	*outOfMem*	135
60000	*outOfMem*	*outOfMem*	187



Şekil 3.23. Farklı sayıda makine ile eğitim süreleri grafiği

4. SONUÇ

Bu tez çalışmasında bulut bilişim konusu ele alınmış, büyük verilerin yönetimi ve büyük veriden otomatik olarak faydalı bilgilerin çıkartımı konusundaki yeni teknoloji ve yöntemler incelenmiştir. Bu kapsamda açık kaynaklı bulut yazımları incelenmiş, OpenStack'ın kurulumu ve yönetimi anlatılmıştır. Oluşturulan bulut sistemi üzerinde açık kaynaklı büyük veri işleme teknolojileri olan Hadoop ve Spark teknolojileriyle uygulamalar geliştirilmiştir. Büyük verilerden otomatik olarak faydalı bilgi çıkarımında kullanılan makine öğrenmesi teknikleri ve bu yöntemlerin dağıtık bir sistem üzerinde uygulanışı yine uygulamalar geliştirilerek incelenmiştir.

İnternet üzerinden otomatik bir şekilde ekonomi, spor, kültür, politika ve dünya kategorilerinden haberler toplanıp bu haberlerin yine otomatik bir şekilde sınıflandırılması Naive Bayes makine öğrenmesi algoritmasının paralel bir şekilde çalıştırılması ile gerçekleştirilmiştir. Yapılan sınıflandırma, hem kategorilerin doğru olarak tahmin edilmesi açısından başarı göstermiş hem de büyük miktarda verinin yüksek performanslı bir şekilde paralel olarak sınıflandırılabildiğini göstermiştir.

Makine öğrenmesi tekniklerinin incelenmesinin yanında, bu tekniklerinin uygulanması aşamasına kadar olan sürecin de paralel bir şekilde gerçekleştirilmesi incelenmiş ve bu amaçla log analizi, dağıtık doğal dil işleme çalışmaları yapılmıştır. Log analizi ile büyük miktarda log verisinin Hadoop ile analizi yapılmıştır. Birden fazla Hadoop işinin zincirleme olarak çalıştırılması gösterilmiştir. Aynı işlemler Spark ile gerçekleştirilip Hadoop ve Spark teknolojileri için performans karşılaştırılması yapılmıştır. Dağıtık doğal dil işleme uygulaması ile dağıtık dosya sistemi üzerindeki dokümanların makine öğrenmesi algoritmalarının uygulaması öncesindeki hazırlanma işlemi yapılmıştır. Yapılan işler paralel ve normal yöntemle yapılıp performansları karşılaştırılmıştır.

KAYNAKLAR

1. **Tanenbaum, A.S. and M.** Van Steen, Distributed systems: principles and paradigms. 2002. Cited in: p. 326.

2. **Moore, G.E.**, Cramming more components onto integrated circuits, 1965, McGraw-Hill New York, NY, USA.

3. **Olukotun, K. and Hammond L.,** The future of microprocessors. Queue, 2005. 3(7): p. 26-29.

4. **Duncan, R.** A survey of parallel computer architectures. Computer, 1990. 23(2): p. 5-16.

5. **Hennessy, J.L. and Patterson, D.A.** Computer architecture: a quantitative approach. 2012: Elsevier.

6. **Wu, C.-C., et al.,** Designing parallel loop self-scheduling schemes using the hybrid MPI and OpenMP programming model for multi-core grid systems. The Journal of Supercomputing, 2012. 59(1): p. 42-60.

7. **Foster, I. and C. Kesselman.** The Grid 2: Blueprint for a new computing infrastructure. 2003: Elsevier.

8. **Juve, G., et al.** Scientific workflow applications on Amazon EC2. in E-Science Workshops, 2009 5th IEEE International Conference on. IEEE. 2009.

9. **Hoffa, C., et al.** On the use of cloud computing for scientific workflows. in eScience, 2008. eScience'08. IEEE Fourth International Conference on. IEEE.2008.

10. **Mell, P. and T. Grance,** The NIST definition of cloud computing. 2011.

11. **Buyya, R., J. Broberg, and Goscinski, A.M.** Cloud computing: Principles and paradigms. John Wiley & Sons.Vol. 87. 2010:

12. **IAAS, PAAS, SAAS ve Windows Azure.** Available from : http://daron.yondem.com/tr/post/IAAS_PAAS_SAAS_ve_Windows_Azure

13. **IBM Blue Cloud.** Available from: http://www-03.ibm.com/press/us/en/pressrelease/22613.wss

14. **Amazon Web Services.** Available from: http://aws.amazon.com,http://aws.amazon.com.

15. **Nirvanix**. [cited 2014 03.07.2014]; Available from: http://www.nirvanix.com.

16. **Deelman, E., et al.** The cost of doing science on the cloud: the montage example. in Proceedings of the 2008 ACM/IEEE conference on Supercomputing. IEEE Press.2008.

17. **Gunarathne, T., et al.** MapReduce in the Clouds for Science. in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE.2010.

18. **Keahey, K., et al.** Science clouds: Early experiences in cloud computing for scientific applications. Cloud computing and applications, 2008. 2008: p. 825-830.

19. **Ostermann, S., et al.** A performance analysis of EC2 cloud computing services for scientific computing, in Cloud Computing. 2010, Springer. p. 115-131.

20. **OpenStack.** [cited 2014 03.07.2014]; Available from: http://www.openstack.org.

21. **OpenNebula.** Available from: http://www.opennebula.org.

22. **Nimbus.** [cited 2014 03.07.2014]; Available from: http://workspace.globus.org.

23. **Eucalyptus** [cited 2014 03.07.2014]; Available from: https://www.eucalyptus.com/eucalyptus-cloud/iaas.

24. **Science Clouds.** [cited 2014 03.07.2014]; Available from: https://portal.futuregrid.org.

25. **Dean, J. and Ghemawat S.** MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008. 51(1): p. 107-113.

26. Lecture Notes by Rob Schapire, Princeton University. [cited 2014 04.07.2014]; Available from: http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf.

27. **Sebastiani, F.** Machine learning in automated text categorization. ACM computing surveys (CSUR), 2002. 34(1): p. 1-47.

28. **Ikonomakis, M., Kotsiantis, S., and Tampakas, V.** Text classification using machine learning techniques. WSEAS Transactions on Computers, 2005. 4(8): p. 966-974.

29. **Zemberek.** [cited 2014 02.07.2014]; Available from: https://code.google.com/p/zemberek/.

30. **Bilski, A.** A review of artificial intelligence algorithms in document classification. International Journal of Electronics and Telecommunications, 2011. 57(3): p. 263-270.

31. Amasyali, M, Beken A. Measurement of Turkish word semantic similarity and text categorization application. In Signal Processing and Communications Applications Conference, IEEE 17th. 2009. IEEE. 2009. SIU 2009.

32. Güven, EN, Onur H, Sağıroğlu Ş. Yapay Sinir Ağları ile Web İçeriklerini Sınıflandırma. Bilgi Dünyası, 2008. 9(1): p. 158-178.

33. **Al-Shalabi, R., Kanaan G., and Gharaibeh M.** Arabic text categorization using kNN algorithm. in Proceedings of The 4th International Multiconference on Computer Science and Information Technology. 2006.

34. **Vapnik, V.** The nature of statistical learning theory. 2000: springer.

35. **Joachims, T.,** Text categorization with support vector machines: Learning with many relevant features. 1998: Springer.

36. **Von Laszewski, G., et al.** Design of the futuregrid experiment management framework. In Gateway Computing Environments Workshop (GCE), 2010. 2010. IEEE.

37. **Guide**. Available from: http://docs.openstack.org/.

38. Official Hadoop Web Site. [cited 2014 04.07.2014]; Available from: http://hadoop.apache.org/

39. **White, T.** Hadoop: the definitive guide: the definitive guide. 2009: " O'Reilly Media, Inc.".

40. Hadoop Wiki [cited 2014 04.07.2014]; Available from: http://wiki.apache.org/hadoop/PoweredBy.

41. **Bu, Y., et al.,** HaLoop: Efficient iterative data processing on large clusters. Proceedings of the VLDB Endowment, 2010. 3(1-2): p. 285-296.

42. **Lee, R., et al.** Ysmart: Yet another sql-to-mapreduce translator. in Distributed Computing Systems (ICDCS), 2011 31st International Conference on. 2011. IEEE.

43. **Zaharia, M., et al.** Spark: cluster computing with working sets. in Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. 2010.

44. **Sweeney, C., et al.,** HIPI: a Hadoop image processing interface for image-based mapreduce tasks. Chris. University of Virginia, 2011.

EKLER

Ek 1. Hadoop Kurulum Dosyaları Konfigürasyonu

Kurulum dizinini /usr/local/hadoop olarak belirlenmiştir. Konfigürasyon için aşağıdaki adımlar izlenir:

a) konfigürasyon dosyalarının olduğu bölüme geçilir

cd /usr/local/hadoop/conf

hadoop@master:/usr/local/hadoop/conf$

b) hadoop-env.sh

Bu dosyada JAVA_HOME sistemdeki Java kurulum yolu verilir.

```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0
```

c) conf/masters

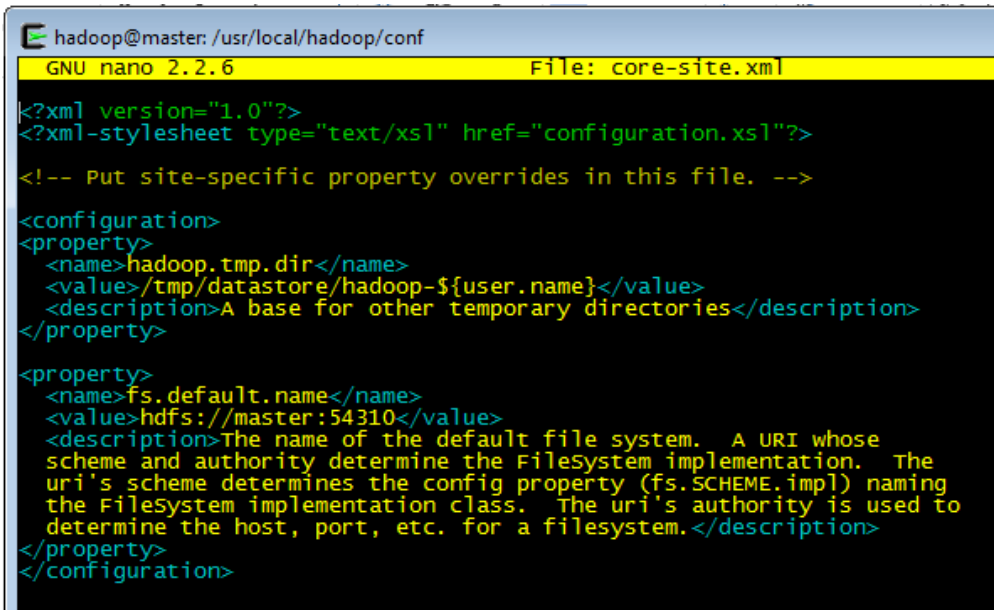
Master olarak görev yapan bilgisayarlar bu dosyada gösterilir. Bizim oluşturduğumuz küme için bu master isimli bilgisayardır.

d) conf/slaves

Slave olarak görev yapan bilgisayarlar bu dosyada gösterilir. Bizim oluşturduğumuz küme için bu master,slave1,slave2 ve slave3'tür. Master aynı zamanda slave olarak da kullanılabilir.

e) conf/core-site.xml

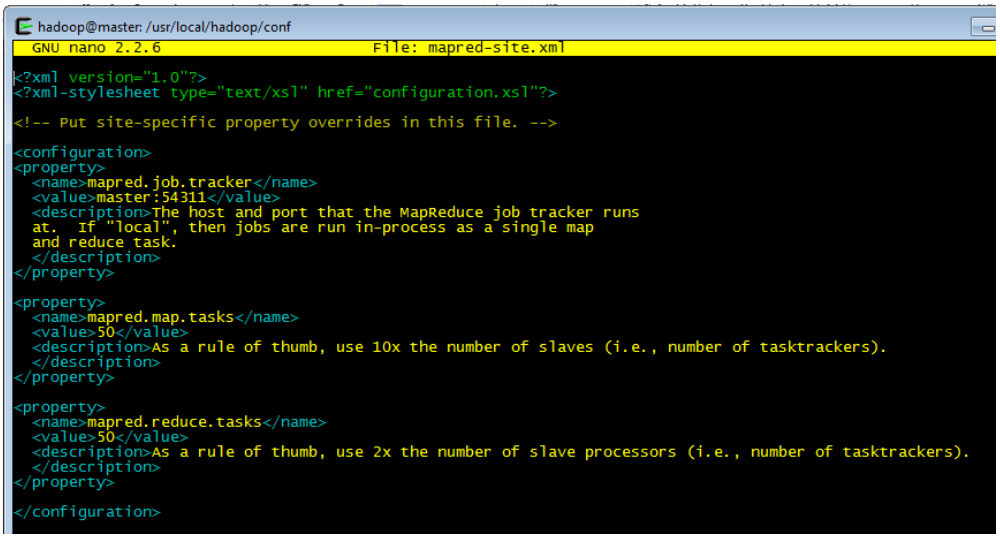
Bu dosyadaki konfigürasyonlar Ek Şekil 1.1 'deki gibidir.



Ek Şekil 1.1. conf/core-site.xml

f) conf/mapred-site.xml

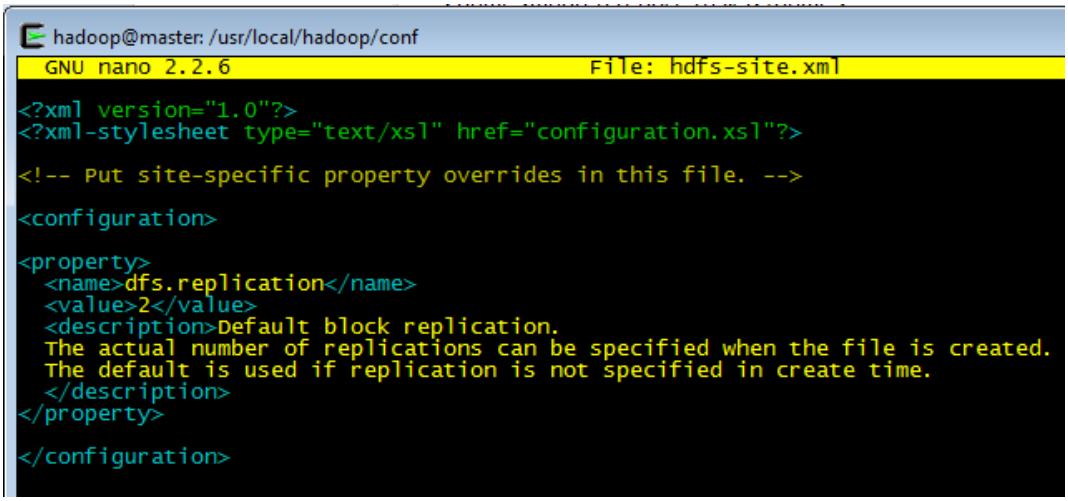
Bu dosyadaki konfigürasyonlar Ek Şekil 1.2'deki gibidir.



Ek Şekil 1.2. conf/mapred-site.xml

g) conf/hdfs-site.xml

Bu dosyadaki konfigürasyonlar Ek Şekil 1.3 'teki gibidir.



Ek Şekil 1.3. conf/hdfs-site.xml

Ek 2. Hadoop WordCount Java Programı Kodları

WordCount programının kodları aşağıda verilmiştir.

```
public class WordCount {

  public static class Map extends MapReduceBase implements
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, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
      }
    }
  }
    public static class Reduce extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
```

```
        while (values.hasNext()) {
          sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
      }
    }
      public static void main(String[] args) throws Exception {
     JobConf conf = new JobConf(WordCount.class);
     conf.setJobName("wordcount");

     conf.setOutputKeyClass(Text.class);
     conf.setOutputValueClass(IntWritable.class);

     conf.setMapperClass(Map.class);
     conf.setCombinerClass(Reduce.class);
     conf.setReducerClass(Reduce.class);

     conf.setInputFormat(TextInputFormat.class);
     conf.setOutputFormat(TextOutputFormat.class);

     FileInputFormat.setInputPaths(conf, new Path(args[0]));
     FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

ÖZGEÇMİŞ

Ad Soyad:

İbrahim Rıza HALLAÇ

Eğitim Durumu:

Lisans - Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü

Yüksek Lisans - Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü Kuramsal Temeller ABD

İş Deneyimi:

Araştırma Görevlisi, Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü

Yonca CBS, Ar-Ge Mühendisi

Sahip Olunan Beceriler:

MapReduce, Spark, Shark

Postgres SQL DB, MySQL, and PostGis

Java, JSP, Servlets

C/C++/Java/C#/Bash scripting

JavaScript; OpenLayers and ExtJS Libraries

Ajax, JSON

Mesleki Sertifika ve Eğitimler:

1.CCNA-1

2. Design and Practice of Embedded Systems, Bialystok Technical University, Poland

3. IBM DB2 Pure XML Eğitimi (IBM Software Academy)

4. Web Service Development, Turkcell Akademi