T.C
KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

BÜYÜK VERİDE HADOOP VE MAPREDUCE UYGULANMASI VE HDFS'YE ALTERNATİF DOSYA SİSTEMİ GELİŞTİRİLMESİ

SALİHA GÜLSEN KESKİN

AĞUSTOS 2018

Bilgisayar Mühendisliği Anabilim Dalında Saliha Gülsen KESKİN tarafından hazırlanan BÜYÜK VERİDE HADOOP VE MAPREDUCE UYGULANMASI VE HDFS'YE ALTERNATİF DOSYA SİSTEMİ GELİŞTİRİLMESİ adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Prof. Dr. Hasan ERBAY
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Dr.Öğr.Üyesi Atilla ERGÜZEN
Danışman

Jüri Üyeleri

Başkan : Prof. Dr. Mehmet Reşit TOLUN _____
Üye (Danışman) : Dr. Öğr. Üyesi Atilla ERGÜZEN _____
Üye : Dr. Öğr. Üyesi Erdal ERDAL _____

……/…../…….

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Prof. Dr. Mustafa YİĞİTOĞLU
Fen Bilimleri Enstitüsü Müdürü

Sevgili Aileme..

ÖZET

BÜYÜK VERİDE HADOOP VE MAPREDUCE UYGULANMASI
VE HDFS'YE ALTERNATİF DOSYA SİSTEMİ GELİŞTİRİLMESİ

KESKİN, Saliha Gülsen
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans tezi
Danışman: Dr. Öğr. Üyesi Atilla ERGÜZEN
Ağustos 2018, 105 sayfa

Şirketlerin ve bireylerin günlük faaliyetlerinin çoğunda kullandığı akıllı telefonlar, bilgisayarlar, giyilebilir teknoloji ürünleri, ev aletleri, sensörler gibi birçok cihazın yaygın kullanımıyla sağlık, hükümet, sosyal ağlar, pazarlama, finansal gibi çeşitli kaynaklardan daha önce görülmemiş oranda her an veri üretilmektedir. Verinin hacmi, karmaşıklığı ve akış hızı, geleneksel veritabanı yazılım araçları ve teknolojileri kullanılarak yakalanmasını, yönetilmesini, işlenmesini veya analiz edilmesini zorlaştıran metin, video, resim, ses, web sayfası günlük dosyaları, bloglar, tweetler, konum bilgileri, sensör verileri gibi yapılandırılmamış, yarı yapılandırılmış ve yapılandırılmış veri kümeleri büyük veri olarak adlandırılmaktadır. Bu tür büyük veri kümelerinden yararlı bilgiler elde etmek için akıllı ve ölçeklenebilir analiz hizmetleri, programlama araçları ve uygulamaları kullanmak gerekmektedir. Apache Hadoop çeşitli sunucu düğümleri arasında büyük veri kümelerinin güvenilir bir şekilde saklanmasını, yönetilmesini, dağıtılmasını ve paylaşılmasını sağlayan ölçeklenebilir ve hataya dayanıklı bir yazılım çerçevesidir. Hadoop, dağıtılmış dosyalarda çok miktarda veri depolamayı sağlayan bir dosya sistemine, Hadoop Dağıtık Dosya Sistemine (HDFS) sahiptir ve iş yükünü ve depolamayı paralel hale getirerek hesaplamaları gerçekleştirmek ve büyük miktarda veri işlemek için MapReduce programlama modelini kullanmaktadır. Ayrıca hem bireylerin hem de kuruluşların büyük verileri kullanmaları için, herhangi bir veri depolama ve işleme tesislerine yatırım yapmayı ortadan kaldıran bulut bilişim teknolojisi de

geliştirilmiştir. Bulut bilişim ihtiyaca göre ölçeklenebilen, anında kullanıma hazır hizmet sağlayıcılardan oluşan internet ortamını ifade etmektedir. Verilerin internet ortamında saklanması ve internet vasıtasıyla işlenmesi çeşitli güvenlik ve gizlilik sorunlarını gündeme getirmektedir.

Bu tez kapsamında öncelikle büyük veri kavramı ayrıntılı olarak incelenmiştir. Daha sonra bir ana, üç bağımlı düğüm olarak dört düğüm ile bir hadoop kümesi oluşturulmuş ve kümenin performansı test edilmiştir. Düğüm, çoğaltma (replica), harita (map) ve indirgeme (reduce) sayısında, girdi dosyalarının ve HDFS blok boyutunda değişiklikler uygulanarak çeşitli testler yapılmıştır. Sonuç olarak, hadoopun büyük dosyaları işlemek için tasarlanan ve çok sayıda küçük dosya ile çalışırken performans sıkıntısı çeken, ayrıca kümeyi yöneten tek bir ana düğümle çalışan dosya sistemine sahip olduğu gözlemlenmiştir. Son olarak, buna alternatif bir sistem önerilmiştir. Bu sistem ile şifreleme teknikleri kullanılarak güvenlik sıkıntısı çözülmüş ve dosya yapılandırılmasında birden fazla blok boyutu kullanılarak küçük ve büyük dosyaların depolanması kolaylaştırılmıştır. Ayrıca sistem tek bir ana düğüme bağlı olmayıp birden fazla sunucu ile çalışmakta ve düğümsel problemi ortadan kaldırmaktadır.

Anahtar Kelimeler: Büyük Veri, Hadoop, Hadoop Dağıtılmış Dosya Sistemi, MapReduce, Bulut Bilişim, Şifreleme

ABSTRACT

IMPLEMENTATION OF HADOOP AND MAPREDUCE IN BIG DATA AND ALTERNATIVE FILE SYSTEM DEVELOPMENT FOR HDFS

KESKİN, Saliha Gülsen
Kırıkkale University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering, M.Sc. Thesis
Supervisor: Asst. Prof. Dr. Atilla ERGÜZEN
August 2018, 105 pages

With the widespread use of many devices such as smart phones, computers, wearable technology products, home appliances, sensors, etc., which are widely used by companies and individuals in daily activities, data is being produced at any time in places like health, government, social networks, marketing and financial. Unstructured, semi-structured and structured data sets, such as text, video, picture, audio, web page log files, blogs, tweets, location information, sensor data, which make it difficult to capture, manage, process or analyze using traditional database software tools and technologies due to volume, complexity and flow rate of the data, are called Big Data. To obtain useful information from such large data sets, it is necessary to use intelligent and scalable analysis services, programming tools and applications. Apache Hadoop is a scalable and fault tolerant software framework that enables reliable storage, management, distribution, and sharing of large data sets across various server nodes. Hadoop has the Hadoop Distributed File System (HDFS), a file system that allows you to store large amounts of data in distributed files, and uses the MapReduce programming model to perform calculations by making workload and storage parallel and to process large amounts of data. In addition, cloud computing technology has been developed that removes investment in any data storage and processing facilities for both individuals and organizations to use large amounts of data. Cloud computing refers to an internet environment

consisting of ready-to-use service providers that can be scaled according to their needs. The storage of data on the internet and the processing by means of internet brings various security and confidentiality problems.

In this thesis firstly the concept of big data is examined in detail. Then, a hadoop cluster with four nodes as one main (master) and three dependent nodes (slaves) was created and the performance of the cluster was tested. Various tests have been done by applying changes in the node, replica, map and reduce numbers, and in different sizes with input files and HDFS block size. As a result, Hadoop has been observed to have a file system that is designed to handle large files and has performance bottlenecks when working with many small files, and also running on a single main node that manages the cluster. Finally, an alternative system has been proposed. With this system, the security problem is solved by using encryption techniques and it is facilitated to store small and large files by using more than one block size in file configuration. In addition, the system is not connected to a single master node but works with multiple servers and thus removes the node problem.

Key Words: Big Data, Hadoop, Hadoop Distributed File System, MapReduce, Cloud Computing, Cryptography

TEŞEKKÜR

Tez çalışmamda planlanmasında ve yürütülmesinde ilgi ve desteğini esirgemeyen, lisans ve yüksek lisans eğitimim boyunca bilgi ve tecrübeleri ile yönlendiren danışman hocam Sayın Dr. Öğr. Üyesi Atilla ERGÜZEN'e, sadece akademik olarak değil her konuda destek olan, hiçbir yardımı esirgemeyen değerli hocam Dr. Öğr. Üyesi Erdal ERDAL'a ve maddi manevi desteğiyle her zaman yanımda olan aileme çok teşekkür ederim.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİL Sayfa

ÇİZELGELER DİZİNİ

ÇİZELGE

KISALTMALAR DİZİNİ

DBMS	Database Management System
DFS	Distributed File System
EB	ExaByte (10^{18} bytes)
G / Ç	Girdi / Çıktı
GB	GigaByte (10^{9} bytes)
GFS	Google File System
HDFS	Hadoop Distributed File System
IAAS	Infrastructure as a Service
IBM	International Business Machines Corporation
ICT	Information And Communications Technology
IDC	International Data Corporation
IOT	Internet of Things
KB	KiloByte
KVM	Kernel-based Virtual Machine
MB	MegaByte
NoSQL	Not Only SQL
PAAS	Platform as a Service
PB	PetaByte (10^{15} bytes)
RDBMS	Relational Database Management System
RDF	Resource Description Framework
SAAS	Software as a Service
SPOF	Single Point of Failure
TB	TeraByte (10^{12} bytes)
TCP	Transmission Control Protocol
VM	Virtual Machine
ZB	ZettaByte (10^{21} bytes)

1. GİRİŞ

Dijital teknolojinin ve akıllı cihazların ortaya çıkmasıyla birlikte, her gün çok miktarda veri üretilmektedir. E-posta, nesnelerin interneti (Internet of Things, IoT), web sayfalarında gezinti, çevrimiçi (online) alışveriş, telefon görüşmeleri, Facebook gibi sosyal medya platformlarında paylaşımlar, Google arama motoru, sohbet (chat), çevrimiçi randevu, ödeme, kablolu yayın, akıllı telefon, oyun, resim, video ve daha birçok şekilde veri elde edilmektedir [1][2].

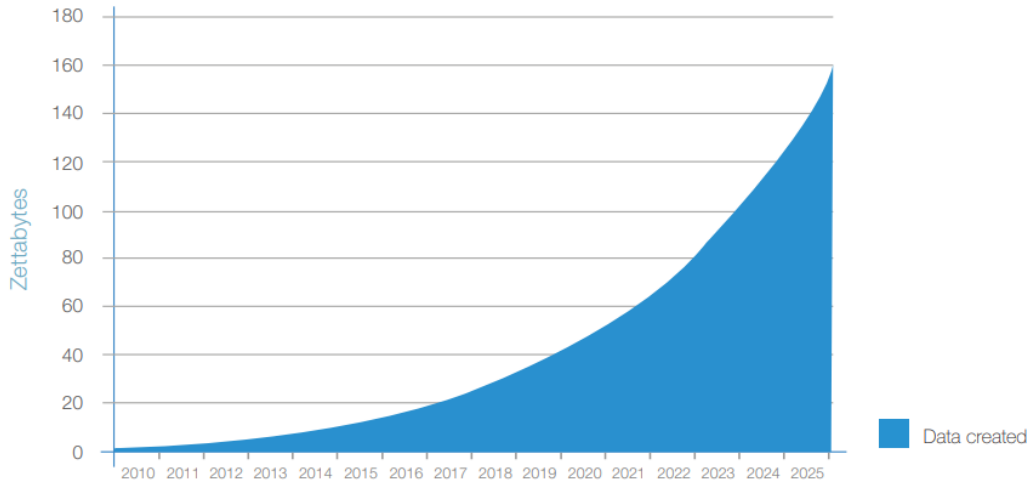
IBM'ye göre otuz yıl öncesinde, 1980'lerde en büyük sabit diskler yaklaşık 10 MB kapasiteye sahipken, bu zamandan itibaren insan başına üretilen veri her kırk ayda iki katına çıkmıştır. 2012'de IBM, "*İnsanlık tarihi boyunca toplanan verinin % 90'ı sadece son iki yılda üretilmiştir ve her gün 2,5 kentilyon (2,5 exabyte) veri üretilmektedir."* ifadesini kullanmıştır [3].

Günümüzde teknolojinin gelişmesiyle nesnelerin interneti, drone'lar, yapay zeka, finansal teknoloji, ses ve görüntü tanıma teknolojisi, giyilebilir teknoloji derken veri artış hızında patlama olmuştur ve her geçen gün bu hız katlanarak artmaktadır.

IDC'ye göre 2025 yılına gelindiğinde, dünyanın herhangi bir yerindeki kişi, günde yaklaşık 4800 kez mobil cihazlarla etkileşim kuracaktır - ki bu da her 18 saniyede bir etkileşim demektir- [3].

IDC'nin yaptığı incelemelere göre, Şekil 1.1'de görüldüğü gibi küresel veri kümesinin 2025 yılına kadar 163 ZettaByte'a (trilyonlarca GB) ulaşması beklenmektedir. Bu, 2016 yılında üretilen 16,1 ZB verilerin on katıdır [3]. Şu anda küresel veri havuzunda yaklaşık olarak 33 ZB veri bulunmaktadır.

Küresel veri havuzunda toplanan verilerin dörtte birinden fazlası gerçek zamanlı veri olup ve bunun % 95'inden fazlasını IoT, makine öğrenmesi, bilgi ve dil işleme, yapay zekâ gibi bilişsel bilim olarak bilinen yeni bir dizi teknolojiden elde edilen veriler oluşturmaktadır [3].



Şekil 1.1 Küresel Veri Kümesinin Yıllık Boyutu [3].

IDC, 2025 yılına kadar, küresel veri havuzundaki verilerin % 20'sinin günlük hayatımız için kritik olacağını ve bunun ise sadece % 10'unun hiper-kritik (insan sağlığında ve refahında doğrudan ve anında etkisi olan veriler - tıbbi veriler gibi) olacağını tahmin etmektedir [3].

Çalışmanın bu bölümünde veri, veri türleri, veri güvenliği, büyük veri kavramı ve büyük verinin özellikleri, kullanıldığı ortamlar hadoop ve bulut bilişim, paralel olarak veri işlemeyi sağlayan MapReduce programlama modeli, verilerin saklanmasını sağlayan HDFS ve NoSQL hakkında bilgiler verilmiştir.

Çalışmanın ikinci bölümünde, büyük veri ile ilgili son 5 yıldaki literatür taraması yapılmıştır. Üçüncü bölümünde, kurulan hadoop kümesi ve performansını ölçmek için kümenin üzerinde uygulanan MapReduce testleri yer almaktadır. Ayrıca hadoopun dosya sistemine alternatif olarak geliştirdiğimiz sistemin mimarisi ve yapısı anlatılmıştır. Dördüncü ve sonuç bölümünde ise yapılan uygulamalardan elde edilen bilgi ve değerlendirmelere yer verilmiştir.

1.1. Dijital Çağda Veriler

Veriler bir amaç için (genellikle analiz için) toplanan ve dönüştürülen karakter kümesidir. Metin, resim, ses, video dahil olmak üzere elektrik sinyalleri şeklinde saklanıp iletilebilen ve manyetik, optik veya mekanik kayıt ortamına kaydedilebilen karakterler veya semboller olabilir. Veriler toplanıp analiz edilerek; bir şekilde karar vermek için uygun bilgiler haline getirilmektedir [4].

Çok miktarda verinin üretimi internet vasıtasıyla sağlanmaktadır. Günümüzde, daha uygun fiyatlı akıllı telefonlar ve mobil veri paketleri internet kullanıcılarının artmasında büyük katkı sağlamaktadır. 200 milyondan fazla nüfus geçtiğimiz sene ilk mobil cihazını almış ve günümüzde 7,6 milyar olan dünya nüfusunun bugün üçte ikisi bir cep telefonuna sahiptir. Kullanılan bu cihazların yarısından çoğu akıllı telefondur ve bu sayede, insanlar oldukları herhangi bir yerden zengin bir internet deneyiminin tadını çıkarabilmektedir [5].

Sosyal medya kullanımı da hızla artmaya devam etmekte ve her ülkede bu sosyal medya platformlarını kullanan kişilerin sayısı her geçen gün yaklaşık 1 milyon artmaktadır. Dünya nüfusunun yarısından çoğu artık sosyal medyayı kullanmaktadır. Seçtiği sosyal medya platformuna her on kişiden biri mobil cihaz ile erişim sağlamaktadır[5].

Aşağıda 2018 yılının raporlarından elde edilen bazı istatistiksel bilgiler bulunmaktadır:

- İnternet kullanan kişi sayısı, yıllık % 7 artarak 4,021 milyar,
- Sosyal medya kullanan kişi sayısı, yıllık % 13 artarak 3,196 milyar,
- Cep telefonu kullanan kişi sayısı yıllık % 4 artarak 5,135 milyar olmuştur [5].

İnternetin kullanım oranları o kadar büyüktür ki, oranlara aylık olarak bakmak hatta günlük rakamları kullanmak bile mantıklı değildir. Bunun yerine, yalnızca tek bir dakikada internette ne olduğuna bakılmıştır ve elde edilen bilgilere göre;

- 3,5 milyon Google araması yapılmakta,
- 900000 Facebook girişi yapılmakta,

- 16 milyon metin mesajı oluşturulmakta,
- 4,1 milyon Youtube video izlenmekte,
- 452000 tweet gönderilmekte,
- 156 milyon mail gönderilmekte,
- 1,8 milyon Snap (anlık fotoğraf) oluşturulmaktadır [6].

İnternet, verinin ortaya çıkmasında ve gelişmesinde büyük etkendir. Ancak verinin tek kaynağı internet değildir. Günümüzde firmalar, sensörler, bilgisayarlar ve otomatik kontrol cihazları tarafından da sürekli olarak veri üretimi yapılmaktadır. Ayrıca akıllı telefonlar ve diğer mobil cihazlar aracılığıyla da çok miktarda veri üretimi sağlanmaktadır.

1.1.1. Veri Türleri

Büyük veriler çağrı merkezi ses verisinden biyolojik araştırma ve ilaca ait genomik ve proteomik verilere kadar her şeyi kapsamaktadır. Her gün yalnızca Google, 24 PB (veya 24000 TB) veri işlemektedir. Ancak bu verilerin çok azı geleneksel veritabanlarında düzgün bir biçimde satır ve sütunlarda saklanabilmektedir [1].

Elde edilen bu veriler yapılandırılmış, yarı yapılandırılmış ya da yapılandırılmamış olarak üç farklı türde karşımıza çıkmaktadır [7].

1.1.1.1. Yapılandırılmış Veriler

Yapılandırılmış veriler, standart bir veritabanı içindeki satır ve sütunlara uyan geleneksel verilerdir. İlişkisel veri tabanlarında belli bir yapıda (formatta) gruplandırılmışlardır. Veritabanlarında kolayca saklanır ve verileri veritabanlarından almak için çok çeşitli araçlar mevcuttur [8].

Yapılandırılmış veriler ilişkisel tablolarda düzgün bir şekilde tutulması nedeniyle bu verilere basit sorgularla erişim mümkündür. Ayrıca, kontrol etmek daha kolaydır [9].

1.1.1.2. Yapılandırılmamış Veriler

Yapılandırılmamış veriler, düzgün bir şekilde satır ve sütunlara sığmayan, ilişkisel tablolara kolayca endekslenemeyen verilerdir. E-postalar, PDF belgeleri, sensör beslemeleri, görüntüler, ses ve video gibi formatlarda bulunabilir. Yapılandırılmamış verilerin sınıflandırılması daha zordur ve bu verileri işlemek için mevcut ticari araçlar yetersizdir [8].

Yapılandırılmamış veriler, günümüzde daha fazla üretilmekte ve kurumlarda kullanımı daha yaygın hale gelmektedir. Bu tür veriler, geleneksel veri yapıları ile sınırlı değildir. Daha büyük boyuttaki ve çeşitli türdeki bu veriler, çoğu zaman dış veri kaynaklarından elde edilmektedir. Gelişmekte olan ve öngörülemez doğası nedeniyle yapılandırılmamış verileri, genellikle yönetmek daha zordur. Sonuç olarak, bu tipteki verileri yönetmek ve analiz etmek için yeni çözümler geliştirilmesine ihtiyaç vardır [9].

1.1.1.3. Yarı-Yapılandırılmış Veriler

Yarı-yapılandırılmış veriler, yapılandırılmış veriler gibi geleneksel veritabanına uygun biçimde değildir, ancak işlenmesini kolaylaştıran bazı kurumsal özellikler içeren verilerdir [9].

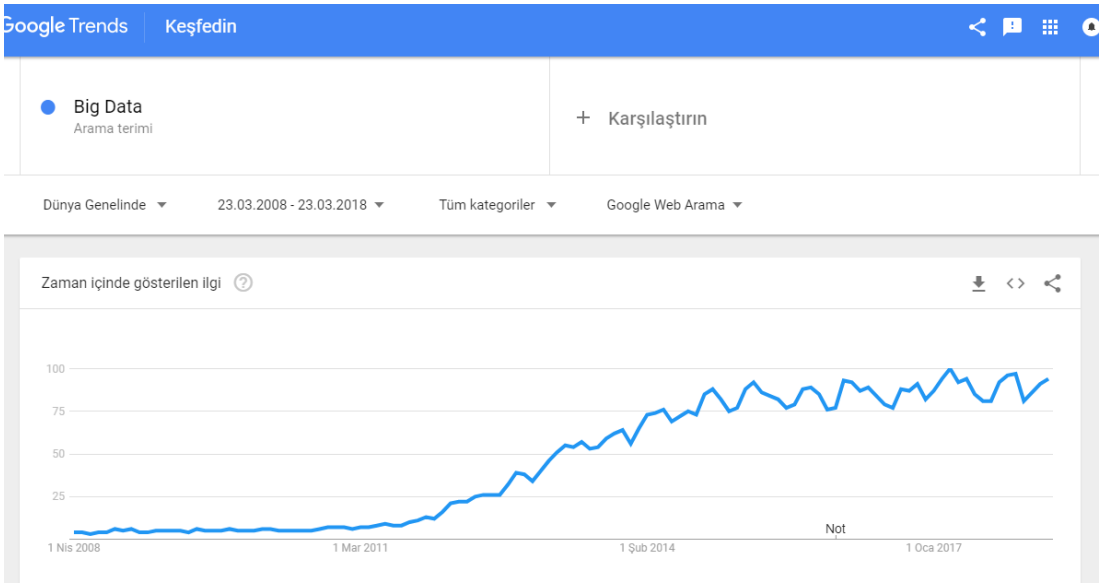
Yarı-yapılandırılmış veriler, net ve sabit bir şemaya uymayan veri şeklidir. Bir şemanın önceden tanımlanmasını gerektirmez ve sürekli olarak gelişebilir; yani, yeni özellikler herhangi bir zamanda eklenebilir. Ayrıca, aynı sınıftaki veriler, birlikte gruplandırılsalar bile farklı özniteliklere sahip olabilirler ve özniteliklerin sırası önemli değildir. XML ve JSON verileri, web günlükleri ve sosyal medya yayınları gibi verilerdir [8].

1.2. Büyük Veri

Büyük veri konusuyla ilgili ilk bilim araştırması 1974'te yapılmıştır, ancak bu alandaki araştırmaların kapsamı sadece son on yılda, 2008'den bu yana, hızla artmaktadır. Bu terim genellikle bilgisayar bilimleriyle ilişkili olsa da, veriler sağlık, mühendislik, sanat, toplum ve çevre gibi birçok farklı alanda kullanılmaktadır [10].

2005 yılında O'Reilly Media'dan Roger Magoulas'ın, büyük veri yarışını başlatan "Web 2.0 Nedir?" adlı makalesinde ilk defa "Büyük Veri (Big Data)" terimi kullanılmıştır. Magoulas, "*Geleneksel iş zekası araçlarını kullanarak yönetme ve işlemenin neredeyse imkansız olduğu büyük bir veri kümesi"* şeklinde bu kavrama yer vermiştir [11][12].

Günümüzde büyük veri, popüler bir kavram haline gelmiştir. 2006 yılında, arama motorunda ne kadar arama yapıldığı konusunda değerli bilgileri sağlayan Google Trends uygulaması sunulmuştur. Google Trends uygulaması kullanılarak son 10 yılda "Büyük Veri (Big Data)" terimine olan ilgi Şekil 1.2'deki grafikte gösterilmektedir [13].



Şekil 1.2. Büyük Veri Kavramına Zaman İçerisinde Gösterilen ilgi [13].

Arama sonucuna göre bu kavramın kullanımında özellikle 2011 yılından sonra büyük artış görülmektedir. 2017 yılında ise terimin en yüksek popülerliğe sahip olduğu görülmektedir. (Şekilde 100 değeri, terimin en yüksek popülerliğini; 50 değeri ise, terimin bunun yarısı kadar popüler olduğu anlamına gelmektedir.)

1.2.1. Büyük Veri Nedir?

Büyük veri, terabyte'lardan petabyte'lara (ve hatta exabyte'lara) kadar dayanan, geleneksel veritabanı sistemlerinin işlem kapasitesini aşan veri kümelerini ifade etmektedir. Veriler çok büyüktür, çok hızlı hareket etmekte ve mevcut veritabanı mimarilerinin yapılarına uymamaktadır [14].

Büyük veri, yeni teknolojilerin hızla ilerlemesiyle farklı türde büyük miktardaki verilerin toplanması ve depolanması için geliştirilen yeni yöntemlerin bir sonucudur. Büyük veri ile geleneksel veritabanlarına ve veri uygulamalarına alternatif bir çözüm amaçlanmıştır. Sadece veri depolama veya veriye erişim değil, verileri anlamak ve değerlerinden yararlanmak için verileri analiz etmek de amaçlanmaktadır. Veriyi işlemek ve bunlardan değer elde etmek için yöntemler geliştirip mümkün olduğu kadar faydalı bilgiler de çıkararak daha önemli hale gelmektedir [15].

Büyük veri kavramı, şirketler arasında önemli bir mevzu haline gelmiştir. Bu kavram, şirketler tarafından müşterilerini anlamak, onlara yakın olmak ve onların eğilimlerini tahmin etmek için bir araç olarak görülmektedir [16].

Geleneksel veri işleme teknolojileri, büyük verileri etkili bir şekilde işleme kabiliyetleri sağlamadıkları için, büyük veri birçok farklı endüstride birçok kurum için sorun haline gelmiştir. Büyük veriler, kuruluşların verilerden daha verimli bilgi almalarına ve daha iyi bir değer sunmalarına yardımcı olarak daha iyi yönetim için büyük verileri hazırlama ve saklama yöntemlerini ele almaktadır. Ayrıca, kâr amaçlı olarak büyük verilerin doğru bir şekilde yakalanması ve gerçek zamanlı olarak incelenmesi için kuruluşlar bu teknolojilere yönelmektedir [1].

Bu kavram çeşitli sektörlerde güncel bir konudur ve önümüzdeki yıllarda kavramın popülaritesinin daha da artması beklenmektedir.

1.2.2. Büyük Verinin Karakteristik Özellikleri

2000'li yılların başlarında e-ticaretin yükselişiyle şirketlerin karşı karşıya kaldığı veri yönetimi zorluklarını sunarken Laney, veri hacmi, hızı ve çeşitliliğinde 3 boyutlu artışı ifade eden bir çerçeve sunmuştur. Laney'in çalışması [17], büyük veriden açıkça bahsetmemesine rağmen, daha sonra "3V" olarak adlandırılan model, büyük veri kavramına bağlanmış ve tanımı olarak kullanılmıştır.

1.2.2.1. Hacim

Hacim, büyük veri sisteminin ele aldığı veri kümelerinin boyutunu açıklamaktadır [16]. Sensör ve cihaz veri beslemeleri, multimedya verileri, kurumsal verilerin hepsi dijital verilerde hacmin büyümesine katkıda bulunmaktadır [8].

Hacim göreceli bir terimdir; bazı küçük ölçekli kuruluşlar, büyük küresel işletmelerin sahip olduğu petabyte veya exabyte verilerin aksine yalnızca gigabyte veya terabyte veri kapasitesine sahiptir. Kuruluşun boyutuna bakılmaksızın veri hacmi büyümeye devam etmektedir. Şirketlerin her türden veri depolaması için yeni teknolojiye doğru bir eğilim vardır: finansal veriler, tıbbi veriler, çevresel veriler vb. bu şirketlerin veri kümelerinin birçoğu bugün terabyte aralığındadır ancak yakında petabyte'lara ve hatta exabyte'lara ulaşabilirler [8][18].

1.2.2.2. Çeşitlilik

Veriler çeşitli kaynaklardan çeşitli tiplerde toplanmaktadır. Veri kümeleri sadece yapısal olan geleneksel veriler değil, aynı zamanda yarı yapılandırılmış ve yapısal olmayan verileri de içermektedir [18]. Videolar, resimler, metin, ses vb. gibi veriler

sensörler, akıllı telefonlar veya sosyal ağlar aracılığıyla toplanan farklı veri türlerini oluşturmaktadır.

Verinin çeşitliliği, veri analistlerinin bilmesi gereken temel bir konudur. Bu, kişilerin verileri etkili bir şekilde analiz etmelerine yardımcı olabilir [19].

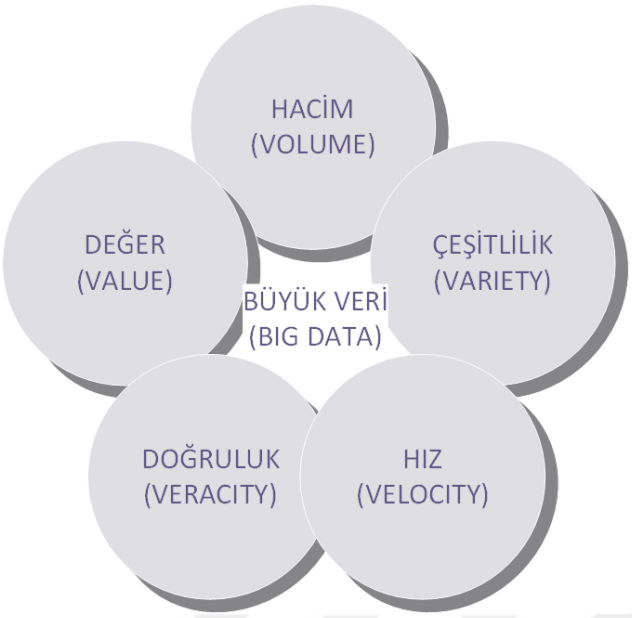
1.2.2.3. Hız

Hız, veri akışlarının sisteme girip çıktığı farklı oranlarla ilgilidir ve büyük veri teknolojileri, sistemlerine gelen veya giden orandan bağımsız olarak verilerin hızlı bir şekilde depolanabilmesi için bir soyutlama katmanı sağlamaktadır [16].

Verilerin hızı duran veriler, kullanılmış veriler ve hareket halindeki veriler olarak ayrı ayrı tanımlanabilir. Duran veriler tipik olarak ana verilerdir; arşivlenmiş veriler ve statik olan diğer veri kaynakları ile ilişkilendirilmekte ve değişmeyen verilerdir. Kullanılmış veriler genellikle işlem verileriyle ilişkilendirilmektedir. Hareket halindeki veriler ise bir uygulamadan başka bir uygulamaya gönderilen verilerdir [8][15][20].

Büyük veri çağında, veri çeşitliliğindeki ve sensör ağ dağıtımındaki artış, geleneksel sistemlerin işleyişini zorlaştıracak hatta imkansız kılacak şekilde sürekli olarak veri akışına yol açmıştır [19].

Çeşitlilik, hacim ve hız kombine edildiğinde bu üç özellik 3V olarak etiketlenmiştir. Ancak bu üç özellik, büyük verileri doğru bir şekilde etiketlemek için yeterli olmamıştır. Büyük veriler, aynı zamanda, çeşitli teknolojik yollarla çıkarılan bazı değerlere sahiptir. Büyük veride en önemli husus değerdir; verinin alacağı değer de ancak verinin doğruluğuna bağlıdır [21][22]. Kuruluşlar ve büyük veri uzmanları bu 3V modelini değer ve doğruluk kavramlarını ekleyerek 5V modele (Şekil 1.3) genişletmişlerdir [15].



Şekil 1.3. Büyük verinin 5V'si

1.2.2.4. Doğruluk

Verinin doğruluğu; verinin gerçekliğini, bütünlüğünü ve kullanılabilirliğini ele almakta, verinin güvenilirliğini ifade etmektedir. Organizasyonlar, veri üzerinde yapılan analizlerin yanı sıra verilerin doğruluğunu sağlaması gerekmektedir [16].

Büyük verilerin analizinin güvenilir olması ile işletmeler daha doğru kararlar alacak ve daha kaliteli sonuçlar verecektir [9][21][22].

1.2.2.5. Değer

Değer, işletmelerde taktik ve stratejik yönetimin büyük verilere uygulanmasıyla elde edilen sonuçtur [8]. Verilerin gerçek değerini (yani, içerdikleri bilgilerle ilgili verilerin potansiyel değerini) ilgilendirmektedir. Değer vermedikçe veri yetersizdir [16].

İşletmeler, verileri bir maliyet olarak kabul etmekle birlikte, işletmenin en değerli varlıklarından biri olarak veriler ele alınmaktadır. Doğru, eksiksiz ve güvenilir veriler insanları ve teknolojiyi geliştirmektedir [8].

Veriden değer elde edilmesi, yani veriden değerli bilgilerin çıkarılması, genellikle büyük veri analizi olarak adlandırılmaktadır. Değer, büyük veri için en önemli özelliktir, çünkü yararlı iş bilgilerinin üretilmesi sağlanmaktadır [23].

Büyük veri, değere dönüştürülene kadar kullanılmaz. Verinin değere dönüştürülmeden kullanımı, eksik kalitede sonuç çıkışına ve hesaplama maliyetlerinde artışa sebep olmaktadır. Bu da, etkili ve kolay kazançların sağlanmasına engel olmaktadır [20].

1.2.3. Büyük Verinin Kullanım Alanları

Büyük veri, otomotiv endüstrisi, yüksek teknoloji, petrol ve gaz endüstrisi, telekomünikasyon sektörü, ilaç ve sağlık hizmetleri, medya ve şov işi, seyahat ve ulaşım sektörü, sosyal medya ve çevrimiçi hizmetler, bilgi ve iletişim sektörü gibi birçok faaliyet alanında verimli bir şekilde kullanılmaktadır [9][15].

Sağlık alanında, temel olarak teşhis ve tedavi için medikal verilerin doğruluğu hayati önem taşımaktadır [24]. Büyük veri, hastaların sağlık durumlarını izlemek için kullanılmakta ve gelen yapılandırılmamış yada yarı yapılandırılmış verileri, düzgün bir şekilde analiz edilebilmesi için hazırlayıp temizlemektedir [25].

Perakende sektöründe, tüketicinin perakende alışkanlıklarını anlamaya dayanmaktadır. En iyi perakendeciler, müşterilerin verilerini incelemektedir. Büyük veri teknolojilerini pazarlama kampanyaları, ticaret planlaması ve tedarik zinciri yönetimi hakkında kararlar almaya yardımcı olması için kullanmaktadırlar [14].

Eğitimi, büyük veri teknikleri daha iyi hale getirme potansiyeline sahiptir ve benzeri görülmemiş eğitim fırsatları yaratmaktadır - örneğin, bir öğrencinin öğrenme stiline

dersleri uyarlayarak, çevrimiçi platformlar aracılığıyla kurslar açarak ve ebeveynler, öğretmenler ve öğrenciler için daha kolay hale getirmektedir-. Bu yeni teknolojiler, öğrenci performansını büyük ölçüde artırarak ve araştırmacılara, öğrenim ile ilgili değerli bilgiler sunarak, düşük teknolojili eğitim için gerekli müdahaleler geliştirilmesini sağlayabilir [26].

Ulaşımda, büyük veri analitiği, toplu taşıma için daha etkili hizmet sağlarken yolcu talebi hakkında kritik bilgiler edinme fırsatı sunmaktadır [14].

Finansta büyük veri, özellikle Kompleks Olay İşleme (Complex Event Processing, CEP) uygulamasıyla dolandırıcılık tespitinde önemli rol oynamaktadır [14].

Büyük verileri edinen kuruluşların, verilerin toplanması, depolanması ve analiz edilmesi için yeni bir mimariye gereksinimi vardır [27][28]. Bazı veri merkezleri, organizasyonel verilerde yıllık % 100'lük bir büyümeye sahiptir. Bir kuruluş, yüksek değerli verileri düşük veri depolama alanına depoladığında depo alanı bozuksa ya da bozulursa teknik risk oluşabilir [28]. Büyük verilerin analizi, kurumların daha iyi karar vermesi için bilgi sunma süreci anlamına gelmektedir [29]. Büyük verileri analiz eden kurumlarların, büyük verileri analiz etmeyen kurumlardan kar performansı daha yüksektir [28].

Büyük verilerin bir diğer önemli özelliği de politikaya etkisidir. Örneğin, 2012 yılındaki Başkan Barack Obama'nın seçilmesinde kısmen sosyal medyanın hedefli kullanılması "*Oylama ile kazanılan bir koalisyon için büyük veri analizi ve deneysel yöntemler kullanılmıştır"* [30] şeklinde ifade edilmiştir.

Benzer şekilde, arap baharı döneminde Twitter'ın rejim değişikliğinde büyük verinin bir etken olduğu düşünülmektedir. Günümüzde sosyal medya terör için bir araç olarak yaygın bir şekilde kullanılmaktadır. Aynı zamanda, terör ve suç ile mücadele edenler yasal olarak internet ve telefon verilerine erişerek daha fazla gözetim yetkisi istemekte, böylece tüm vatandaşın şahsi verileri açığa çıkmaktadır. Büyük veriler, suç ve teröre hizmet etmesi yönü ile demokrasi ve özgürlüğe hizmet etmesi yönü bakımından iki taraflı bir silahtır. Ayrıca, Rus korsanlarının 2016 ABD başkanlık

seçimlerine internet ve kitle iletişim araçları yardımıyla müdahale ettiği de iddia edilmektedir [31].

1.2.4. Verinin Güvenliği

Büyük veri çağında, daha önce görülmemiş miktarda veri üretilmektedir. Kurum ve kişiler için veri gizliliği ve güvenliği önemli bir mevzudur. Bu, özellikle de gizli bilgileri internet üzerinden gönderirken, verinin ne kadar güvende olduğu konusunda büyük bir endişe oluşturmaktadır.

Şirket ve kuruluşlar, çalışanlarının günlük iş prosedürleri için harici cihazları, medya ve web uygulamaları kullanması nedeniyle verileri koruma ve veri kaybını önleme zorluğuyla karşı karşıyadır. Çalışanların verileri çıkarılabilir cihazlara kopyalaması veya bulutlara yüklemesi nedeniyle hassas veriler artık şirketin kontrolü ve koruması altında olmayabilir [32]. Veri dağıtımları davetsiz misafirlere yönelik kolay hedeflerdir. Yetkisiz bir kullanıcı, verileri kullanmak için verilere erişebilmektedir. Bu erişimler de çeşitli sıkıntılar oluşturabilir, şirket ve kuruluşlar veri güvenliği tehdidine maruz kalabilirler [33].

Büyük veri güvenliğinin görevi; güvenlik duvarları, güçlü kullanıcı kimlik doğrulaması, son kullanıcı eğitimi, izinsiz giriş koruma sistemleri (Intrusion Prevention Systems, IPS) ve izinsiz giriş algılama sistemleri (Intrusion Detection Systems, IDS) ile yetkisiz kullanıcılar ile ilgilenmektir [33].

1.2.4.1. Kriptoloji

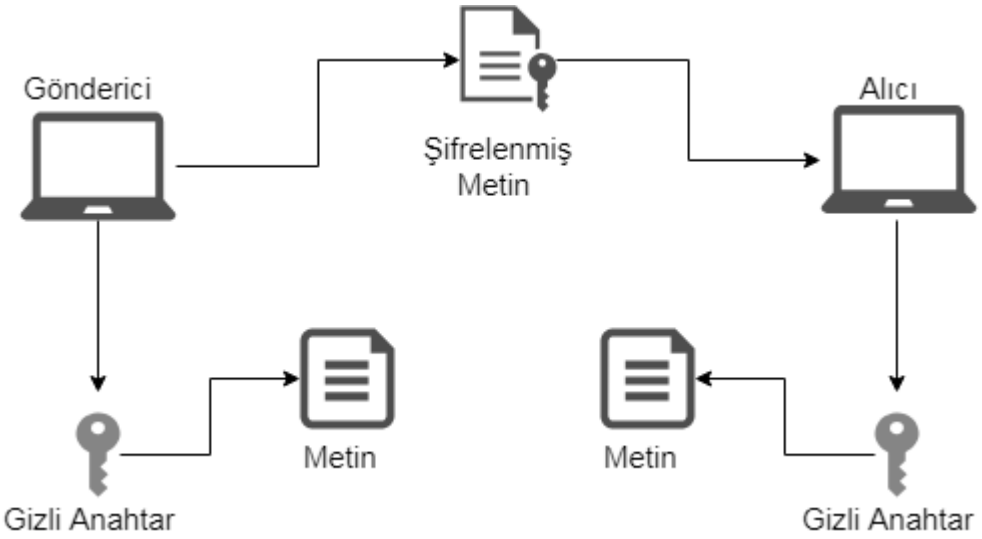
Şu anda, kuruluşlar tarafından kullanılan en popüler ve en etkili veri güvenliği yöntemlerinden biri kriptolojidir [32]. Kriptoloji, önemli bir bilgiyi sahibi olmayan yada yetkili olmayan kimsenin okuyamayacağı bir forma getirmekte ve çeşitli teknikler kullanarak orijinal haline geri döndürmektedir [34]. Kriptoloji, özellikle de

kontrol altında olmayan sistem veya ağ aracılığıyla bilgilerin depolandığı veya iletildiği durumlarda bilgileri korumak için etkili bir önlemdir.

İki temel veri şifreleme türü vardır: genel anahtar şifreleme olarak da bilinen asimetrik şifreleme ve simetrik şifreleme [32].

I. Simetrik anahtar şifreleme, Şekil 1.4'te görüldüğü gibi bir mesajı veya dosyayı şifrelemek ve şifresini çözmek için aynı gizli anahtarı kullanmaktadır. Simetrik anahtar şifrelemenin en büyük dezavantajı, bilginin birden çok taraf arasında güvenli bir şekilde paylaşılması için yeterince uygun olmamasıdır. Yani, aynı anahtarı hem bilgiyi şifrelemek hem de şifre çözmek için kullanması, anahtarları güvenli bir şekilde dağıtmayı gerektirir. Anahtar elektronik olarak gönderildiğinde bir güvenlik açığı ortaya çıkmaktadır. Anahtarların güvenli bir şekilde paylaşılması kolay olsaydı, iletişim taraflarının şifrelemeye ihtiyacı olmazdı, doğrudan bilgi alışverişi yapılabilirdi [35].

Yaygın olarak kullanılan simetrik anahtar şifreleme algoritmaları; Blowfish, Gelişmiş Şifreleme Standartı (Advenced Encryption Standard, 'Rijndael' AES), Veri Şifreleme Standartı (Data Encryption Standard, DES), Uluslararası Veri Şifreleme Algoritması (International Data Encryption Algorithm, IDEA), Üçlü DES (Triple DES, 3DES), RC4 (Rivest Cipher 4 ya da Ron’s Code 4), Twofish, Güvenli Özetleme Algoritması (Secure Hash Algorithm, SHA) [36].

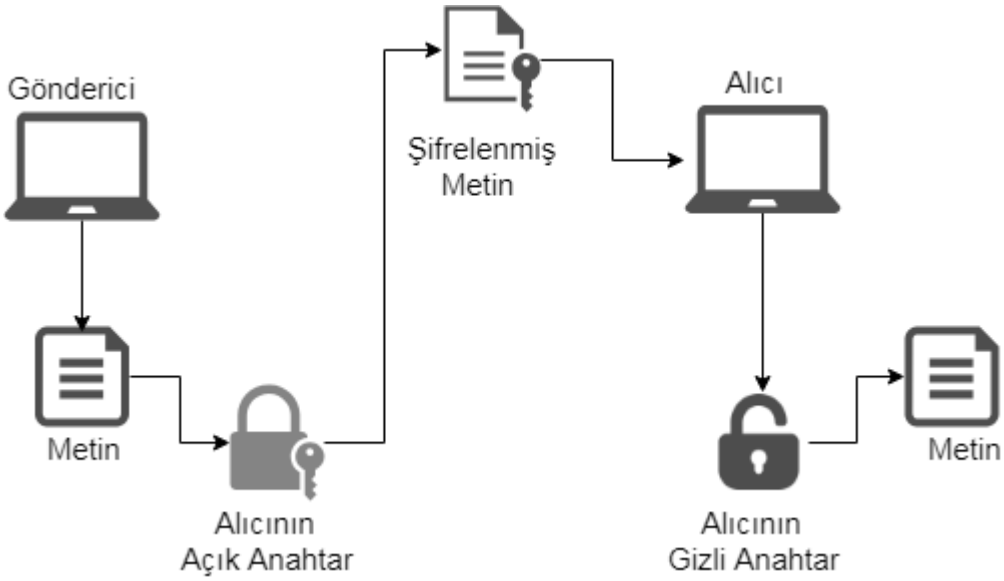


Şekil 1.4. Simetrik Anahtar Şifreleme

II. Genel anahtar şifreleme olarak da bilinen asimetrik anahtar şifreleme, Şekil1.5'te görüldüğü gibi aynı anda özel anahtar ve genel anahtar olmak üzere iki farklı anahtar kullanmaktadır. Özel anahtar yalnızca bilgisayarınız tarafından bilinmekte; genel anahtar ise bilgisayarınız tarafından güvenli bir şekilde iletişim kurmak isteyen herhangi bir bilgisayara verilmektedir. Gönderici taraf, iletiyi alıcının özel anahtarını kullanarak şifreler ve iletir. Şifrelenmiş bir iletinin kodunu çözmek için, kaynak bilgisayar (alıcı) tarafından sağlanan kendi genel anahtarını kullanır. Bir bilgisayardan başka bir bilgisayara asimetrik şifreleme ile şifrelenmiş mesaj gönderildiğinde, alıcı bilgisayara ait gizli anahtar olmadan mesajı alan kimse bu mesajı okuyamaz [37].

Anahtar çifti, büyük uzunluktaki asal sayılara (2, 3, 5, 7, 11 vb.) bağlıdır ve bu, sistemi son derece güvenli kılmaktadır; çünkü sonsuz sayıda asal sayı mevcuttur, yani anahtarlar için neredeyse sonsuz olasılıklar vardır [37].

Hem genel hem de özel anahtarları kullanıp bütünlüğünü, doğruluğunu, gizliliğini ve güvenilirliğini sağlayarak bir mesajı şifrelemesinden dolayı asimetrik şifreleme daha popülerdir [32]. Özellikle internet gibi güvensiz bir yolla gönderildiğinde, hassas verileri korumak için daha yaygın olarak kullanılmaktadır.



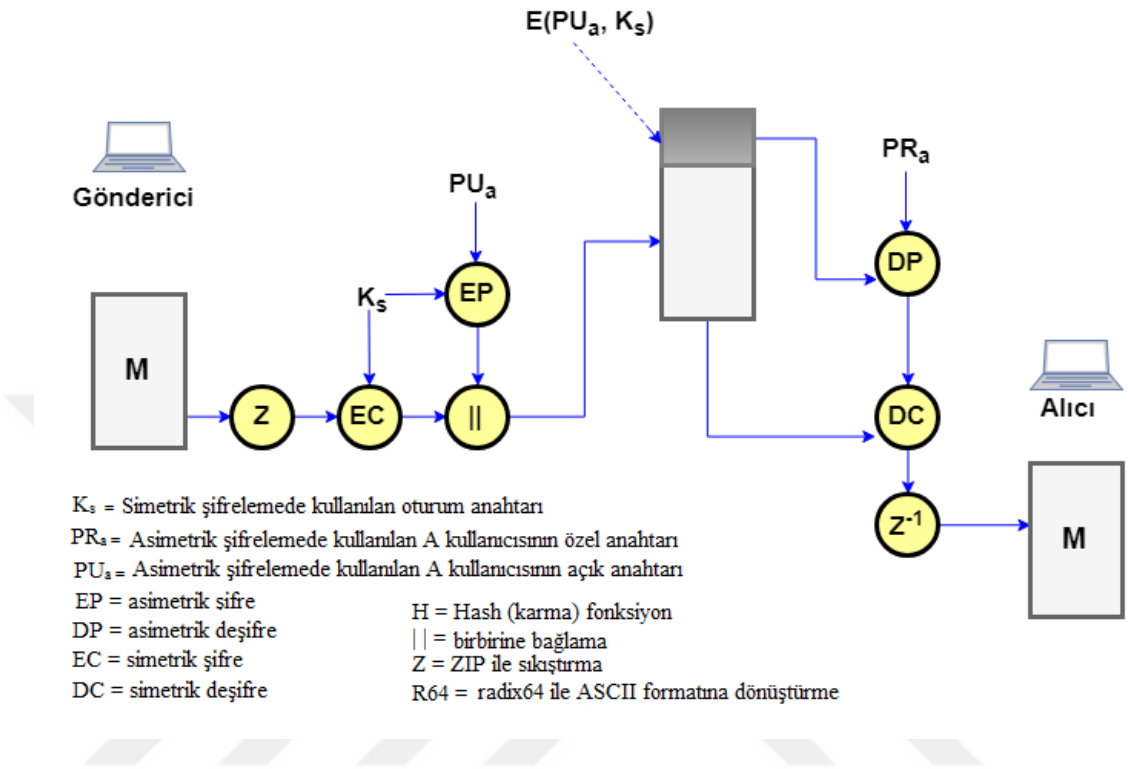
Şekil 1.5. Asimetrik Anahtar Şifreleme

Yaygın olarak kullanılan asimetrik anahtar şifreleme algoritmaları şunlardır: RSA (R. L. Rivest, A. Shamir ve L. Adleman), Dijital İmza Algoritması (Digital Signature Algorithm, DSA), Diffie Helman [36].

PGP (Pretty Good Privacy, Oldukça İyi Gizlilik), 1991'de güvenli bir şekilde iletişimi sağlamak için Philip Zimmermann tarafından yazılmıştır. PGP, asimetrik anahtar şifreleme sistemini ve simetrik anahtar şifreleme sistemini birlikte kullandığı için hibrit kripto-sistem olarak tanımlanmaktadır [38].

Düz metnin PGP ile şifrelenmesi durumunda (Şekil 1.6), PGP önce düz metni sıkıştırır. Metni sıkıştırma, iletim zamanında ve disk alanında fayda sağlamaktadır, daha da önemlisi kriptografik güvenliği güçlendirmektedir. Metni sıkıştırmasından sonra, tek kullanımlık bir oturum anahtarı oluşturulur. Simetrik şifreleme yöntemlerinden biri (örn, AES, 3DES) kullanılarak oluşturulan bu oturum anahtarı ile metin şifrelenir. Metin şifrelendikten sonra, oturum anahtarı da asimetrik şifreleme tekniği (örn, RSA) kullanılarak alıcının genel anahtarı ile şifrelenir. Şifreli oturum anahtarı ile şifreli metin birlikte alıcıya iletilir. Alıcı tarafında ise, işlem tersine çalışmaktadır; Asimetrik şifreleme tekniği kullanılarak kendi özel anahtarı ile

oturum anahtarının şifresi çözülür, simetrik şifreleme kullanılarak da oturum anahtarı ile şifreli mesaj çözülmektedir [38].



Şekil 1.6. PGP Şifreleme

PGP, internetteki kişisel e-postalar için yaygın olarak kullanılmaktadır. PGP'nin ücretsiz oluşu ve çalışmak için herhangi bir ek altyapı gerektirmemesi popülaritesini artırmaktadır [38].

1.2.5. Büyük Veri Uygulama Biçimleri

Devlet, sağlık, bilim, mühendislik, yapay zeka, genomik, trafik, sosyal medya, psikoloji gibi farklı alanlarda büyük verilerin genel konusuyla ilgili çok sayıda akademik ve profesyonel makale vardır. Araştırmacıların genel amacı, Amazon Web Hizmetleri'nin (AWS) Bulut ve Elastik MapReduce (EMR), Hadoop ve çeşitli

Apache gibi kaynak ürünlerinin veri depolamak ve işlemek için nasıl kullanıldığını gözlemlemektir [39].

BT yöneticileri, büyük veriyi, düşük maliyetli ve etkili bir şekilde analiz etmek ve yönetmek için en uygun teknik ve en iyi uygulamaları kullanmaktadırlar [39].

Dünyanın dört bir yanında şirketler, büyük verinin işlerine kattığı verimliliği fark etmeye başlamıştır. Her gün daha fazla şirket büyük veriyi uygulamaktadır. Büyük veriyi kullanmanın iki yolu vardır; şirketin tesislerinde kurularak kullanmak veya bulutta büyük veri platformu sunan bir sağlayıcı ile kullanmaktır. Geçmişte, şirketlerin sadece onu yerel olarak tesislerde kullanma seçeneği vardı, ama artık bu geçerli değildir. Her işletme farklıdır, yani bir şirket büyük veriyi yerel sunucularda kullanmayı tercih edebilirken, diğeri bulutta büyük veri kullanmayı tercih edebilir. Tercih etmeyi etkileyen dört faktör vardır [40].

I. Maliyet

Maliyet önemli ve çoğu zaman belirleyici bir faktördür. Yerel diskte büyük veri ile buluttaki büyük veri arasındaki maliyet farklılıkları nelerdir?

Yerel diskteki büyük veriler, verilerin toplanması, saklanması ve analiz edilmesi için şirketlerin yüksek maliyetli altyapı kurmasını gerektirmektedir. Genelde önden ödeme yapılan milyonlarca dolarlık bir süreçtir. Geçmişte birçok küçük işletme, büyük başlangıç maliyetleri nedeniyle büyük veriyi uygulayamamıştır. Şimdi, bulut bilişim ile, bu başlangıç maliyetleri çoğunlukla ortadan kaldırılmaktadır. Bulutta büyük veri çok daha ucuzdur, şirketlerin kullanımı için aylık ücretler vardır.

Ayrıca, yerel diskte bulunan büyük veri, genellikle ekipmanı izlemek ve veri toplama, depolama ve analiz işlemlerini yürütmek için bir uzman ekibi gerektirmektedir. Yine, bu pek çok şirketin sahip olmadığı ve işe yeni personel alma göze alamayacağı bir şeydir. Buluttaki büyük veriler, şirketler için bunu halletmektedir. Bulutta büyük veri ile hiçbir bakım ücreti yoktur.

II. Güvenlik

Yerel diskte bulunan büyük verinin şirketlere sağladığı en büyük avantajlardan biri veri güvenliğidir. Tüm veriler yerel sunucularda saklanabilir ve izlemek de kolaydır. Bulutta ise her zaman bir risk vardır. Bazı büyük bulut depolama şirketleri, verilerin güvenli olmasını sağlamak için gerekli adımları atmaktadır; Endüstri standardı şifreleme yöntemleri, diğer güvenlik önlemleri ile birlikte verilerin kaybolma riskini ortadan kaldırmaktadır. Bulutta veriler yerel diskte olduğu kadar güvenli değil, ancak buna yakındır.

III. Mevcut Yetenekleri

Bu tercihte göz önünde bulundurulması gereken önemli bir nokta, büyük veri kullanımı için gerekli ihtiyaçlardır. Uygulamalarda destek olacak personel var mı? Büyük veri'nin tüm yönlerini denetleyebilecek bir ekip var mı? Uygun bakım ve iş akışını sağlayabilecek bir ekip var mı? Eğer bunlar yoksa, onları işe alma sağlanabilir mi? Büyük veride önemli oranda personel ihtiyaçları vardır. Bulutta, büyük veriler için personel sağlanmaktadır.

IV. Ölçeklenebilirlik

Ölçeklenebilirlik, bir şirketin veri toplama yeteneklerini arttırmak veya azaltmak için sahip olduğu esnekliktir. Yerel diskte büyük veriler için ölçeklenebilirlik çok daha zordur. Normalden daha fazla veriye sahipse, aşırı maliyetli olabilecek daha fazla altyapı yüklenmesi gerekmektedir. Daha az veriye sahipse, maliyetli ve kullanılmayan altyapıya takılmış olur. Buluttaki büyük veriler için ölçeklendirme inanılmaz derecede kolaydır ve olumsuz finansal çıkarımlar yapmadan yukarı veya aşağı ölçeklendirmeye izin vermektedir.

Her işletme farklıdır. Bazıları, yerel diskte büyük veri ile birlikte gelen güvenlik ve kontrol kolaylığını tercih edebilir ve bunları karşılayacak kaynakları vardır. Diğerleri ise, bulutta büyük veri ile sağlanan esnekliği ve rahatlığı tercih edebilir. Her iki durumda da büyük veriyi uygulamak gerekebilir [40].

1.2.6. Büyük Verinin Yararları

Büyük veri analizi her türlü kuruluş için avantaj sağlayabilir. Sadece dijital olarak faaliyet gösteren şirketler değil (örneğin, internetteki şirketler), aynı zamanda geleneksel firmalar da avantajlarından yararlanabilir ve büyük veri sağlayabilir. Açıkçası bu, büyük verileri kullanarak, işletmeler için daha fazla ölçüm yapılması anlamına gelmekte ve bu şekilde kuruluşlar, işletmeleri hakkında daha fazla bilgi sahibi olmaktadırlar. Bu faydalar / avantajlar üç kategoriye ayrılabilir: 1) Maliyet azaltma ve daha iyi marjlar, 2) Daha hızlı ve daha iyi karar verme ve 3) Süreçlerin ve ürünlerin optimizasyonu [41].

1.2.6.1. Maliyet Azaltma ve Daha İyi Marjlar

Büyük verilerin en çok bahsedilen faydalarından biri maliyetleri azaltmasıdır [42]. Bu maliyet avantajı, operasyonel büyüklüğün artması nedeniyle ortaya çıkan ölçek ekonomileri (üretimin artması maliyetin düşmesi) ve kapsam ekonomileri (kurumun bir alandaki gücünü farklı sektörlerde kullanması maliyetin düşmesi) ile elde edilmektedir. Kurumlar verilerini analitik uygulamalar için özel olarak donatılmış kurumsal depolara taşımasıyla maliyetin azalması mümkündür [43]. Arz / talep zincirindeki ürün ve hizmetlerin hızının artması beklenmekte ve ölçek ekonomileri ortaya çıkmaktadır. Maliyet azalması, çalışanların büyük veri kullanma konusunda eğitilmesi ile de kolaylaştırılmaktadır. Bu durum söz konusu olduğunda ve kurum içinde bir öğrenme ve eğitim ortamı oluşturulduğunda, görevlerin daha etkin bir şekilde yapılması ve maliyet düşmesi mümkündür.

Maliyet azalmasının yanı sıra, kar marjları ve karları artırmaya yönelik diğer faydalar da büyük verilerle ortaya çıkmaktadır. İyi bilinen bir örnek dinamik fiyatlandırmadır [44]. Bununla birlikte, arz ve talep daha iyi koordine edilip, bu da ürün ve hizmetlerin fiyatına yansımaktadır. Ziyaret sayıları veya alımların sayısına bağlı olarak, şirketler belirli ürünlere yönelik büyük taleplerde, ürünleri rakiplerinden daha ucuza satabilir veya daha yüksek kar marjları elde etmek için fiyatları artırabilirler.

1.2.6.2. Daha Hızlı ve Daha İyi Karar Verme

Büyük veri tekniklerini kullanarak, verilerin toplanması, depolanması ve analiz edilmesi daha kolay ve daha ucuz hale gelmektedir. Ayrıca verinin kullanılabilirliği ve bilginin şeffaflığı artmaktadır. Bu büyük veri teknikleri, veride yeni model ve bağlantılar bulunmasını mümkün kılarak karar vermede çeşitli avantajlar sağlamaktadır [45][46]. Bu tekniklerle müşterilerin davranışları hakkında daha fazla bilgi edinilebilir ve daha önce sezgiye dayalı olarak tahmin edilen müşteri talepleri bu bilgilerle daha doğru tahmin edilebilir [46]. Kuruluşlar satış etkinliklerini geliştirmek için büyük veri kullanmaktadır. Daha hızlı ve daha iyi karar verme, organizasyonda ve örgütsel stratejide verimliliğin ve etkinliğin artmasını sağlayarak daha iyi bir firma performansına yol açabilir [47].

Büyük verileri kullanarak, hem firma performansı artırılabilir, hem de sahip olduğu performans bilgileri daha iyi ölçülebilir. Bilgilerin daha iyi kullanılması için geliştirilen teknikler ile daha iyi yönetim kararları vermek mümkün bir hale gelebilir.

1.2.6.3. Süreç ve Ürünlerin Optimizasyonu

Büyük veri, kurumlarda belirli bir hedef grubun belirli bir zamanda hangi özel ürünlere ihtiyaç duyduğunu keşfetmesini sağlamaktadır. Böylece kuruluşlar müşterilerin heterojenliğini veya kişisel tercihlerini daha iyi anlayabilmektedir. Bu sayede müşterilerin isteklerini ve ihtiyaçlarını organizasyonun süreçleriyle uyumlu hale getirmek mümkün olmaktadır. Bunun yanı sıra iş süreçlerini optimize etmek de mümkündür, çünkü iş süreçleri ve tedarik zinciri hakkında daha fazla bilgi ortaya çıkmaktadır; hangi noktada gereksiz maliyetlerin ortaya çıktığı konusunda daha fazla bilgi sağlayabilir. Bu karmaşık teşhisler sayesinde, belirli problemler için özel çözümler bulunmaktadır. Bu da, problem çözme ve süreçlerin optimizasyonu için potansiyeli artırmaktadır [45].

1.2.7. Büyük Verinin Zorlukları

Büyük veri analizi şu anda popüler olan bir konudur. Mevcut literatürde açıklanan faydalar ve avantajların yanı sıra, büyük verileri uygulamaya yönelik zorluklar da bulunmaktadır [42]. Büyük verilerin zorlukları, teknolojik zorluklara ve yönetimsel zorluklara bölünebilir [46].

1.2.7.1. Teknolojik Zorluklar

Teknolojik zorluklar BT (Bilgi Teknolojisi) altyapısına, güvenlik, gizlilik ve diğer teknolojik zorluklara dayanmaktadır. Büyük veriler veri depolama, veri işleme ve veri alışverişi için mevcut veritabanı sistemlerini veya BT altyapısı sınırlarını aşan kapasite sorunlarına yol açmaktadır [42][48]. Bu nedenle, büyük veri kullanımı için fırsatları ve gereksinimleri tam olarak sağlayabilecek bir BT altyapısına sahip olmak önemlidir.

BT altyapısının yanı sıra, verilerin güvenliği ve gizliliği de teknolojik bir zorluktur [49]. Büyük veri depoları birçok değerli (gizli) bilgi içermekte ve bu bilgilerin üçüncü taraflarca yetkisiz kullanıma karşı korunması önemli bir husustur. Korumayı sağlamak için, verilere erişim sadece gerekli olması durumunda bu konuda yetkilendirilmiş çalışanlar tarafından gerçekleştirilmelidir.

1.2.7.2. Yönetim Zorlukları

Teknolojik zorlukların yanı sıra yönetimsel zorluklar da ortaya çıkmaktadır. Büyük verilerden önce birçok karar sezgi ve deneyime dayanmaktaydı. Büyük veri ile birlikte, karar verme bilgilere dayanarak gerçekleşmektedir. Bu nedenle yönetim süreçleri daha önemli hale gelmekte ve bu da yönetimsel zorluklarla sonuçlanmaktadır [46]. Bu yönetim zorlukları yönetici ve çalışan becerilerine, ekip sıkıntılarına ve karar verme problemlerine bölünebilir. Bu sıkıntılar, yönetim ile ilgili

zorluklarla baş edebilecek veri bilimcileri ve programcılarına olan ihtiyacı ortaya çıkarmıştır.

Son yıllarda büyük veri uygulamalarının geliştirilmesi giderek önem kazanmaktadır. Farklı sektörlerden organizasyonlar bugün, büyük miktarda veriden elde edilen bilgiye bağımlıdır. Bununla birlikte büyük veri için, geleneksel veri teknikleri ve platformları yavaş tepki vermekte ve ölçeklenme, performans ve doğruluk bakımından yetersiz kalmaktadır. Karmaşık büyük veri zorluklarıyla yüzleşmek için çok fazla çalışma yapılmakta ve çeşitli teknolojiler geliştirilmektedir.

1.3. Bulut Bilişim

Bulut bilişim, büyük veri sorununa ölçeklenebilir ve uygun maliyetli bir çözüm sunmaktadır. Çevrimiçi olan herhangi bir şeyi temsil etmek için kullanılmış olup, her ne kadar büyük ölçüde yanlış tanımlanmış olsa da bulut bilişim, Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından, "Ç*ok az yönetim çabasıyla ya da servis sağlayıcı etkileşimiyle hızlı bir şekilde hazırlanabilen ve piyasaya sürülebilen, yapılandırılabilir hesaplama kaynaklarına (Örneğin ağlar, sunucular, depolama ortamları, uygulamalar ve hizmetler) her yerde, uygun bir şekilde, isteğe bağlı olarak erişimi sağlayan bir model"* olarak tanımlanmaktadır [50].

Bulut bilişim, internet ve uzak veri merkezleri üzerinden sağlanan güvenilir yazılım, donanım vaat etmektedir [51]. Bulut hizmetleri, büyük ölçekli bilgi işlem görevlerini yerine getiren; depolama ve hesaplamadan, veritabanı ve uygulama hizmetlerine kadar birçok BT işlevini kapsayan güçlü bir mimari haline gelmiştir. Çok sayıda veri kümesini saklama, işleme ve analiz etme ihtiyacı, birçok kurum ve kişiyi bulut bilişimi benimsemeye itmiştir [52].

Bulut hizmeti sağlayıcıları, kullanıcıların bulut kaynaklarına erişmelerini ve programlarını dağıtmalarını sağlayarak paralel veri işlemeyi kullanmıştır. Büyük çaplı deneyler için çok sayıda bilimsel uygulama günümüzde bulut ortamında sağlanmakta; ayrıca sermaye maliyetlerindeki azalma, üretilen ve tüketilen veri

miktarında artma ve özellikle de yerel sunuculardaki bilgi işlem olanaklarının eksikliği nedeniyle bulut bilişimin kullanımında artma devam edecektir [53][54].

Bulut bilişim, zengin bir dizi hesaplamayı, altyapıyı ve depolama hizmetlerini birleştirerek oldukça çekici bir ortam sunmaktadır [55]. Kablosuz ağların ve mobil cihazların artan popülaritesi, her bir cihazın sınırlı işlem ve depolama kapasitesi ve sınırlı pil ömrü nedeniyle bulut bilişimini yeni boyutlara taşımaktadır [56].

Bulut bilişim, küçük ve büyük ölçekli işletmelere büyük veri uygulama olanağı sunmaktadır. Veri kaynakları e-posta, mobil cihaz verileri ve sosyal medya verileri içerecek şekilde geleneksel veritabanının ötesine uzanmaktadır. Büyük veri büyük depolama alanı gerektirir. Depolama fiyatı düşmeye devam etmesine rağmen, büyük verilerden yararlanmak için ihtiyaç duyulan kaynak hala küçük ve orta ölçekli işletmeler için finansal zorluklar oluşturmaktadır. Bulut bilişim; veri depolama ve büyük veri analizlerinin kullanımı göz önüne alındığında, küçük ve orta ölçekli işletmeler için de uygun bir seçenektir [57].

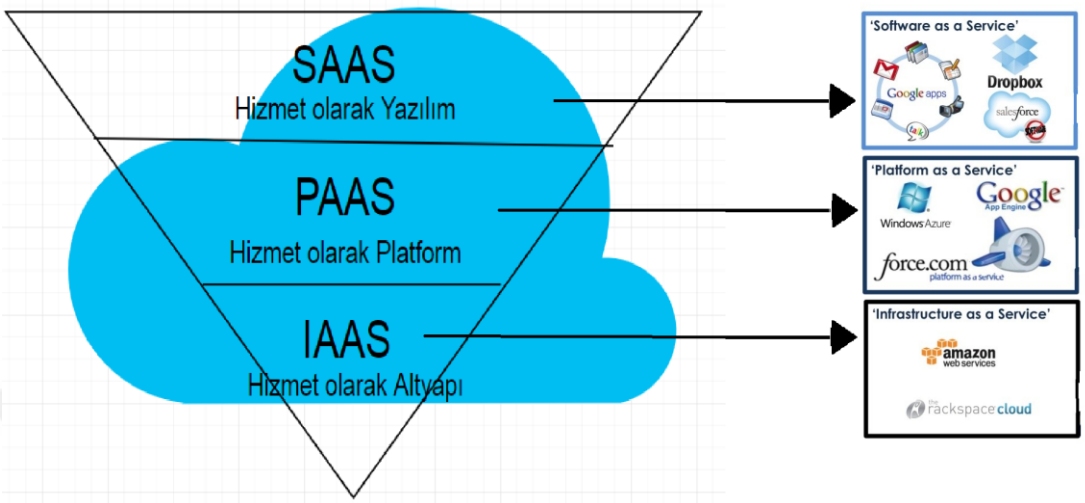
Bulut bilişim, bilgi işlem kaynaklarını bir yardımcı program olarak sağlama amacıyla sanallaştırma teknolojisinden yararlanmaktadır.

Sanallaştırma, fiziksel donanımın daha az maliyetle paylaştırılmasıdır ve kullanıcılara hizmet için soyut (sanal) kaynaklar sağlayan bir teknolojidir. Sanallaştırılmış bir sunucu sanal makine (Virtual Machine, VM) olarak adlandırılır. Sanallaştırma, bulut bilişimin temelini oluşturmaktadır. Sanal kaynakların isteğe bağlı, dinamik olarak atanması hızlı ve esnek bir şekilde yönetimini sağlamaktadır [58].

Bulut bilişim; kullanıcılara terabyte'lık depolama alanına, yüksek işlem gücüne ve rahat kullanıma sahip bir modele erişim olanağı tanımaktadır [16].

Bulut sağlayıcılar genellikle üç farklı temel hizmet sunmaktadır: Hizmet Olarak Altyapı (IAAS), Hizmet Olarak Platform (PAAS) ve Hizmet Olarak Yazılım

(SAAS). Bu üç hizmet yakından ilişkilidir: Şekil 1.7'deki gibi, PAAS, IAAS'nin üstüne, SAAS da, PAAS'nin üstüne geliştirilip ve inşa edilmiştir. [16]



Şekil 1.7. Bulut Bilişim Platformları

1.3.1. IAAS

IAAS, bulut bilişim sağlayıcılarının yüksek performanslı bilgi işlem (High Performance Computing, HPC) için veya veri merkezi altyapısı kurmak ve sürdürmek için sermaye harcadıkları yer denebilir. Kullanıcılar, bu hizmet için geniş bant bağlantılarını kullanıp sunucu veya depolama altyapısına erişim sağladıkları sürece yani sistemi kullandıkları sürece ödeme yapmaktadırlar. IAAS genellikle yardımcı yazılım veya elastik -talep üzerine büyütme veya küçültme yeteneği olan- bilgisayar olarak adlandırılmaktadır. IAAS sağlayıcıları, kendi belirledikleri işletim sistemi ile sanal makineleri oluşturabileceği ve bazı durumlarda gerekli olan uygulamaların yapılabileceği sanallaştırma teknolojisini kullanmaktadır [16][50].

Flexiscale ve Amazon EC2 gibi IAAS, servis sağlayıcılar tarafından sağlanıp talep üzerine kullanılan ve bulut üzerinde çalışan donanım ekipmanı anlamına gelmektedir[7].

1.3.2. PAAS

PAAS, kullanıcıların bulut sağlayıcısı tarafından geliştirilen yazılım kütüphaneleri veya geliştirme platformları üzerine inşa ederek çeşitli uygulamalar oluşturmasına izin vermektedir [16] [50].

Google'ın Uygulama Motoru, Force platformu ve Microsoft Azure gibi PAAS, kullanıcılara platform sağlamak için bulutta çalışan farklı kaynaklar anlamına gelmektedir.

Microsoft Azure hizmet platfomu, bilgisayar bilimi alanında devrim niteliğinde olan ve büyük veri alanındaki en yeni teknolojilerden veri depolama ve analizini sağlayan Hadoop açık kaynak çerçevesini kullanmaktadır.

1.3.3. SAAS

SAAS, en bilinen bulut modellerinden biridir ve doğrudan bulut sağlayıcısında çalışan uygulamalardan oluşmaktadır [16].

SAAS, internet aracılığıyla erişilebilen bulut altyapısında çalıştırılan uygulamalardan yararlanmayı sağlayan hizmettir. Kullanıcıların kendi sistemlerine herhangi bir kurulum yapmadan uygulamalara erişmelerini sağlamaktadır; burada kullanıcıya gerekli olan sadece internet bağlantısı ve tarayıcıdır. Kullanıcı daha sonra tüm yazılımların kurulu olduğu bir VM aracılığıyla bir masaüstü ortamına bağlanmaktadır [50].

Google Dokümanlar, Gmail, Salesforce.com ve Dropbox gibi SAAS, bulut sağlayıcı tarafından internet üzerinden erişilebilen hizmetler olarak bulut altyapısında çalışan uygulamalar sağlamaktadır.

Bulut bilişim ortamında, Hadoop büyük veri analitiği için yaygın olarak kullanılmaktadır. Tüketicilerine pek çok avantaj sağlamakta; ancak, çözülmesi

gereken problemlerle de karşılaşılmaktadır. Hadoop'un günümüz bulut altyapısında karşılaştığı sorunlardan bazıları aşağıdaki gibidir:

- Bulut bilişim kümelerinde, bir arıza meydana gelebilir ve bu, sistemi orijinal hallerine geri döndürmek için çok maliyetli olabilir.
- Hadoop'taki ad düğümü, tek hata noktası (SPOF) problemidir. Bellek kaynağı tükenmesi, ad düğümü üzerinde büyük bir hataya neden olabilir.
- Bulut bilişim kümelerinin boyutu büyüdükçe, bu kümelerin sağlığını korumak gittikçe zorlaşır ve sistemlerin her an çökme olasılığı oluşur [59].

1.4. Hadoop

Hadoop, emtia donanımı üzerinde büyük miktarda verileri işleyen dağıtık uygulamalar yazmak ve çalıştırmak için kullanılan java tabanlı bir yazılım kütüphanesidir. Hadoop, 2005 yılında iki Yahoo çalışanı Doug Cutting ve Mike Cafarella tarafından, Nutch arama motoru projesinin dağıtımını desteklemek için geliştirmiştir. Cutting bu projeyi, oğlunun oyuncak filinin adı olan Hadoop olarak adlandırmıştır. Geliştirme sonrasında Hadoop, Apache Yazılım Kurumunun tescilli ticari markası olmuştur [60].

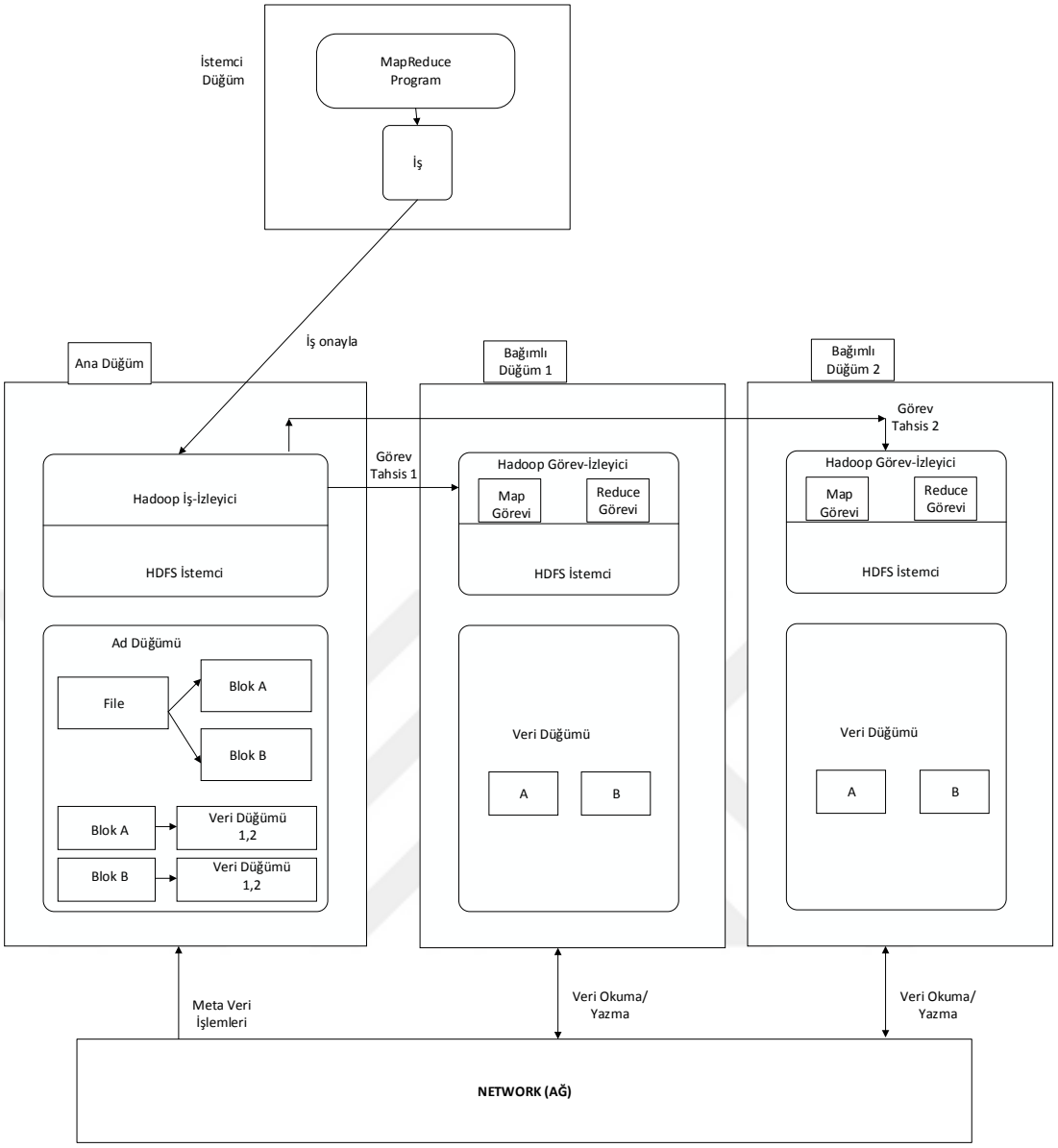
Apache Hadoop [61], Google'nin MapReduce ve Google Dosya Sistemi (Google File System, GFS)'den türetilmiş açık kaynaklı bir yazılım çerçevesidir. Büyük Veri analitik alanında son zamanlarda popüler olan dağıtılmış bir mimariye sahiptir. Çok sayıda emtia makinelerinde veya Amazon EC2 gibi bulut bilişim hizmetlerinde kullanılmaktadır [62] [63].

Geleneksel yönetim sistemleri (RDBMS), son zamanlarda üretilen bu büyük hacimdeki veriyi temel veri yapılarıyla idare edememektedir. Buna çözüm olarak Apache Hadoop, büyük veriyi işlemek için kullanılan en popüler teknolojidir.

Bir Hadoop kümesi veri kaybı ve veri aktarımında ek maliyete gerek kalmadan yeni sunucular veya kaynaklar ekleyerek genişletilebilmektedir. Hadoop, büyük hacimli verilerin maliyet-etkin depolanmasını ve işlenmesini sağlamaktadır. Hadoop'un bir

başka güçlü yönü de hataya dayanıklı olmasıdır. Bu güvenli ve kesintisiz veri işleme anlamına gelmektedir. Veriler tek bir düğümde depolanmaz. Kümede bir düğüm kaybolduğunda sistem, verileri başka bir düğüme yönlendirip işlemeye devam etmektedir. Öte yandan, yedekleme sistemleri de çalışmaktadır. Hadoop, büyük ölçekli veri setlerinde dağıtılmış hesaplama ile güvenilir, ölçeklenebilir ve paylaşımlı bir depolama amaçlamaktadır. Verileri işlemek için MapReduce programlama modeli ve depolama için Hadoop Dağıtılmış Dosya Sistemi (HDFS) kullanılmaktadır. Her iki mimari de yüksek kapasiteli veri setleri ile çalışmaya uygundur. Hadoop, katman yapısı olduğundan büyük veriler üzerinde toplu olarak paralel işlemeyi sağlamaktadır [60].

Hadoop kümesi, Şekil 1.8'de görüldüğü gibi birden fazla bağımlı düğüm ve bir ana düğümden oluşur. Ana düğüm, her katmanın ana bileşenlerini, yani HDFS depolama katmanı için ad düğümü (NameNode) ve MapReduce işleme katmanı için iş takipçisi (JobTracker), makinelerin geri kalanı ise bağımlı bileşenleri, yani HDFS katmanı için veri düğümü (DataNode) ve MapReduce katmanı için görev takipçisi (TaskTracker) çalıştırmaktadır. Ana düğüm ayrıca bağımlı düğüm rolü oynayabilir. Böylece, ana düğümün bileşenlerine ek olarak ana düğüm, bağımlı düğümün bileşenlerini de çalıştırabilmektedir. Ana düğümde çalışan bileşenler, veri depolama ve işleme için iş yürüten tüm düğümlerdeki bağımlı bileşenlerini koordine etme ve yönetme sorumluluğundadır [64].



Şekil 1.8. Hadoop Mimarisi

1.4.1. MapReduce

MapReduce, Google'ın büyük verileri işlemek için sunduğu bir çözümdür ve internet arama motoru sağlayıcıları MapReduce'u, milyarlarca web sayfasını hızlı ve anlamlı bir şekilde işlemesi için geliştirmişlerdir. MapReduce, Java tabanlı, bir dizi makineyi dağıtılmış bir şekilde çalıştırmak üzere tasarlanmış bir programlama modelidir [50].

MapReduce programlama modeli, verilerin kümeler boyunca küçük parçalar halinde dağıtılıp verileri paralel olarak işlemek için kullanılmaktadır [65]. MapReduce, derin veri analizi ve yüksek hızda paralel programlama sağlayan bir araç olarak tasarlanmıştır [60].

Kullanıcılar, harita ve indirgeme işlevi açısından hesaplamayı belirlemektedir. MapReduce çalışma sistemi de, büyük veri kümeleri arasındaki bu hesaplamayı otomatik olarak paralelleştirmekte ve makine arızalarını kontrol ederek ağ ve disklerin verimli kullanılması için küme içi iletişimi sağlamaktadır. Programcılar sistemi kullanmayı kolay bulmaktadır; Google'da son dört yıl içinde on binin üzerinde farklı harita analiz programı dahili olarak uygulanmıştır ve Google'nin kümelerinde her gün ortalama yüz bin harita görevi (toplamda günde 20 PB'den fazla veri işlenmiştir) yürütülmüştür [66].

1.4.1.1. MapReduce İşlerinin Akışı

MapReduce işlemi, veri kümesindeki n adet veri için çalıştırılmaktadır. Bu işlem herhangi boyuttaki girdiler için yürütülebilmektedir. Herhangi bir hacimde yapılandırılmamış verilerin başarılı bir şekilde yapılandırılmasını, verilerin hızlı ve verimli bir şekilde işlenmesini desteklemektedir [65].

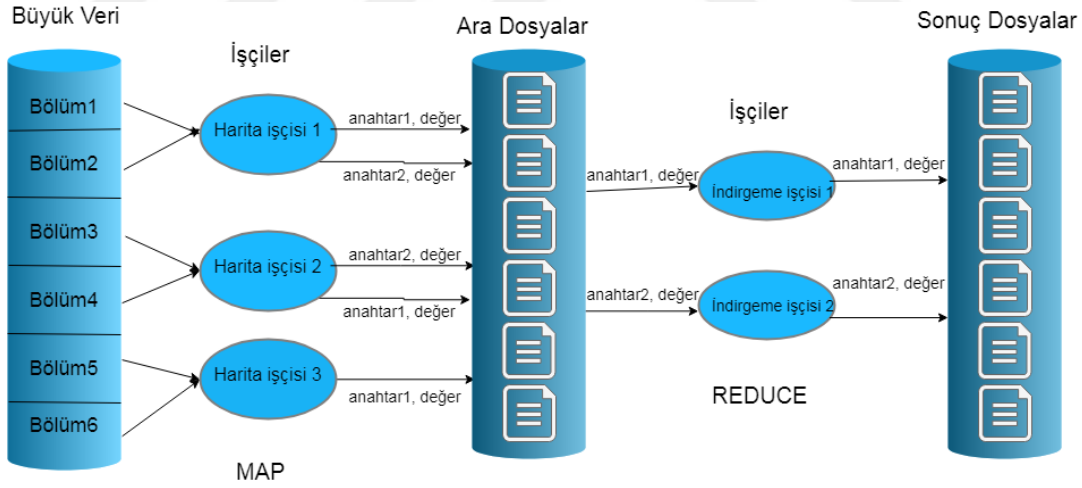
MapReduce, büyük veri uygulamaları için büyük fayda sağlamaktadır. Etkin ve düşük maliyetli mekanizmaları ile büyük verinin işlenmesini basitleştirmektedir. Paralel işleme destekleyen programlar yazılmasını sağlamaktadır.
MapReduce programlama modeli, veri hesaplamalarını işleyen iki fonksiyonu kullanır: harita (map) fonksiyonu ve indirgeme (reduce) fonksiyonu. Bir MapReduce programı aşağıdaki operasyonlara dayanmaktadır:

1. Öncelikle harita fonksiyonu, girdi verisini (örneğin, uzun metin dosyası) anahtar / değer çifti oluşturan bağımsız veri bölümlerine ayırmaktadır.
2. Ardından tüm anahtar / değer çifti, her birini ayrı ayrı işleyen harita görevlerine gönderilmektedir. Her veri bölümü benzersiz bir hesaplama düğümüne atanmaktadır. Harita görevi, bir veya daha fazla ara anahtar / değer

çifti çıkışı üretmektedir. Bu aşamada, tüm ara anahtar / değer çiftlerini toplayıp anahtarla sıralamak ve gruplandırmak için yükümlüdür.

(Harita görevini çalıştıran düğüm başarısız olursa, başka bir makinede aynı veriler üzerinde çalışan harita görevinin yeni bir örneği başlatılmaktadır.)

3. Bu ara anahtar / değer çiftleri giriş olarak indirgeme fazına taşınmaktadır. İndirgeme işlevi, tüm ara veriler aktarılıp sıralanıncaya kadar başlamamaktadır. İndirgeme işlevi bir anahtar için değerleri birleştirmektedir. Her bir benzersiz anahtar için indirgeme fonksiyonu, önceden tanımlanmış bir programa (yani, filtreleme, özetleme, sıralama, karma, ortalama alma veya maksimum bulma) göre anahtarla ilişkilendirilen değerleri toplamaktadır. Bundan sonra, bir veya daha fazla çıktı olarak anahtar / değer çifti üretilmektedir.
4. Harita ve indirgeme işlerinden sonra, sonuç Hadoop Dağıtılmış Dosya Sistemi'nde saklanmaktadır. Şekil 1.9'da MapReduce işleminin akışını göstermektedir [60] [65].



Şekil 1.9. MapReduce İş Akışı

MapReduce, iş-izleyici uygulamasının ana düğümde çalıştığı görev-izleyici uygulamasının da bağımlı düğümünde çalıştığı ana / bağımlı mimarisini kullanmaktadır [67].

1.4.1.2. İş-İzleyici

İş-izleyici hizmeti ana düğümde çalışmakta ve görev-izleyici tarafından bağımlı düğümlerinde yürütülen MapReduce görevlerini izlemektedir. Kullanıcı, ana düğümle etkileşim içinde işi iş-izleyiciye göndermektedir. İş-izleyici daha sonra ad düğümünden HDFS'deki gerçek veri konumunu istemektedir. İş-izleyici, görev-izleyicileri bağımlı düğümlerinde bulup işleri bu düğümlerdeki görev-izleyicilere göndermektedir. Bağımlı düğümdeki görev-izleyici, canlı / aktif olduğunu ve kendisine ayrılan işi yürüttüğünü göstermek için düzenli olarak iş-izleyiciye kalp atışı (heartbeat) mesajı göndermektedir. Belirtilen süre içinde kalp atışı mesajı alınmazsa, belirli bir bağımlı düğümdeki görev-izleyici işlevsiz olarak kabul edilmektedir ve atanan görev başka bir görev-izleyici üzerinde planlanmaktadır. İş-izleyici ve görev-izleyici, MapReduce motoru olarak bilinmektedir. İş-izleyici, MapReduce hizmeti için bir hata noktasıdır, iş-izleyici düşerse, tüm işler durdurulmaktadır [60] [67].

1.4.1.3. Görev-İzleyici

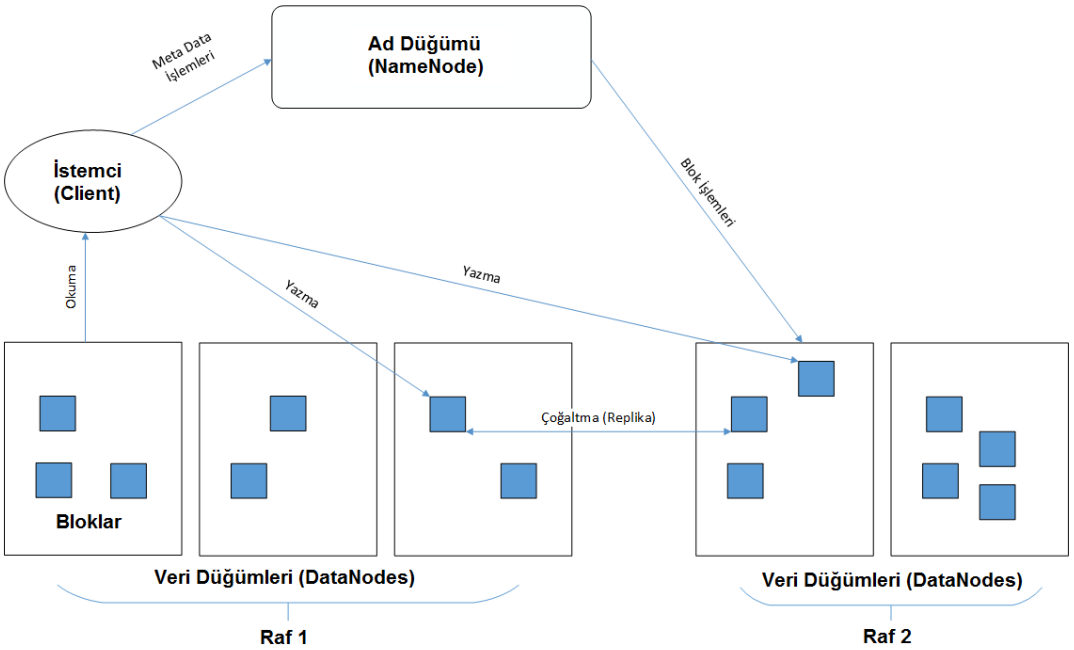
Görev-izleyici hizmeti bir kümenin bağımlı düğümlerinde çalışmaktadır. İş-izleyiciden işleri kabul etmekte ve MapReduce işlemlerini yürütmektedir. Ve ilerleme raporunu iş-izleyiciye göndermektedir. Her görev-izleyici, bir düğümün donanım şartlarına bağlı olarak sınırlı sayıda görev yuvası içermektedir. İş-izleyici, bir bağımlı düğümündeki görev-izleyicide kaç tane görev yuvasının bulunduğunu aldığı kalp atışı mesajı ile öğrenmektedir. İş-izleyici, gelen işin kaç tane boş görev yuvasına ihtiyaç olduğuna bağlı olarak belirli görev-izleyicisinde gereken yuva sayısınca tahsis edilmesini sağlamaktadır [60] [67].

1.4.2. HDFS

HDFS, emtia donanımı üzerinde güvenilir, ölçeklenebilir ve hataya dayanıklı veri depolama sağlayan, kendi kendini iyileştiren, dağıtılmış bir dosya sistemidir. HDFS,

ihtiyaçlara ve sorgulara bağlı olarak depo alanını ölçeklendirilebilmekte, MapReduce ile ortak çalışarak büyük kümeler arasında hesaplamayı dağıtık bir şekilde sağlamaktadır. HDFS, mimariden bağımsız olarak metin, resim, video vb. formatlarda veri kabul etmekte ve yüksek bant genişliğinde akış için optimize etmektedir. HDFS'nin en önemli avantajı hata toleransıdır. Düğümler arasında hızlı veri aktarımı sağlayarak ve Hadoop'un düğüm hataları durumunda bile hizmet vermeye devam etmesini sağlayarak felaket riskini azaltmaktadır [67].

Ana / bağımlı mimarisini kullanan HDFS depolama katmanı iki bileşenden oluşmaktadır. Şekil 1.10'da görüldüğü gibi, ana düğümde çalışan, meta verileri saklayan tek bir ad düğümü ve her bir bağımlı düğümde çalışan, gerçek depolama işlerini yapan çok sayıda veri düğümü vardır. Düğümler raflar halinde düzenlenmekte ve hata toleransı sağlamak için küme içindeki farklı raflarda veri blokları çoğaltılmaktadır. HDFS'de bir dosyayı okumak veya yazmak için istemcinin ad düğümü ile etkileşime girmesi gerekmektedir, çünkü ad düğümü kümedeki merkez kısımdır. Veri düğümleri, canlı olduğunu göstermek için periyodik olarak ad düğümüne bir kalp atışı mesajı göndermektedir [60] [67].



Şekil 1.10. HDFS Mimarisi

1.4.2.1. HDFS Özellikleri

HDFS, Java'da yazılmış ve Apache tarafından geliştirilmiş bir dosya sistemidir. Çok büyük verileri saklamak için tasarlanmıştır. Tüm verileri bloklara ayırarak parçaları sunuculara dağıtmaktadır. Her blok birden fazla çoğaltılmata ve çoğaltmalar farklı düğümlerde saklanmaktadır. Bu, arızaların tespit edilmesini ve hızlı bir şekilde otomatik olarak kurtarmayı sağlamaktadır. HDFS, kullanıcıların tek bir adreslenebilir ad alanına sahip olmalarını sağlamakta; böylece tek bir büyük dosya sisteminden oluşan yüzlerce veya binlerce sunucudaki verilerin yönetimini kolay bir şekilde gerçekleştirmektedir [60].

1.4.2.2 Ad Düğümü

Tek bir ad düğümü, ana düğümde çalışmaktadır. Ad düğümü, dosya sistemindeki veriler hakkındaki meta verileri, fsimage adlı bir dosyada depolamakta ve yönetmektedir. Bu meta veriler, okuma / yazma isteklerinde istemcilere daha hızlı erişim sağlamak için ana bellekte önbelleğe alınmaktadırlar. Ad düğümü, HDFS'de bir dosya yöneticisi gibi davranmakta ve belli bir dosya için tüm blokların bulunduğu veri düğümlerini bilmektedir. Girdi / Çıktı görevlerini yürütmek için dosyaların nasıl bloklara ayrıldığını yönetmekte ve denetlemekte, dağıtılan dosya sisteminin genel durumu ve uygunluğuna göre, bu blokları hangi veri düğümlerde depolaması gerektiğini tanımlamaktadır [60][67].

1.4.2.3. Veri Düğümü

Hadoop kümesinde, her bağımlı düğümünde çalışan veri düğümü bileşenleri mevcuttur. Veri düğümleri, HDFS'nin depolanan dosyalarda veri bloklarını ve okuma / yazma isteklerini depolayan birincil depolama öğeleridir. Bunlar ad düğümü tarafından kontrol edilmektedir. Veri düğümlerinde depolanan bloklar, güvenilirlik ve yüksek kullanılabilirlik sağlamak için konfigürasyona göre çoğaltılmaktadır. Bu çoğaltılmış bloklar hızlı hesaplama sağlamak için kümeye dağıtılmaktadır [60][67].

1.4.3. NoSQL

Büyük verinin depolama sorununu çözmek için kullanılan önemli teknolojilerden biri de HDFS'ye alternatif olan NoSQL'dir. NoSQL, RDBMS modelinden önemli ölçüde farklı olan geniş bir DBMS sınıfındandır. NoSQL'in özelliklerine bakıldığında;

- İlişkisel olmayan, genellikle birleştirme işlemlerinden kaçınan veri modellerini desteklemekte ve sabit tablo şemaları gerektirmeyen verilerin depolanmasını sağlamaktadır. Grafik veritabanlarını da desteklemekte ve grafik yapısı üzerinde sorgulama yapılmasına izin vermektedir.
- Yapısal veya yapısal olmayan formatlarda (JSON, BSON veya XML) büyük hacimli verileri depolamaktadır ve esnek -ölçeklenebilir- bir yapıya sahiptir.
- Dağıtılmış ortamda yüksek kullanılabilirlik sağlamaktadır. Herhangi bir makinede bir arıza meydana geldiğinde, yürütülen işin sonlanması gerekmemektedir.
- RDBMS tarafından sağlanan ACID (Transaction ilkeleri; Atomicity -bütünlük-, Consistency -tutarlılık-, Isolation -bağımsızlık-, Durability -duyarlılık-) metodunu desteklemez, BASE (Basically Available - Kullanılabilirlik üzerine yoğunlaşmıştır- Soft state services with Eventual consistency -nihai veri tutarlıdır ancak, işlem sırasında veri tutarlılığını ve verilerin tamamına erişimi garanti etmez) metodunu desteklemektedir [68].

MongoDB, 10gen şirketi tarafından ticari olarak desteklenen açık kaynaklı bir NoSQL veritabanıdır. MongoDB ilişkisel olmamasına rağmen, sıralama, ikincil indeksleme ve aralık sorguları gibi ilişkisel veritabanlarının birçok özelliğini uygulamaktadır. MongoDB, sütun ve satır içeren tablolar kullanmamaktadır. Bunun yerine, veriler her biri skaler değerler, listeler veya iç içe geçmiş ilişkilendirilebilir dizilerden oluşan belgelerde saklanmaktadır [69].

Çizelge 1.1'de, HDFS ve MongoDB'nin veri erişimi ve güvenilirlik özellikleri gösterilmektedir. HDFS, nispeten büyük parçalar halinde sıralı okumalar ve veri yazmaları için optimize edilmiştir. MongoDB, rastgele ve paralel erişim, yani verilere yönelik sorgulamalar için optimize edilmiştir. MongoDB global yazma kilidi

nedeniyle paralel yazmalarda düşük performans göstermektedir. Hem MongoDB hem de HDFS, çoğaltma yoluyla veri güvenilirliği sağlamaktadır [69].

Çizelge 1.1. HDFS ve MongoDB kıyaslaması

Özellikler	HDFS	MongoDB
Depolama	Dağıtık Dosya Sistemi	Dağıtık şemasız veritabanı, Veriler RAM'de saklanmakta ve işlenmekte (in-memory)
Okuma	Sıralı, blok erişimi	Rastgele ve sıralı erişim (BTree dizini)
Yazma	Yerel olarak önbelleğe alınır daha sonra veri düğümüne gönderilir	Günlük olarak dizin ve bloklara yazılır, her sunucu yazma kilidine sahip (global write lock)
Güvenilirlik	Çoğaltma (replica)	Çoğaltma

1.4.4. Hadoop Sisteminde Sıkıntılar

1.4.4.1. Bellek Yapısı

Ana düğüm belleğinde HDFS'deki her dosya dizin ve blok olarak işlev görmektedir. Varsayılan boyutta HDFS bloğu 64 MB veya 128 MB'dir. Boyutu, HDFS'deki varsayılan blok boyutundan daha küçük olan dosyalar küçük dosyalar olarak adlandırılmaktadır. Bellek kullanımıyla ilgili olarak, HDFS'de çok sayıda küçük dosya varsa, bu sisteme ek bir yük oluşturmaktadır. HDFS, aşağıdaki nedenlerden dolayı çok sayıda küçük dosyayla iyi çalışmamaktadır:

1. Her bir blok, tek bir dosyayı işleme yeteneğine sahiptir, ayarlanmış blok boyutundan daha küçük dosyalar birçok bloğu işgal eder. Hem bu blokları birbiri ardınca okumak zaman almaktadır, hem de dosya sisteminde boş alanları kullanım dışı bırakmaktadır.

2. Ad düğümü her dosyanın kaydını tutmakta ve bu verileri belleğe kaydedip saklamaktadır. Çok sayıda dosya daha fazla bellek alanı gerektirmektedir [70].

HDFS'yi tasarlamanın birincil hedefi büyük dosyaların depolanması ve dosyalara erişilmesi içindir. HDFS, küçük dosyalara verimli bir şekilde erişmek için tasarlanmamıştır. Küçük dosyalar üzerinden okumak, her bir küçük veriyi almak için birçok veri arayışına sebep olur; bir veri düğümünden başka bir veri düğümüne atlamaya neden olmaktadır [71].

Bu mimari, küçük boyuttaki veriler için iyi çalışmadığı için Hadoop, büyük ölçekli verilerde kullanılmalıdır. Aksi takdirde verilerin analiz süresi uzayabilir, performans düşebilir [60].

1.4.4.2. Tek Ana Düğüm

HDFS, blok yapılı bir dosya sistemidir, "Chunk" adı verilen sabit boyutta bloklara ayrılmaktadır. Bir dosya birkaç parçadan oluşmakta ve farklı veri düğümlerinde saklanmaktadır. Hadoop, verileri ve veriler hakkındaki meta verileri için depolama alanlarını ayırmaktadır. Meta veriler, farklı veri düğümlerinde saklanan veriler hakkındaki bilgilerdir. Uygulanan tüm işlemlerden sonra HDFS'de meta veriler güncellenmektedir. Meta verilerini ana bellekte saklayan ad düğümü adlı tek bir sunucu vardır ve küme ile ilgili tüm bilgiler yalnızca ad düğümü aracılığıyla alınabilmektedir. Ad düğümü HDFS mimarisinin temel taşıdır; bu nedenle, ad düğümün güvenilirliği HDFS'de önemli bir değere sahiptir [60] [72].

Veri düğümü başarısız olduğunda, sadece bir makine çökecektir ve ad düğümü başarısız veri düğümün çalışmasını diğer mevcut veri düğüme yönlendirecektir [73]. Ancak ad düğümü çöktüğünde Hadoop devre dışı kalmaktadır [74].

Ad düğümü olmaksızın dosya sistemi kullanılamaz. Ad düğümü çalıştıran makine başarısız olursa, dosya sistemi üzerindeki tüm dosyalar kaybolur, çünkü dosyaları

veri düğümünde bloklardan yeniden nasıl yapılandırılacağı bilinmemektedir [75]. Bu nedenle, ana düğüm, Hadoopta tek bir hata noktası (SPOF) olarak görülmektedir[59].

1.4.4.3. Güvenlik/Gizlilik

Verilerin yakalanması ve bilginin elde edilmesi için analitik araçların kullanımı gizlilik konusunu gündeme getirmektedir. Bilgi yayıldığından ve dünya çapında çoğaltıldığından veri güvenliğini sağlamak ve gizliliği korumak son derece zordur. Verinin gizliliği, sistemler her gün büyük miktarlarda kişisel bilgi depoladıkça daha fazla iyileştirme gerektiren bir konu olmuştur [16].

Bu yeni dönemde birçok işletme, şirket ve müşterileri ile ilgili büyük miktardaki verilerini depolamak ve analiz etmek için büyük veri teknolojilerini kullanmaktadır, ancak özellikle güvenlik açısından temel varlıklara sahip olmayabilir. Büyük veri güvenliğini sağlamak için, şifreleme, günlüğe kaydetme gibi teknikler gerekmektedir.

Endüstrilerde ve federal organizasyonlarda sadece güvenlik değil, aynı zamanda veri gizliliği de problem olmaktadır. İş dünyasındaki büyük veri teknikleri ile, birçok şirket gizlilik sorunlarıyla da uğraşmaktadır. Veri gizliliği bir şirket için yükümlülüktür, bu nedenle şirketler gizliliği sağlamalıdır [14].

2. LİTERATÜR

Sriramoju A. Babu ve diğerleri (2017), geleneksel veritabanı sistemlerinin işlem kapasitesini aşan ve geleneksel veri işleme yöntemlerini kullanarak işlenemeyen büyük verilerden bahsetmiştir. Büyük verilerin hızla büyümesiyle ortaya çıkan veri tutarsızlığı ve eksikliği, ölçeklenebilirlik ve güvenlik gibi çeşitli sıkıntıları ele almışlardır. Bu makale, büyük veri teknolojisine ve çağdaş dünyadaki büyük verinin önemine kısa bir giriş niteliğindedir. Büyük veri teknolojisinde kullanılan araçları da ayrıntılı olarak tartışmışlardır [76].

Venkata Rami Redd ve diğerlerine göre (2015), büyük veri uygulamalarının en temel amacı, çok sayıda veriyi çok sayıda kaynak üzerinde depolamak ve yararlı bilgi elde etmek için işlemektir. Çalışmada araştırmacılar, çok düğümlü bir küme tasarlayarak hadoopun temel bileşenlerini uygulamışlardır; çok sayıda kaynaktan büyük veri depolamak için HDFS'yi kullanmışlar ve bu çoklu düğümlerde saklanan veriler üzerinde MapReduce programlama modelini uygulamışlardır [19].

Jens Dittrich ve diğerleri (2012), MapReduce ile veri yönetimi çözümlerini vurgulayarak, büyük veri işlemenin ana konularını ve zorluklarını belirtmişlerdir. İş optimizasyonu, fiziksel veri organizasyonu, veri düzenleri, indeksler vb. çeşitli mevzulardan bahsedilmiştir. Son olarak, benzerlik ve farklılıklarını vurgulayarak, Hadoop-MapReduce ve paralel DBMS arasında bir karşılaştırmalı analiz yapılmılştır [77].

Katarina ve diğerleri (2014), büyük veri projelerinin daha iyi planlanması ve yönetilmesi için büyük veriler hakkında genel bir bakış sağlamış, MapReduce'nin sorunlarını ve zorluklarını tanımlamışlardır. Tanımlanan zorluklar dört ana kategoride toplanmıştır: 1) veri depolama, 2) veri analizi, 3) çevrimiçi işlem, 4) güvenlik ve gizlilik. Ayrıca, MapReduce'nin belirlenen bu zorluklar için geliştirilmesine yönelik ilişkisel veritabanları ve NoSQL depoları, makine öğrenimi ve etkileşimli analiz gibi teknikler sunulmuştur [78].

Manolis Gergatsoulis ve diğerleri (2013), yaygın olarak kullanılan MapReduce programlama modelini kullanmışlar ve büyük miktarda bağlı veriyi sorgulamak için iki aşamalı bir MapReduce algoritması önermişlerdir. Algoritma, büyük verinin, bir dizi emtia bilgisayar kümesinin farklı düğümlerinde depolanabilen veri bölümlerine rasgele ayrılabileceği fikrine dayanmaktadır. Bir kullanıcı bir Q sorgusunu çalıştırdığında, bu sorgu alt sorgu kümesine ayrıştırılmaktadır. İlk adımda, alt sorgular düğümlerde izole bir şekilde yürütülerek ara sonuçlar elde edilmekte ve ikinci adımda, giriş sorgusuna nihai cevabı elde etmek için ara sonuçlar birleştirilmektedir. Bu araştırma çabasının getirdiği yenilik, önerilen sorgu algoritmasının, modelin tüm parametrelerinden bağımsız olması, yani, bağlı veriyi bölme, veriyi depolama, sorgu ayrıştırma mekanizması ve sorgu algoritmasının birbirinden bağımsız bir şekilde çalışmasıdır [79].

Subramaniyaswamy ve diğerleri (2015), yapısal olmayan verileri MapReduce tekniklerini kullanarak yapılandırmış ve işlemişlerdir. Kullanıcı zevklerini tahmin etmek için, işbirlikçi filtreleme (collaborative filtering) tekniği uygulanmıştır. MapReduce'nin, büyük miktarda veriyi işlemek için en etkili teknik olduğunu savunmuşlardır. İşbirlikçi filtreleme ve duyarlılık analizinin uygulanması, girdi olarak veri için öneri oluşturması sağlanmıştır. Tarih öncesinde kullanıcılar için öneriler oluşturulurken önbellek tabloları (cache table, Sıklıkla erişilen ancak değiştirilmeyen veri deposu) kullanılmaktaydı. Ancak günümüzde ifade tabanlı kümeleme ve etiketleme teknikleri kullanılarak, metodlar geliştirilmiştir. Önerilen üretim işleminin daha etkili olduğunı ve daha verimli bir şekilde optimize edildiğini savunmuşlardır [80].

Jacopo Urbani ve diğerleri (2013), MapReduce ile RDF işlemek için sıkıştırma algoritmalarının nasıl uygulanacağını incelemişler. RDF, veri modeli kullanılarak yayınlanan milyarlarca ifadenin bulunduğu semantik web'in özel durumunu işaret etmektedir. Büyük miktarda RDF verisini etkili bir şekilde sıkıştırmak için MapReduce algoritması tasarlamayı önermişlerdir. Sonuç olarak, Hadoop çerçevesi kullanarak bir prototip uygulanmıştır. Performansı, hem çeşitli girdi boyutlarıyla hem de farklı düğüm sayılarıyla test edilmiştir [81].

Kim, J. S. ve diğerleri (2016), DFS'ye entegre bir veritabanı kullanarak büyük veri analitiği için yeni bir yaklaşım önermişlerdir. Buna, PARADISE (İlişkisel DBMS İçin Paralel Analitik Çerçeve ve Tek Depolama Sistemine Entegre Dağıtılmış Dosya Sistemi) demişlerdir. Büyük verileri paralel olarak işlemek için sistemde, DFS ile entegre edilmiş bir MapReduce programlama modeli kullanılmıştır. Ayrıca MapReduce iş bölme sorunlarını çözmek için, mantıksal bölünme denilen uygun bir iş bölme yöntemi kullanılmıştır. Sistemde verileri tek yükleme ile, sorgu işlerinde performansın daha iyi ve karmaşık sorgu türlerinde başarılı olduğunu savunmuşlardır [82].

Sachin Bende ve diğerleri (2016), Hadoop Dağıtılmış Dosya Sisteminde küçük dosyalar sorunuyla ilgilenen yöntemlerin karşılaştırmalı analizini yapılmışlardır. HDFS'de bu problemle başa çıkmak için çeşitli yöntemler önermişlerdir. Bazı ortak parametrelere dayanarak, her bir dosyayı bir MapReduce görevine göndermenin çok fazla yüke yol açtığı için birden fazla dosyayı birleştirmeyi önermişlerdir. Sonuç olarak ek yük açısından en iyi performansı sergilediğini, ayrıca küçük dosyaların okuma verimliliğini de büyük ölçüde arttığını gözlemlemişler [71].

Priya ve diğerleri (2014), ad düğümü başarısızlığı altında Hadoop Dağıtılmış Dosya Sistemi mimarisini analiz etmiş ve elde ettikleri bulguları tartışmışlardır. Ad düğümü arızalarının üstesinden gelmek için, güvenilirliği ve hadoopun kullanılabilirliğini artıran bir mimari önermişlerdir. Mimarileri, diğer veri düğümleri üzerindeki ad düğümü sayısını artırmaktır. Bu mimari ile meta verilere erişimin kolaylaştığını, bununla da veri kaybını ve gecikmeyi azalttığını savunmuşlardır [72].

Shivam Gupta ve diğerleri (2018), büyük verinin 5V'sinin ele alınmasında istatistiksel rolüne ve ortaya çıkan zorluklara odaklanmışlardır. Bir kent seviyesinde çevresel izleme bağlamında büyük verilerle ilgili sıkıntılar kısaca sunulmuştur. Çevresel veri kaynaklarının mekânsal ve zamansal analizi için değişkenlerin ve konumların seçimini optimize etmeyi amaçlayarak iki iyi bilinen istatistiksel yöntemleri birleştirmeyi önermişlerdir. Arazi kullanım regresyonu (Land Use Regression, LUR) ve mekansal simüle tavlama (Spatial Simulated Annealing, SSA)

olan iki yöntemin bir arada kullanımı, veri edinme süreçlerinin tasarlanmasında yardımcı olacağını, böylece maksimum bilgi çıkarılabileceğini savunmuşlardır [20].

Saraladevi ve diğerleri (2015), dünya çapında kullanılan büyük verinin bilgilerini ve özelliklerini belirtmişlerdir. Büyük veride güvenliği artırmak için güvenlik konusuna daha fazla dikkat çekmişlerdir. Çok sayıda blok içeren Hadooptaki temel katman olan Hadoop Dağınık Dosya Sisteminde şu üç yaklaşımı birleştirerek büyük verilerin güvenliğini artırabilecek bir model önermişler; ilk yaklaşım, HDFS'de Kerberos'a - dağıtılmış servislere erişimi doğrulamak için kullanılan bir sistem- dayanmaktadır, veri bloklarına doğru ve aynı zamanda sadece yetkili bir kullanıcı tarafından erişilmesini sağlamaktır, ikinci yaklaşım ise, Bull Eye algoritma yaklaşımına dayanmaktadır, düğümden düğüme güvenlik düzenini sağlayıp ve düğümleri saldırılardan korumak için tüm açılardan taramaktır. Üçüncü yaklaşım, gelecekteki referanslar için sunucu çökmelerini azaltmak üzere bir ad düğümünü çoğaltarak güvenliğin gerçekleştirilmesini sağlamaktır [83].

Ibrahim Abaker ve diğerleri (2015), bulut bilişimde büyük verilerin kullanımının artışını gözlemlemişler ve büyük verilerde bulut hizmetleri için bir sınıflandırma modeli önermişlerdir. Bu modeli çeşitli bulut platformları ile karşılaştırmışlar ve hadoop teknolojisinin ve temel bileşenlerinin, yani MapReduce ve HDFS'nin arka planını tartışmışlardır. Güncel MapReduce projeleri ve ilgili yazılımları sunmuşlardır. Ölçeklenebilirlik, kullanılabilirlik, veri bütünlüğü, veri dönüşümü, veri kalitesi, veri çeşitliliği, gizlilik, yasal ve düzenleyici konular ve yönetim odaklı araştırma sıkıntılarını incelemişlerdir [7].

Philip Chen ve diğerleri (2014), bilgi ve iletişim teknolojisindeki (ICT, Information and Communication Technology) ilerlemelerin veri üretmeyi kolaylaştırdığını ve bulut bilişim tekniklerindeki hızlı gelişimin bu süreci hızlandırdığını, buna ek olarak bu tür tekniklerin veri depolamayı ve erişimini basitleştirebileceğini savunmuşlardır. 2011 yılına kadar veri analizi için araç veya teknoloji geliştirmeye yönelik büyük veri gibi bir terim yoktu. Ancak o zamandan itibaren, şirketler, hükümetler ve bilim adamları, büyük veriden anlam ve değer elde etmeye kendini adamışlardır. Chen'e göre ICT, büyük verinin gelişini hızlandırmıştır [84].

Ruchi ve diğerleri (2013), farklı boyutlarda girdi dosyalarını bulut tabanlı homojen -aynı özelliklerle konfigure edilmiş- ve heterojen -farklı özelliklerle konfigure edilmiş- hadoop kümelerinde Wordcount MapReduce uygulamasını test etmiş ve sonuçlarını gözlemlemişler. Homojen kümede, daha fazla düğüm ekleyerek belli bir eşik değerine kadar performansta bir artış sağlamadığı ancak bu değerden sonra veri düğümlerin sayısı arttıkça, hadoop kümesinin performansının arttığı sonucuna varmışlardır. Heterojen kümeler için, düğümlerin doğru kombinasyonu ile hadoop kümesinde performansın homojen olandan daha iyi olduğu, ancak yanlış kombinasyon performans düşüşüne ve veri düğümler arasında ek yüke neden olduğu sonucuna varmışlardır [85].

Parth Gohil ve diğerleri (2014), bulut tabanlı Hadoop kümesindeki büyük veri üzerinde Wordcount, Pi, Terasort ve Grep gibi çeşitli MapReduce uygulamalarının sonuçlarını incelemişler. MapReduce uygulamalarının sonuçlarının hadoop kümesi boyutuna bağlı olduğu; düğüm sayısı arttıkça, uygulamanın çalışma süresinin azaldığı ve performansın arttığı sonucuna varmışlardır [86].

Arun Devadiga ve diğerleri (2014), Cloudtack, Hadoop ve KVM'yi birlikte kullanmışlar. Bu entegrasyon, kullanıcıların eş zamanlı olarak çok miktarda veriyi verimli bir şekilde kullanmalarını sağlayacak olan sanal hadoop ile sonuçlanmıştır. Sanal hadoopun, yerel sunucudaki fiziksel hadooptan yürütme süresinin daha az olduğunu gözlemlemişlerdir. Daha kolay yönetilmesi, bilgisayar kaynaklarından tam olarak yararlanılması, daha güvenilir ve tasarruflu olmasıyla, sanal hadoopun daha avantajlı olduğu sonucuna varmışlardır [87].

Pratiksha Mandal ve diğerleri (2016), MapReduce'nin yeteneklerini özel -kişisel- bulutlara eklemeyi amaçlayan bir sistem önermişlerdir. Amazon EMR hizmetlerinin özel bulutlarda kullanılmasına izin vermemesi gibi Amazon EMR'nin eksiklerini belirtmişler, ayrıca hadoopun fiziksel sunucularda manuel olarak dağıtıldığı ve esneklikten yoksun olduğu sonucuna varmışlardır. Önerilen yöntem, dinamik esnekliğe sahip kümeleri desteklemekte ve aynı zamanda Amazon EMR ile aynı işlevselliği sağlamaktadır. Bununla, kullanıcıların bulut kaynak yönetimi ve maliyeti

konusunda endişelenmeden kendi bulutları için MapReduce hesaplama işlemlerini kullanabileceğini savunmuşlardır [88].

Weiyi Shang ve diğerleri (2013), büyük veri için bulut dağıtımlarını incelemişler. Çalışmada, büyük ölçekli ve sahte (pseudo - sözde bulut ortamında küçük bir veri örneği kullanarak geliştirdikleri ortam) bulut dağıtımları arasındaki farklılıkları ortaya çıkarmak için bir yaklaşım önermişler; yaklaşımları, bu platformlardan kolayca erişilebilen, ancak nadiren kullanılan günlük dosyalarını (log files) kullanmaktır. Üç farklı hadoop tabanlı BDA(büyük veri analitiği) Uygulaması yapmışlar ve iki ortamdaki BDA Uygulaması'nın yürütülmesindeki farkları belirtmişlerdir. BDA Uygulamaları için en uygun ve ekonomik platformu bulmak için, bir platformdan diğerine taşınması gerekebileceğini savunmuşlardır [89].

Ergüzen A. ve Erdal E. (2017), tıbbi görüntüleri ROI ve ROI-olmayan şeklinde kısımlara ayırmışlardır. ROI kısmında kayıpsız sıkıştırma uygulayarak, ROI-olmayan kısmında ise OCR ve Huffman algoritmalarını kullanarak verileri işlemişler ve depolamışlardır [90]. Daha sonra bu sistemi geliştirmek için araştırmacılar (2018), tıbbi görüntülerde Hadoop / MapReduce kullanarak sıkıştırma işlemi uygulamışlar ve elde edilen verileri MongoDB'de saklamışlardır. Geliştirdikleri sistemin, hızlı ve verimli arama ile güvenli bir şekilde verilere erişim sağladığını savunmuşlardır [91].

3. MATERYAL VE YÖNTEM

Çalışma Çizelge 3.1'de verilen iki farklı ortamda sağlanmıştır. Çalışmanın Hadoop Kümelemesi 64 GB RAM, 512 GB SSD ve 1 TB HDD'ye sahip Intel Xeon CPU E5-2620 v4 iş istasyonunda ve geliştirdiğimiz sistem 8 GB RAM ve 1 TB HDD'ye sahip olan bir kişisel bilgisayarda sağlanmıştır.

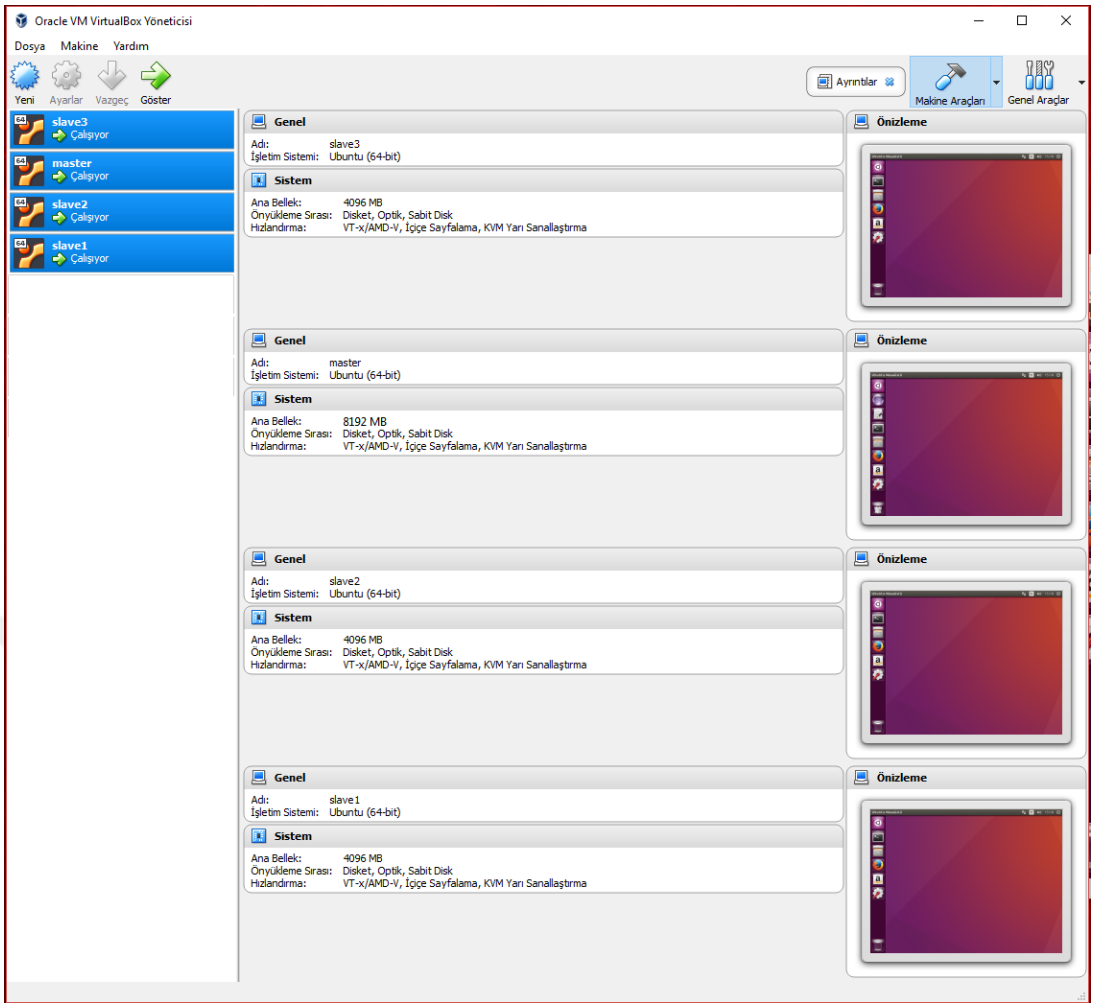
Çizelge 3.1. Kullanılan Bilgisayarlar

	Işletim sistemi	Ram	Harici disk	İşlemci
İş istasyonu	Windows 10	64 GB	1 TB HDD + 512 GB SSD	Intel Xeon E5
PC	Windows 7	8 GB	1TB SATA HDD	Intel i5

3.1. Hadoop/MapReduce Uygulaması

Çalışmaya, Şekil 3.1'de görülen Oracle Virtualbox ile Ubuntu işletim sisteminde dört adet sanal makine kurularak başlanmıştır.

Ubuntu Linux 16.04 LTS 64 Bit işletim sistemi olarak tercih edilmiş ve KVM sanallaştırma ortamı olarak seçilmiştir. Makinelerden biri ana düğüm, diğer üçü ise bağımlı düğüm olarak ayarlanmıştır. Bağımlı düğümler 4 GB bellek, ana düğüm ise 8GB bellek ve tüm düğümler Intel Xeon 2.10 GHz iki işlemci ve 20 GB SATA katı-hal-sürücüsü (SSD) olarak yapılandırılmıştır. Her bir düğümde, Java Geliştirme Kiti (JDK) jdk.1.8.0-161 sürümü ve Hadoop 2.7.2 kurulmuştur ve gerekli ayarlamalar yapılmıştır. Kesintisiz iletişimi sağlamak için şifresiz Güvenli Kabuk - Secure Shell (SSH) ayarları da yapılarak ana / bağımlı mimarisi ile Hadoop kümesi oluşturulmuştur.



Şekil 3.1. Oracle Virtualbox VM Arayüzü

Hadoop kümesini test etmek için MapReduce-Hadoop uygulamalarından bazıları kullanılmıştır.

1. TeraGen: Çok sayıda rastgele veri üretir ve genellikle sıralama ölçütleri (TeraSort vb.) için kullanılmaktadır. Üretilen verinin her satırı 100 Byte boyutundadır. TeraGen aynı zamanda herhangi bir girişe sahip olmayan ancak sabit boyutlu kayıtlardan oluşan büyük bir çıktı yazabilen tek bir harita-Hadoop uygulamasıdır. Verileri oluşturmak için harita görevlerini çalıştırmakta ve herhangi bir indirgeme görevi yürütmemektedir.

Şekil 3.2'de 1 GB veri üretimi yapan TeraGen uygulamasının çalışması gösterilmektedir. Girilen ilk parametre "-Dmapred.map.tasks = 10", 10 adet harita görevi ile çalışacağını ve ikinci parametre "10 Milyon (10.000.000)" ise satır sayısını göstermektedir.

```
root@namenode1:~# time hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapre
duce-examples-2.7.2.jar teragen -Dmapred.map.tasks=10 10000000 /example/data/1GB-input
18/04/03 03:01:25 INFO client.RMProxy: Connecting to ResourceManager at namenode1/192.1
68.56.101:8032
18/04/03 03:01:26 INFO terasort.TeraSort: Generating 10000000 using 50
18/04/03 03:01:26 INFO mapreduce.JobSubmitter: number of splits:50
18/04/03 03:01:26 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Inste
ad, use mapreduce.job.maps
18/04/03 03:01:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_152271083
5093_0012
18/04/03 03:01:27 INFO impl.YarnClientImpl: Submitted application application_152271083
5093_0012
18/04/03 03:01:27 INFO mapreduce.Job: The url to track the job: http://namenode1:8088/p
roxy/application_1522710835093_0012/
18/04/03 03:01:27 INFO mapreduce.Job: Running job: job_1522710835093_0012
18/04/03 03:01:36 INFO mapreduce.Job: Job job_1522710835093_0012 running in uber mode :
 false
18/04/03 03:01:36 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 03:01:58 INFO mapreduce.Job:  map 4% reduce 0%
18/04/03 03:02:02 INFO mapreduce.Job:  map 6% reduce 0%
18/04/03 03:02:04 INFO mapreduce.Job:  map 7% reduce 0%
18/04/03 03:02:06 INFO mapreduce.Job:  map 8% reduce 0%
18/04/03 03:02:08 INFO mapreduce.Job:  map 10% reduce 0%
18/04/03 03:02:10 INFO mapreduce.Job:  map 12% reduce 0%

18/04/03 03:03:18 INFO mapreduce.Job:  map 100% reduce 0%
18/04/03 03:03:18 INFO mapreduce.Job: Job job_1522710835093_0012 completed successfully
18/04/03 03:03:18 INFO mapreduce.Job: Counters: 32
        File System Counters
        File Output Format Counters
                Bytes Written=1000000000

real    1m56.210s
user    0m4.400s
sys     0m0.328s
```

Şekil 3.2. TeraGen Uygulamasının çalışması

2. TeraSort, belirli miktarda rastgele oluşturulmuş veriyi sıralamak için kullanılmaktadır. Bu kriter, bir Hadoop kümesinin HDFS ve MapReduce katmanlarının birlikte test edilmesini sağlamaktadır. Genel olarak, her sıra 100 Byte uzunluğundadır; Böylece, yazılan toplam veri miktarı, referans noktasının bir parçası olarak belirtilen satırların sayısının 100 katıdır (yani 1 GB veri yazmak için 10 milyon satır kullanılmaktadır).

Şekil 3.3'te, TeraGen tarafından üretilen 1 GB'lık rastgele veri için TeraSort uygulamasının çalışması gösterilmektedir.

```
root@namenode1:~# time hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapre
duce-examples-2.7.2.jar terasort  /example/data/1GB-input   /example/data/1GBout
18/04/05 03:03:59 INFO terasort.TeraSort: starting
18/04/05 03:04:02 INFO input.FileInputFormat: Total input paths to process : 2
Spent 331ms computing base-splits.
Spent 8ms computing TeraScheduler splits.
Computing input splits took 342ms
Sampling 2 splits of 2
Making 1 from 10 sampled records
Computing parititions took 717ms
Spent 1065ms computing partitions.
18/04/05 17:46:03 INFO client.RMProxy: Connecting to ResourceManager at master/192.168.
56.101:8032
18/04/05 03:06:07 INFO mapreduce.JobSubmitter: number of splits:2
18/04/05  03:06:07 INFO hdfs.DFSClient: Exception in createBlockOutputStream
18/04/03 03:06:22 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Inste
ad, use mapreduce.job.maps
18/04/03 03:06:22 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_152271083
5093_0014
18/04/03 03:06:22 INFO impl.YarnClientImpl: Submitted application application_152271083
5093_0014
18/04/03 03:06:23 INFO mapreduce.Job: The url to track the job: http://namenode1:8088/p
roxy/application_1522710835093_0014/
18/04/03 03:06:23 INFO mapreduce.Job: Running job: job_1522710835093_0014
18/04/03 03:06:31 INFO mapreduce.Job: Job job_1522710835093_0014 running in uber mode :
 false
18/04/03 03:06:31 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 03:06:47 INFO mapreduce.Job:  map 5% reduce 0%
18/04/03 03:06:48 INFO mapreduce.Job:  map 12% reduce 0%
18/04/03 03:06:49 INFO mapreduce.Job:  map 15% reduce 0%
18/04/03 03:06:50 INFO mapreduce.Job:  map 24% reduce 0%
18/04/03 03:14:51 INFO mapreduce.Job:  map 88% reduce 21%
18/04/03 03:15:00 INFO mapreduce.Job:  map 100% reduce 21%
18/04/03 03:14:51 INFO mapreduce.Job:  map 88% reduce 21%
18/04/03 03:15:00 INFO mapreduce.Job:  map 100% reduce 21%
18/04/03 03:15:07 INFO mapreduce.Job:  map 100% reduce 100%
18/04/03 03:15:08 INFO mapreduce.Job: Job job_1522710835093_0011 completed successfully
18/04/03 03:15:08 INFO mapreduce.Job: Counters: 49
```

Şekil 3.3. TeraSort Uygulamasının MapReduce Çalışması

3. WordCount, giriş dosyalarındaki kelimeleri sayan bir MapReduce programıdır. Program, her bir kelimenin büyük bir belge koleksiyonundaki oluşumlarını saymaktadır. Harita, anahtar / değer çiftelerini oluşturmaktadır. İndirgeme, tüm harita görevlerinden belirli bir kelime için sayıları toplamakta ve son sayımı vermektedir.

Şekil 3.4'te WordCount uygulamasının MapReduce çalışması gösterilmektedir.

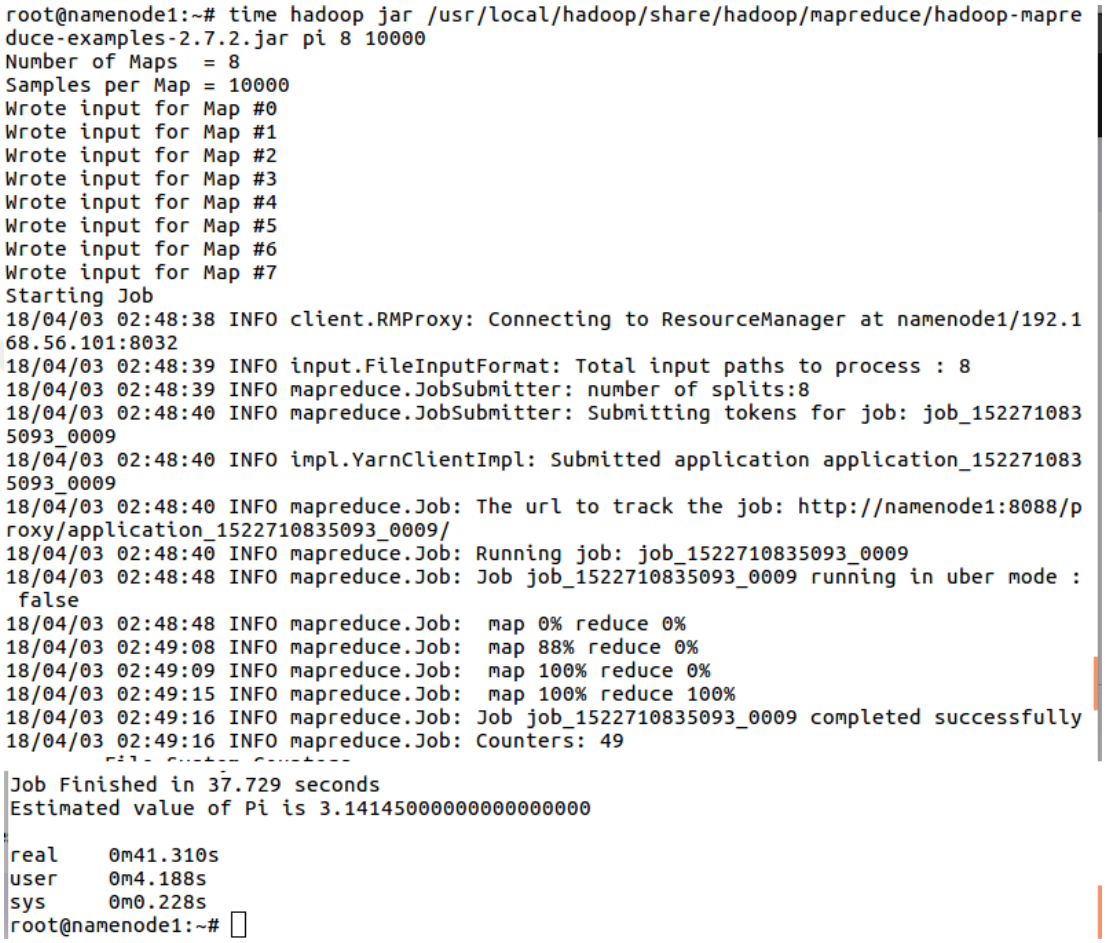
```
root@namenode1:~# time hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapre
duce-examples-2.7.2.jar wordcount /user/admin/wordcount_in /user/admin/wordcount_out
18/04/03 02:31:16 INFO client.RMProxy: Connecting to ResourceManager at namenode1/192.1
68.56.101:8032
18/04/03 02:31:17 INFO input.FileInputFormat: Total input paths to process : 1
18/04/03 02:31:17 INFO mapreduce.JobSubmitter: number of splits:1
18/04/03 02:31:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_152271083
5093_0004
18/04/03 02:31:18 INFO impl.YarnClientImpl: Submitted application application_152271083
5093_0004
18/04/03 02:31:18 INFO mapreduce.Job: The url to track the job: http://namenode1:8088/p
roxy/application_1522710835093_0004/
18/04/03 02:31:18 INFO mapreduce.Job: Running job: job_1522710835093_0004
18/04/03 02:31:26 INFO mapreduce.Job: Job job_1522710835093_0004 running in uber mode :
 false
18/04/03 02:31:26 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 02:31:32 INFO mapreduce.Job:  map 100% reduce 0%
18/04/03 02:31:39 INFO mapreduce.Job:  map 100% reduce 100%
18/04/03 02:31:39 INFO mapreduce.Job: Job job_1522710835093_0004 completed successfully
18/04/03 02:31:39 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=677
                FILE: Number of bytes written=238003
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=979
                HDFS: Number of bytes written=936
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=869
        File Output Format Counters
                Bytes Written=936

real    0m26.448s
user    0m3.928s
sys     0m0.148s
```

Şekil 3.4. Wordcount Uygulamasının MapReduce Çalışması

4. Pi, bir Quasi-Monte Carlo yöntemi kullanarak Pi değerini tahmin eden bir MapReduce programıdır. Harita görevleri birbirinden tamamen bağımsızdır ve tek bir indirgeme görevi, harita görevlerinden verileri toplamaktadır.

Bu programın çalışması, r yarıçaplı yani 2r kenar uzunluğuntaki bir karenin içindeki rastgele bir noktayı örnek seçmekte, daha sonra noktanın dairenin içinde olup olmadığını kontrol etmektedir. Çemberin içinde kaç tane bulunduğunu sayarak Pi değeri tahmin edilmektedir.

Uygulamada girilen ilk parametre harita sayısını, ikinci parametre ise her harita için kullanılacak örnek sayısını göstermektedir. Şekil 3.5'te Pi uygulamasının 8 harita görevi ile 10000 örneklemesinin çalışması gösterilmektedir.



Şekil 3.5. Pi Uygulamasının MapReduce Çalışması

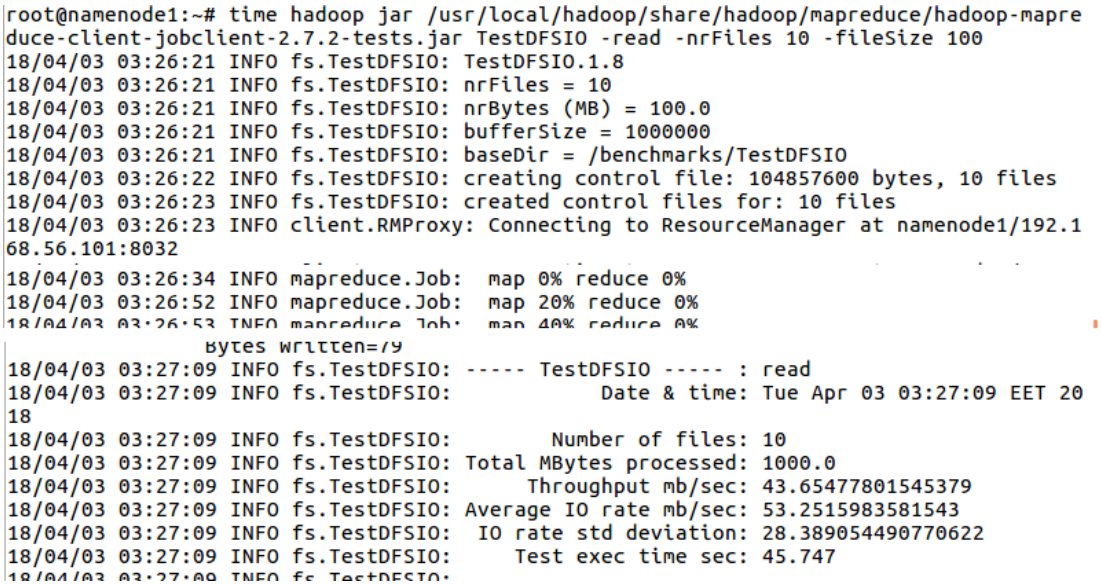
5. TestDFSIO programı, HDFS tarafından sağlanan toplu bant genişliğini hesaplamak için kullanılan Hadoop uygulamasıdır. Dosya sistemi için okuma ve yazma testidir. Test, okuma veya yazma işlemlerini paralel olarak gerçekleştiren çok sayıda görevi ele almaktadır. Bu test, her harita görevinin okuma veya yazma için bir dosya açtığı ve sonuçta aktarılan byte sayısını ve görevin yürütme süresini ölçtüğü bir MapReduce işi olarak çalışmaktadır. Harita görevlerini takiben, her harita görevi

için ortalama G / Ç oranı ve ortalama iş yükünü hesaplayarak tüm harita görevlerinden elde edilen sonuçları toplayan tek bir indirgeme görevi çalışmaktadır.

Şekil 3.6'da toplam 1 GB için 100 MB'lık 10 adet çıktı dosyası üreten bir yazma testi ve Şekil 3.7'de 1 GB'lik 10 adet giriş dosyasını kullanarak karşılık gelen okuma testini çalıştıran uygulmalar gösterilmektedir.

```
root@namenode1:~# time hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapre
duce-client-jobclient-2.7.2-tests.jar TestDFSIO -write -nrFiles 10 -fileSize 100
18/04/03 03:17:52 INFO fs.TestDFSIO: TestDFSIO.1.8
18/04/03 03:17:52 INFO fs.TestDFSIO: nrFiles = 10
18/04/03 03:17:52 INFO fs.TestDFSIO: nrBytes (MB) = 100.0
18/04/03 03:17:52 INFO fs.TestDFSIO: bufferSize = 1000000
18/04/03 03:17:52 INFO fs.TestDFSIO: baseDir = /benchmarks/TestDFSIO
18/04/03 03:17:54 INFO fs.TestDFSIO: creating control file: 104857600 bytes, 10 files
18/04/03 03:17:55 INFO fs.TestDFSIO: created control files for: 10 files
18/04/03 03:17:55 INFO client.RMProxy: Connecting to ResourceManager at namenode1/192.1

18/04/03 03:18:05 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 03:18:17 INFO mapreduce.Job:  map 20% reduce 0%
18/04/03 03:18:28 INFO mapreduce.Job:  map 20% reduce 7%
18/04/03 03:18:36 INFO mapreduce.Job:  map 30% reduce 7%
18/04/03 03:18:38 INFO mapreduce.Job:  map 30% reduce 10%
18/04/03 03:18:45 INFO mapreduce.Job:  map 70% reduce 10%
18/04/03 03:18:46 INFO mapreduce.Job:  map 77% reduce 10%
18/04/03 03:18:56 INFO mapreduce.Job:  map 87% reduce 10%
18/04/03 03:18:57 INFO mapreduce.Job:  map 100% reduce 13%
18/04/03 03:18:58 INFO mapreduce.Job:  map 100% reduce 100%
18/04/03 03:18:58 INFO mapreduce.Job: Job job_1522710835093_0016 completed successfully
18/04/03 03:18:58 INFO mapreduce.Job: Counters: 50

18/04/03 03:18:58 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
18/04/03 03:18:58 INFO fs.TestDFSIO:            Date & time: Tue Apr 03 03:18:58 EET 20
18
18/04/03 03:18:58 INFO fs.TestDFSIO:        Number of files: 10
18/04/03 03:18:58 INFO fs.TestDFSIO: Total MBytes processed: 1000.0
18/04/03 03:18:58 INFO fs.TestDFSIO:      Throughput mb/sec: 9.491898664489858
18/04/03 03:18:58 INFO fs.TestDFSIO: Average IO rate mb/sec: 17.345895767211914
18/04/03 03:18:58 INFO fs.TestDFSIO:  IO rate std deviation: 15.934490452176995
18/04/03 03:18:58 INFO fs.TestDFSIO:     Test exec time sec: 63.425
18/04/03 03:18:58 INFO fs.TestDFSIO:
```

Şekil 3.6. TestDFSIO İle Yazma İşleminin MapReduce Çalışması



Şekil 3.7. TestDFSIO İle Okuma İşleminin MapReduce Çalışması

6. Grep, verilen girdideki düzenli ifadelerin (regex) eşleşmelerini sayan bir MapReduce programıdır. Belgedeki bir modelin tüm oluşumlarını aramaktadır. Birçok veri analizinde arama aracı olarak kullanılmaktadır. Her harita görevi, modellerden birini içeren satırları çıktı olarak vermekte ve indirgeme görevi sayıları toplayarak çift olarak belirtmektedir.

Grep uygulamasında, Şekil 3.8'de girdideki "v" ile başlayan tüm ifadeleri sorgulayan çalışma görülmektedir.

```
root@namenode1:~# time hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapre
duce-examples-2.7.2.jar grep /user/admin/grep_in /user/admin/grep_out 'v[a-z.]+'
18/04/03 03:46:15 INFO client.RMProxy: Connecting to ResourceManager at namenode1/192.1
68.56.101:8032
18/04/03 03:46:17 INFO input.FileInputFormat: Total input paths to process : 1
18/04/03 03:46:17 INFO mapreduce.JobSubmitter: number of splits:1
18/04/03 03:46:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_152271083
5093_0027
18/04/03 03:46:18 INFO impl.YarnClientImpl: Submitted application application_152271083
5093_0027
18/04/03 03:46:19 INFO mapreduce.Job: The url to track the job: http://namenode1:8088/p
roxy/application_1522710835093_0027/
18/04/03 03:46:19 INFO mapreduce.Job: Running job: job_1522710835093_0027
18/04/03 03:46:28 INFO mapreduce.Job: Job job_1522710835093_0027 running in uber mode :
 false
18/04/03 03:46:28 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 03:46:33 INFO mapreduce.Job:  map 100% reduce 0%
18/04/03 03:46:39 INFO mapreduce.Job:  map 100% reduce 100%
18/04/03 03:46:40 INFO mapreduce.Job: Job job_1522710835093_0027 completed successfully
18/04/03 03:46:40 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=102
                FILE: Number of bytes written=237533

18/04/03 03:46:53 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 03:46:59 INFO mapreduce.Job:  map 100% reduce 0%
18/04/03 03:47:05 INFO mapreduce.Job:  map 100% reduce 100%
18/04/03 03:47:05 INFO mapreduce.Job: Job job_1522710835093_0028 completed successfully
```

Şekil 3.8. Grep Uygulamasının MapReduce Çalışması

7. RandomTextWriter, düğüm başına istenilen boyutta rasgele metin verisi yazan bir harita / indirgeme programıdır.

Şekil 3.9'da "-DMapReduce.randomtextwriter.totalbytes=1073741824" parametresi verilerek 1073741824 B = 1 GB veri yazmayı sağlayan RandomTextWriter uygulamasının çalışması gösterilmektedir.

```
root@namenode1:~# time hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapre
duce-examples-2.7.2.jar randomtextwriter -Dmapreduce.randomtextwriter.totalbytes=107374
1824 /home/test
18/04/03 03:54:05 INFO client.RMProxy: Connecting to ResourceManager at namenode1/192.1
68.56.101:8032
Running 1 maps.
Job started: Tue Apr 03 03:54:06 EET 2018
18/04/03 03:54:06 INFO client.RMProxy: Connecting to ResourceManager at namenode1/192.1
68.56.101:8032
18/04/03 03:54:08 INFO mapreduce.JobSubmitter: number of splits:1
18/04/03 03:54:08 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_152271083
5093_0031
18/04/03 03:54:08 INFO impl.YarnClientImpl: Submitted application application_152271083
5093_0031
18/04/03 03:54:08 INFO mapreduce.Job: The url to track the job: http://namenode1:8088/p
roxy/application_1522710835093_0031/
18/04/03 03:54:08 INFO mapreduce.Job: Running job: job_1522710835093_0031
18/04/03 03:54:17 INFO mapreduce.Job: Job job_1522710835093_0031 running in uber mode :
 false
18/04/03 03:54:17 INFO mapreduce.Job:  map 0% reduce 0%
18/04/03 03:54:48 INFO mapreduce.Job:  map 100% reduce 0%
18/04/03 03:54:48 INFO mapreduce.Job: Job job_1522710835093_0031 completed successfully
```

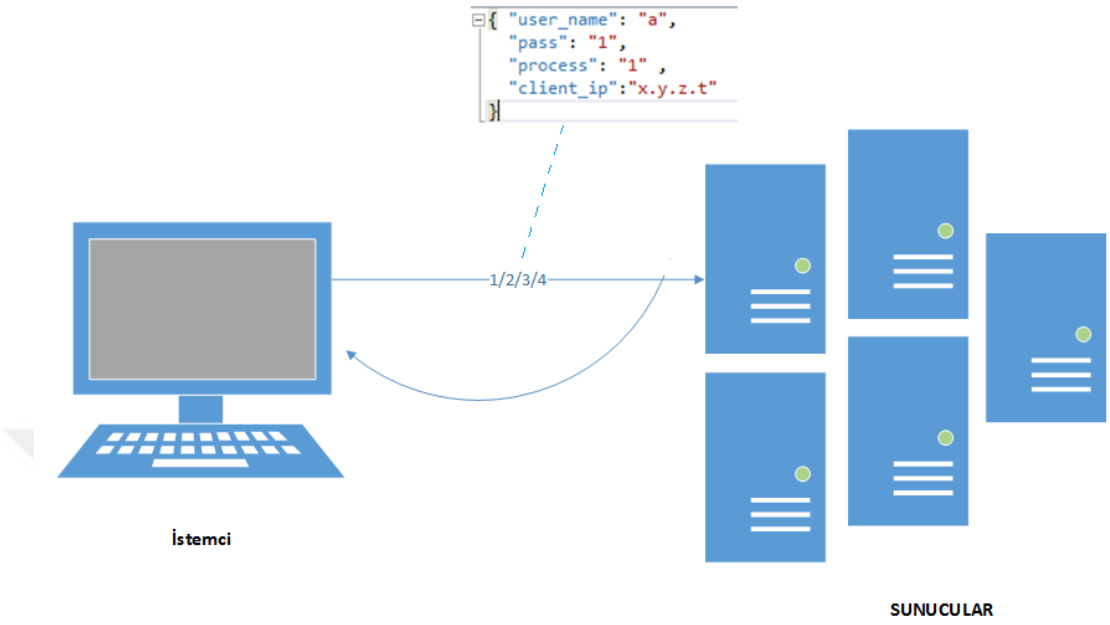
Şekil 3.9. RandomTextWriter Uygulamasının MapReduce Çalışması

3.2. Geliştirilen Sistem

Hadoop sistemindeki sıkıntılar göz önünde bulundurularak Ergüzen A., Ünver M. yeni bir dağıtılmış dosya sistemi önermişlerdir. Önerilen sistemde, Hadooptaki gibi ad ve veri düğümü ayrımı olmaksızın homojen bir küme tasarlanmıştır. Her düğüm, sunucu işlevinde kullanılarak kümede düğümlere düşen yük dengelenmiş olup sağlamlığı ve kullanılabilirliği sağlanmıştır. Sistemde veri güvenliği için simetrik şifreleme kullanılmıştır. Dosya yapısı ise 10 MB'lık bloklar halinde ayarlanmıştır. Böylelikle hem küçük hem de büyük dosyalar için verimli depolama sağlanmıştır. Ancak sistemdeki sabit blok yapısı iç bölünmeye sebep olmuştur [92]. Bu çalışmada, bu probleme çözüm olarak esnek blok yapılı yeni bir sistem geliştirilmiştir.

Bu sistem; Visual Studio ortamında, C# programlama dili kullanılarak geliştirilmiştir. Sistemde TCP protokolü üzerinden veri alışverişini sağlayan sunucu-istemci şeklinde soket programlama kullanılmıştır. Sunucu kısım [EK1] Windows Service ile istemci kısım [EK2] ise form uygulaması ile hazırlanmıştır.

3.2.1. Sistemin Çalışma Şekli



Şekil 3.10. Geliştirilmiş Sistemin Mimarisi

Güvenliği sağlamak için hibrit şifreleme yöntemi olan PGP açık kaynak kodlu yazılım kullanılmıştır. İstemci herhangi bir sunucuya veri gönderirken, sunucu düğümün genel anahtarını alıp veriyi bu anahtar ile şifreleyerek göndermektedir.

İstemci sunucu ile iletişimi 4 farklı işlem kodu ile sağlamaktadır. Kod yapısı Şekil 3.10'da görüldüğü gibi bir JSON dosyasıdır. Sunucu istemciden aldığı kod bloğunun *process*'ine bakarak hangi işlemi yapması gerektiğini anlamaktadır.

Aşamalar:

- İstemci, veri alışverişinde hangi sunucu ile iletişim sağlayacağını öğrenmek için 1 no'lu kodu herhangi bir sunucuya göndermektedir. Sunucu ise, istemciden 1 no'lu kodu aldığında istemcinin kendisinden bir IP talep ettiğini anlamakta ve kümeden herhangi bir sunucunun IP adresini istemciye göndermektedir.

- İstemci, sunucudan aldığı IP adesine bağlanarak veri alışverişi için 2 no'lu kod ile birlikte dosyayı göndermektedir. Sunucu ise, istemciden 2 no'lu kodu aldığında istemci ile bağlantı kurarak istemcinin gönderdiği veriyi almakta ve dosya sistemine kaydetmektedir. Daha sonra dosyayı kaydettiği bloğun başlangıç adresini istemciye göndermektedir.
- İstemci, dosya silmeyi talep ettiğinde sunucuya 3 no'lu kod ile silmek istediği dosyanın başlangıç adresini göndermektedir. Sunucu ise, istemciden 3 no'lu kodu aldığında istemcinin silmek istediği dosyayı öğrenmekte ve bellekten silmektedir.
- İstemci bir dosyayı almayı talep ettiğinde sunucuya 4 no'lu kod ile talep ettiği dosyanın başlangıç adresini göndermektedir. Sunucu istemciden 4 no'lu kodu aldığında ise istemciye istediği dosyayı göndermektedir.

Ayrıca sistem, Hadoop'taki ad düğümü sıkıtısına çözüm alternatifi olacak şekilde, tek bir ana düğüm ile çalışmamaktadır. İstemci tek bir düğüme bağlı olarak değil de istediği düğüm ile iletişime geçebilmekte ve veri alışverişini sağlayabilmektedir.

3.2.2. Dosya Yapısı

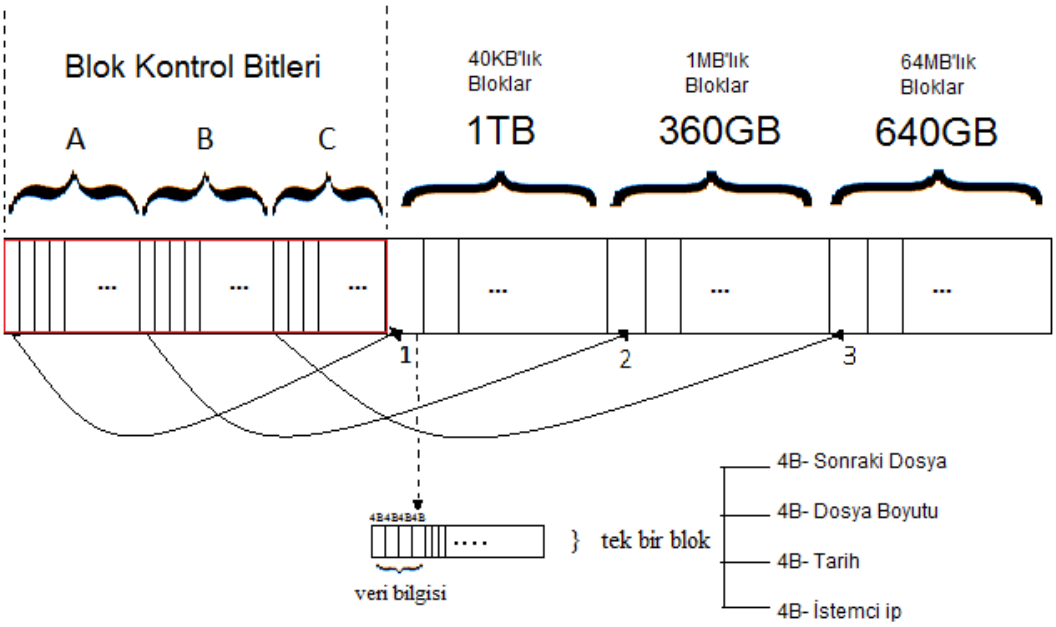
RDBMS yani ilişkisel veri tabanı yönetim sistemlerinden farklı olarak veriler, tek bir bilgisayarda değil de -her birinin kendine ait işlemcisi ve RAM'i olan- düğümlerde saklanmaktadır. Yani tüm veriler merkezde toplanmayıp ayrı ayrı düğümlere dağıtılmaktadır.

HDFS çok büyük dosyaları desteklemek için tasarlanmıştır. HDFS tarafından kullanılan tipik bir blok boyutu 64 MB'dir. Böylece, HDFS'de bir dosya 64 MB'lık parçalara bölünerek saklanmaktadır.

Buna alternatif olarak bizim sistemimizde blok boyutu 40 KB, 1 MB ve 64 MB olacak şekilde üç farklı türde (A, B, C) ayarlanmıştır. Burada 64 MB, Hadoop'taki gibi büyük dosyalar için düşünülürken, 40 KB küçük metin dosyaları için ve 1 MB

ise buna nazaran daha büyük metin dosyaları için ortalama bir boyut olarak düşünülmüştür. Şekil 3.11'de A, B, C, bitleri temsil etmektedir. Her bir bit (1 ve 0) bir bloğa bakmakta ve bloğun dolu olup olmadığını göstermektedir.

Her blok 16 Byte veri bilgisini saklamaktadır. 4 Byte'si dosyanın devamı olan pozisyona, 4 Byte'si dosyanın boyutuna, 4 Byte'si istemci IP'sine, 4 Byte'si ise tarihe reserve edilmiştir.



Şekil 3.11. Geliştirilmiş Sistemin Dosya Yapısı

Toplam dosya boyutu 2 TB'dir. Çizelge 3.2'de gösterildiği gibi bu 2 TB'lik dosyanın 1 TB'si 40 KB'lik 25000000 adet bloktan oluşan A tipi, 360 GB'si 1 MB'lik 360000 adet bloktan oluşan B tipi ve 640 GB'si 64 MB'lik 10000 adet bloktan oluşan C tipi için ayarlanmıştır.

Çizelge 3.2. Dosyanın Blok Yapısı

	A tipi	B tipi	C tipi
Blok boyutu	40 KB	1 MB	64 MB
Bit(Blok) sayısı	25000000 adet	360000 adet	10000 adet
Dosya boyutu	1 TB	360 GB	640 GB

Bir dosya girişi yapıldığında, dosya boyutuna göre bu üç türde kaç blok yer tutacağına bakılmakta ve en az blok hangisi ise o türde gösterdiği bitten itibaren dosyayı yazmaya başlamaktadır.

Örneğin;

- 100 MB veri girişi yapıldığında A tipinde 2500 bloğa, B tipinde 100 bloğa ve C tipinde 2 bloğa ihitiyaç vardır. Dosya en az bloğa ihtiyacı olan C tipinde ve 3. pozisyondan itibaren yazılacaktır.
- 100 KB veri girişi yapıldığında ise A tipinde 3 bloğa ihtiyaç vardır, B ve C tipinde 1 blok yeterlidir, ancak bloğun çok büyük kısmını kullanım dışı bırakıp bellekte boş yer tuttuğu için B ve C tiplerinde değil, A tipinde yazılacak olup 1. pozisyondan itibaren yazılacaktır.

4. ARAŞTIRMA BULGULARI

4.1. Hadoop

1. TeraGen uygulamasında, kümede sırasıyla 2, 10, 25, 50 harita kullanılarak 1 GB rastgele veri üretilmiştir. Elde edilen dosyaların çalışma süreleri Şekil 4.1'de gösterilmektedir. Test sonucuna bakıldığında, harita sayısı arttığında çalışma süresinin arttığı görülmektedir. 1 GB'lık dosyanın 10 harita görevi ile çalıştırılmış hali Hadoop Dosya Sisteminde Şekil 4.2'de gibi görülmektedir.

	Name	User	Type	Status	Progress	Group	Started	Duration
plication_1522710835093_0013	TeraGen	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:03	27.37s
plication_1522710835093_0014	TeraGen	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:06	41.42s
plication_1522710835093_0015	TeraGen	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:07	1m, 10s
plication_1522710835093_0012	TeraGen	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:01	1m, 49s

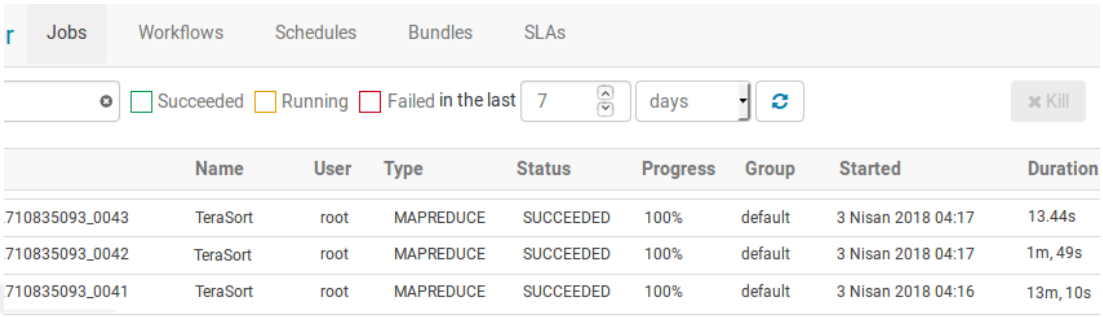
Şekil 4.1. TeraGen Uygulamasının Çalışma Süreleri

Home / example / data / 1GB-input2 History

Name	Size	User	Group	Permissions	Date
_SUCCESS	0 bytes	root	supergroup	-rw-r--r--	April 02, 2018 05:07 PM
part-m-00000	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00001	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00002	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00003	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:07 PM
part-m-00004	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00005	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00006	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:07 PM
part-m-00007	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00008	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:06 PM
part-m-00009	95.4 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:07 PM

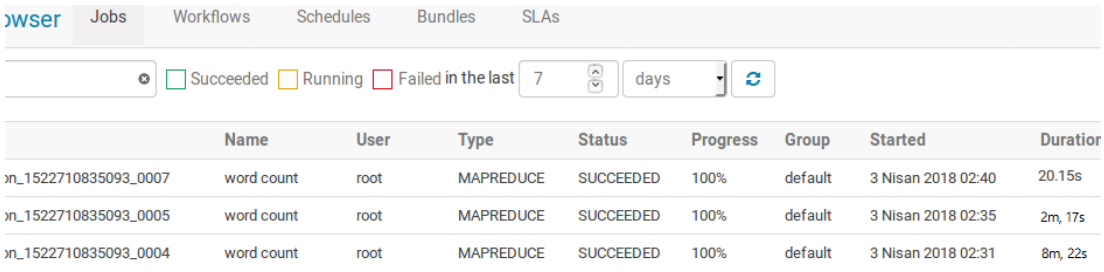
Şekil 4.2. HDFS'de TeraGen Uygulamasıyla 10 Harita ile Elde Edilen 1GB'lık Dosya

2. TeraSort'u çalıştırırken sırasıyla 1, 2, 5 indirgeme parametresi kullanılmıştır. TeraGen ile oluşturulmuş 1 GB'lık rastgele veri TeraSort ile sıralanmıştır. Şekil 4.3'te TeraSort uygulamasının çalışma süreleri verilmiştir. Çalışma sürelerine bakıldığında indirgeme sayısı arttıkça performansın düştüğü görülmektedir.

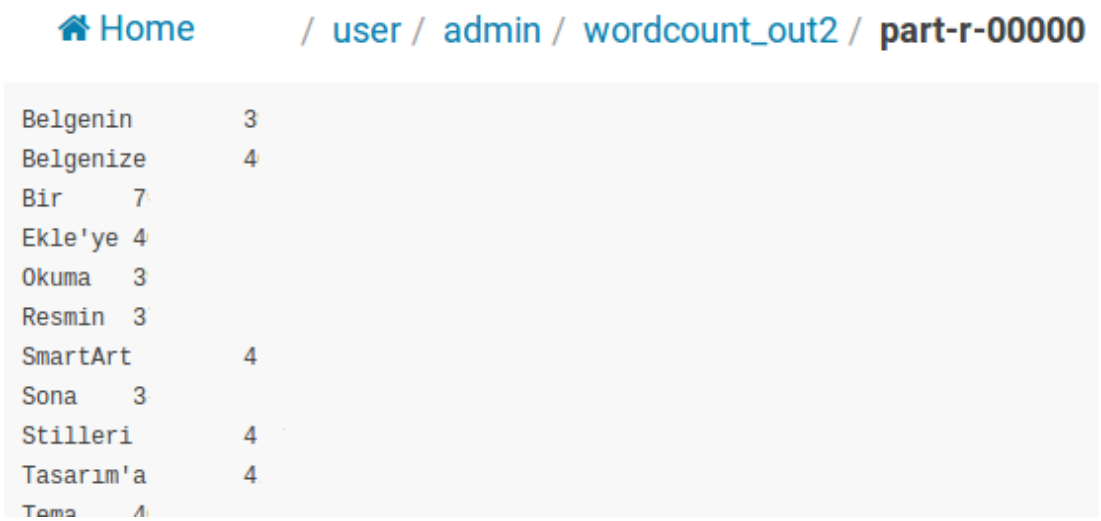


Şekil 4.3. TeraSort Uygulamasının Çalışma Süreleri

3. WordCount uygulaması, sırasıyla 100 MB, 1 GB, 2 GB boyutlarında dosyalar üzerinde test edilmiştir. Çalıştırılan WordCount uygulamalarının çalışma süreleri Şekil 4.4'te verilmiştir. Dosya boyutu arttıkça çalışma süresinin arttığı görülmektedir. Şekil 4.5'te 100 MB olan dosyanın çıktısının bir kısmı verilmiştir.



Şekil 4.4. WordCount Uygulamasının Çalışma Süreleri



Şekil 4.5. WordCount Uygulamasının Program Çıktısı

4. Pi uygulamasının testi için sırasıyla 1, 2 ve 4 adet düğüm kullanılmıştır. Tüm testlerde 8 adet harita görevi ve 1000 adet örnek kullanılmıştır. Çalıştırılan Pi uygulamasının çalışma süreleri Şekil 4.6'da verilmiştir. Sonuçlara göre, düğüm sayısı arttıkça çalışma süresinin azaldığı performansın arttığı görülmektedir.

d	Name	User	Type	Status	Progress	Group	Started	Durati
ıpplication_1522710835093_0011	QuasiMonteCarlo	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 02:50	42.25s
ıpplication_1522710835093_0010	QuasiMonteCarlo	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 02:49	33.64s
ıpplication_1522710835093_0009	QuasiMonteCarlo	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 02:48	21.38s

Şekil 4.6. Pi Uygulamasının Çalışma Süresi

5. TestDFSIO uygulaması için, 64 MB, 128 MB ve 256 MB şeklinde farklı blok boyutlarında testler yapılmıştır. Testlerde 1000 MB boyutunda 10 adet harita görevi ile 1000'er MB 10 adet dosya oluşturulmuştur. Şekil 4.7'de çalıştırılan TestDFSIO uygulamasının çalışma süreleri gösterilmektedir. 128 MB blok büyüklüğünde 64 MB'den yüksek hızda iken, 256 MB ile yakın bir çalışma süresi elde edilmiştir. 128 MB ve 256 MB blok boyutunda zaman içinde daha iyi performans alınmaktadır.

Şekil 4.8'de HDFS'deki 10 adet harita ile oluşturulan 1000 MB'lık dosyalar gösterilmektedir.

	Name	User	Type	Status	Progress	Group	Started	Duration
1835093_0024	hadoop-mapreduce-client-jobclient-2.7.2-tests.jar	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:32	2m, 20s
1835093_0023	hadoop-mapreduce-client-jobclient-2.7.2-tests.jar	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:30	30.56s
1835093_0022	hadoop-mapreduce-client-jobclient-2.7.2-tests.jar	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:28	42.27s

Şekil 4.7. TestDFSIO Yazma Uygulmalarının Çalışma Süreleri

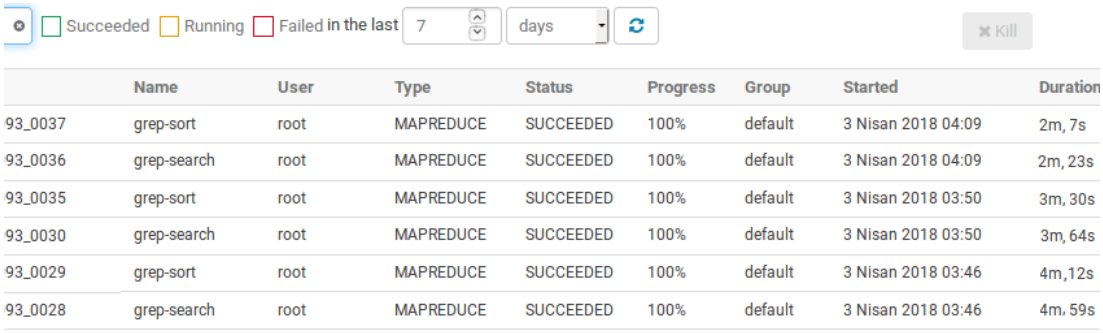
Home / benchmarks / TestDFSIO / **io_data** History Tras

Name	Size	User	Group	Permissions	Date
..		root	supergroup	drwxr-xr-x	April 02, 2018 05:32
.		root	supergroup	drwxr-xr-x	April 02, 2018 05:24
test_io_0	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_1	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_2	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:24
test_io_3	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_4	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_5	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_6	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_7	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:24
test_io_8	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25
test_io_9	1000.0 MB	root	supergroup	-rw-r--r--	April 02, 2018 05:25

Şekil 4.8. TestDFSIO ile Elde Edilen Dosyalar

6. Grep uygulaması, 100 MB, 500 MB ve 1 GB dosya boyutlarında test edilmiştir. Çalışma sırasında önce belge içinde sıralama (sort) işlemi, daha sonra istenilen ifadenin arama (search) işlemi gerçekleşmektedir. Şekil 4.9'da çalıştırılan Grep uygulamasının çalışma süreleri gösterilmektedir. 100 MB'lık dosyanın Grep çıktısı

ise Şekil 4.10'da verilmiştir. Test sonuçlarına göre dosya boyutu arttıkça işlem süresinin arttığı görülmektedir.



	Name	User	Type	Status	Progress	Group	Started	Duration
93_0037	grep-sort	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 04:09	2m, 7s
93_0036	grep-search	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 04:09	2m, 23s
93_0035	grep-sort	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:50	3m, 30s
93_0030	grep-search	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:50	3m, 64s
93_0029	grep-sort	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:46	4m,12s
93_0028	grep-search	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:46	4m, 59s

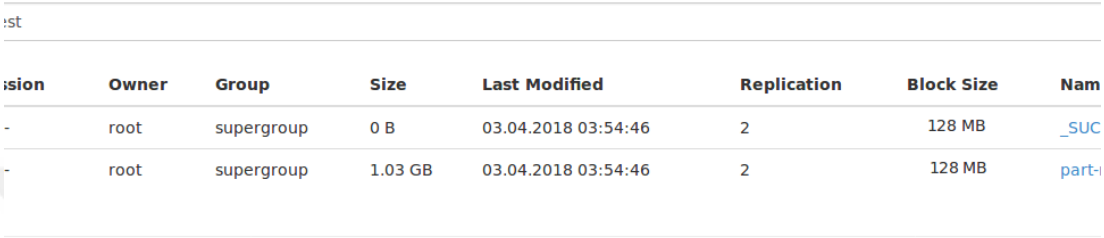
Şekil 4.9. Grep Uygulmalarının Çalışma Süreleri

Şekil 4.10. Grep Uygulamasının Çıktısı

7. RandomTextWriter uygulamasında, veri üretimi sağlanırken 1, 2 ve 3 farklı çoğaltmalarda (replica) test edilmiştir. Düğümler arasındaki boru hattının aktarımı ve yazma süresi nedeniyle, blokları 1'den fazla düğüme çoğaltırken önemli bir fark yaratmıştır. G/Ç oranı, 1 çoğaltma ile daha yüksek bir hıza sahip olmuştur. Şekil4.11'de çalışma süreleri gösterilmektedir. Testlerde 1 GB veri üretimi sağlanmıştır. Şekil 4.12'de üretilen dosyanın HDFS'deki görünümü gösterilmektedir.

	User	Type	Status	Progress	Group	Started	Duratior
ent-2.7.2-tests.jar	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:32	32.51s
ent-2.7.2-tests.jar	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:30	2m, 7s
ent-2.7.2-tests.jar	root	MAPREDUCE	SUCCEEDED	100%	default	3 Nisan 2018 03:28	2m, 49s

Şekil 4.11. RandomTextWriter uygulamasının çalışma süreleri



sion	Owner	Group	Size	Last Modified	Replication	Block Size	Nam
-	root	supergroup	0 B	03.04.2018 03:54:46	2	128 MB	_SUC
-	root	supergroup	1.03 GB	03.04.2018 03:54:46	2	128 MB	part-r

Şekil 4.12. HDFS'de 2 Çoğaltmalı Dosyanın Görünümü

5. SONUÇ

Şu anda yaşamakta olduğumuz bilgi çağında her gün yüksek hızda, büyük hacimde ve çeşitli türde veriler üretilmektedir. Bu verileri eski tekniklerle depolamak ve işlemek zor bir hal almıştır. Bu nedenle değerli bilgileri açığa çıkarmak için veri üzerinde gelişmiş analitik teknikler uygulanmaktadır. Bu teknikler, geleneksel sistemler kullanılarak depolanıp işlenemeyen son derece büyük ve karmaşık veri kümelerinin depolanmasına ve işlenmesine yardımcı olacak şekilde tasarlanmıştır. Hadoop MapReduce programlama modeli ve HDFS, büyük ve yapılandırılmamış veri kümelerini işlemek için giderek daha fazla kullanılmaktadır.

Bu tezde, son zamanlarda oldukça fazla ilgi toplayan büyük veri kapsamlı bir şekilde incelenmiştir. Büyük veri ile ilgili temel kavramlar, karakteristik özellikleri, kullanım alanları, uygulama biçimleri sunulmuştur. Büyük verinin sağladığı avantajlarının yanı sıra karşılaştığı zorluklarından da bahsedilmiştir. Daha sonra sanallaştırma platformu kullanılarak 4 düğümlü bir küme oluşturulmuştur. Bu düğümlerde, farklı veri kümeleri üzerinde testler gerçekleştirilmiş ve Hadoop çerçevesi altında mevcut olan kıyaslama araçları ile çeşitli parametreler uygulanarak performans analizi yapılmıştır. Hadoop kullanımı sırasında, iş uygulamasının performansını düşüren bazı dezavantajlar gözlemlenmiştir.

HDFS'nin varsayılan blok büyüklüğü 64 MB'dir. Her dosya bir blokta saklanır ve blok meta verileri ad düğümü tarafından hafızada tutulur. Ad düğümü üzerinden çok sayıda küçük boyutlu dosyaları depolamak çok fazla bellek tüketir, bu nedenle küçük dosyaları depolamak ve ayrıca tek bir düğüm ile etkileşimde bulunmak HDFS'de sistem ve depolama yığınına sebep olur. Ayrıca HDFS'de çoğaltma faktörü yapılandırılabilir. Ancak sistem yöneticileri tarafından sabit olarak bir kez verilir. Çoğaltma faktörünü belirlemek, uygun bir seçenek değildir; bir dosyaya kısa bir süre içinde tekrar tekrar erişilecekse, daha büyük bir çoğaltma faktörü tercih edilirken, nadiren erişilen arşiv verileri için, daha küçük bir çoğaltma faktörü yeterlidir ve depolama alanından tasarruf sağlanır. Ayarlama, blok veya dosya düzeyinde uygulanabilir.

Sonuç olarak sistem yapılandırmasında ve blok yerleşim durumundaki verimsizliği dikkate alarak HDFS'deki başarısızlığın üstesinden gelmek için, güvenilirliği ve kullanılabilirliği artıran bir mimari önerilmiştir. Sistemimiz tek bir ana düğümle çalışmamakta istemci herhangi bir sunucu ile etkileşime geçebilmektedir, böylelikle ana düğüm sıkıntılarına çözüm sağlanmıştır. Sunucu ve istemci arasında hibrit şifreleme tekniği kullanılarak veri alışverişi sağlanmakta ve veri gizliliği güvence altına alınmıştır. Ayrıca dosya sistemi tek bir blok boyutuna sahip olmayıp üç farklı boyutlarda ayarlanmış ve sadece büyük dosyalar için değil küçük dosyalar için de kullanımı kolaylaştırılmıştır.

Bu dosya sistemi, bir Hadoop kümesinde de yararlı olabilir. "Birden fazla blok boyutlu ve ana düğümsüz bir sistem" prensibini uygulamak, MapReduce'un iş planlayıcısının, hesaplamayı başlatmak için muhtemelen tek düğüm yükünü azaltacak ve kümelenme boyunca hesaplamaları daha kolay paylaşacaktır. Gelecekteki çalışmalar ile daha geniş bir HDFS konfigürasyonu içererek iş performanslarının iyileştirilmesi de sağlanabilir.

KAYNAKLAR

[1] Thomas H. Davenport, Paul Barth ve Randy Bean, "How 'Big Data' Is Different". *Mit Sloan Management Review*, 54(1): 22-24, 2012

[2] Vasant Dhar, "Data Science And Prediction" Communications of the ACM, Doi:10.1145/2500499. 56(12): 64-73, 2013

[3] David Reinsel, John Gantz, John Rydning, "Data Age 2025: The Evolution Of Data To Life-Critical, Don't Focus on Big Data", IDC White Paper Nisan 2017

[4] Samson Oluwaseun Fadiya, Serdar Saydam, Vanduhe Vany Zira, "Advancing Big Data For Humanitarian Needs" doi: 10.1016/j.proeng.20 Procedia Engineering 78: 88 – 95, 2014

[5] "Digital in 2018", https://wearesocial.com/blog/2018/01/global-digital-report-2018 (Erişim Tarihi: 27.03.2018)

[6] "What Happens in an Internet Minute in 2017" https://www.weforum.org/agenda/2017/08/what-happens-in-an-internet-minute-in-2017 (Erişim Tarihi: 27.03.2018)

[7] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, Samee Ullah Khan, "The Rise of 'Big Data' on Cloud Computing:Review and Open Research Issues" Information Systems 47: 98–115, 2015

[8] Charles A. Mathes, "Big Data Has Unique Needs For Information Governance" *Journal of Management Science and Business Intelligence*, 1(1):12-20, 2016

[9] "Understanding and Auditing Big Data" https://www.iia.nl/sitefiles/publicaties/gtag-understanding-and-auditing-big-data.pdf (Erişim Tarihi: 27.03.2018)

[10] Gali Halevi, Henk F. Moed, "The Evolution Of Big Data As A Research And Scientific Topic: Overview Of The Literature" https://www.researchtrends.com/issue-30-september-2012/the-evolution-of-big-data-as-a-research-and-scientific-topic-overview-of-the-literature/ (Erişim Tarihi: 27.03.2018)

[11] Gil Press, "A Very Short History Of Big Data" Aralık 2013 https://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/2/ (Erişim Tarihi: 27.03.2018)

[12] Ramesh Dontha, "The Origins Of Big Data" https://www.kdnuggets.com/2017/02/origins-big-data.html Şubat 2017 (Erişim Tarihi: 27.03.2018)

[13] "Google Trends: Bigdata arama sonucu" https://trends.google.com/trends/explore?date=2008-03-27%202018-03-27&q=big%20data (Erişim Tarihi: 27.03.2018)

[14] M. Moorthy, R. Baby, S. Senthamaraiselvi "An Analysis For Big Data And Its Technologies" *International Journal of Computer Science Engineering and Technology(IJCSET)* 4(12):412-418, 2014

[15] Natalija Koseleva, Guoda Ropaite, "Big Data In Building Energy Efficiency: Understanding Of Big Data And Main Challenges" Procedia Engineering 172: 544 – 549, 2017

[16] Pedro Caldeira Neves, Bradley Schmerl, Jorge Bernardino ve Javier Cámara, "Big Data In Cloud Computing: Features And Issues" *International*

Conference on Internet of Things and Big Data, DOI: 10.5220/0005846303070314. 1: 307-314, 2016

[17] Doug Laney, "3D Data Management: Controlling Data Volume, Velocity, and Variety", http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf Technical report, META Group, 2001

[18] Avita Katal, Mohammad Wazid, R H Goudar, "Big Data: Issues, Challenges, Tools And Good Practices" *Conference: Contemporary Computing (IC3), Sixth International Conference,* DOI: 10.1109/IC3.2013.6612229, 2013

[19] Venkata Rami Redd, Srikanth Kumar, "Hadoop Distributed File System And Map Reduce Processing On Multi-Node Cluster" *International Journal of Science and Research (IJSR)* 4 (8):1424-1430, 2015

[20] Shivam Gupta, Jorge Mateu, Auriol Degbelo, Edzer Pebesma, "Quality Of Life, Big Data And The Power Of Statistics" *Statistics & Probability Letters* 136(2018):101-104, 2018

[21] Yuri Demchenko, Paola Grosso, Cees De Laat, Peter Membrey, "Addressing Big Data Issues In Scientific Data Infrastructure" *International Conference On Collaboration Technologies And Systems (CTS),* San Diego, Ca. doi:10.1109/Cts.2013.6567203, 2013

[22] Anett Hoppe, "Automatic Ontology-Based User Profile Learning From Heterogeneous Web Resources In A Big Data Context" *The 39th International Conference On Very Large Data Bases* 6(12):1428-1433, 2013

[23] Gema Bello-Orgaz, Jason J.Jung, DavidCamacho, "Social Big Data: Recent Achievements And New Challenges", Information Fusion 28: 45-59, 2016

[24] Sharona Hoffman, Andy Podgurski, "Big Bad Data: Law, Public Health, And Biomedical Databases" *Public Health Law Conference: Practical Approaches to Critical Challenges*, 1:56-60, 2013

[25] Ping Jiang, Jonathan Winkley, Can Zhao, Robert Munnoch, Geyong Min, Laurence T. Yang "An Intelligent Information Forwarder For Healthcare Big Data Systems With Distributed Wearable Sensors". IEEE Systems Journal, Doi:10.1109/Jsyst.2014.2308324. 10(3): 1147-1159, 2014

[26] Interim Progress Report: Whitehouse, "Big Data: Seizing Opportunities, Preserving Values" https://obamawhitehouse.archives.gov/sites/default/files/docs/20150204_big_data_seizing_opportunities_preserving_values_memo.pdf (Erişim Tarihi: 27.03.2018)

[27] Steve Lavalle, Eric Lesser, Rebecca Shockley, Michael S. Hopkins, Nina Kruschwitz, "Big Data, Analytics and The Path From Insights To Value" Mit Sloan Management Review, 52(2):21-32, 2011

[28] Paul Tallon "Corporate Governance Of Big Data: Perspectives On Value, Risk, And Cost" Computer Doi: 10.1109/Mc.2013.155. 46(6):32-38, 2013

[29] Cynthia Beath, Irma Becerra-Fernandez, Jeanne Ross, James Short , "Finding Value In The Information Explosion" Mit Sloan Management Review, 53(4): 18-20, 2012

[30] Issenberg, S. "How President Obama's Campaign Used Big Data To Rally Individual Voters". Mit Technology Review. 116(1):38-49, 2012

[31] Harris, S. "Congressional Calls Grow For A Probe Of Russian Hacking In U.S. Election". Wall Street Journal. http://www.wsj.com/articles/congressional-calls-grow-for-a-probe-of-russian-hacking-in-u-s-election-1481151191 (Erişim Tarihi: 27.03.2018)

[32] Nate Lord , "What Is Data Encryption?", Son Güncelleme: 15Ocak 2018, https://digitalguardian.com/blog/what-data-encryption (Erişim Tarihi: 27.03.2018)

[33] Christine Taylor, "Big Data Security" https://www.datamation.com/big-data/big-data-security.html 27 Haziran 2017 (Erişim Tarihi: 27.03.2018)

[34] Robbi Rahim, M Mesran, Muhammad Syahrizal, Andysah Putera, Utama Siahaan, "Data Security With International Data Encryption Algorithm", *Journal Online Jaringan Cot Polıpd (Jojaps)* 8:63-68, 2017

[35] Daniel Castro, Alan Mcquınn, "Unlocking Encryption: Information Security And The Rule Of Law", *Informatıon Technology & Innovatıon Foundatıon* Sayfa:5 http://www2.itif.org/2016-unlocking-encryption.pdf (Erişim Tarihi: 27.03.2018)

[36] Mansoor Ebrahim, Shujaat Khan, Umer Bin Khalid, "Symmetric Algorithm Survey: A Comparative Analysis", International Journal of Computer Applications, 61(20):12-19, 2013

[37] Jeff Tyson, "How Encryption Works" https://mitseu.files.wordpress.com/2014/09/how-encryption-works.pdf (Erişim: Tarihi: 27.03.2018)

[38] Essays, UK. "Principles Of Pretty Good Privacy Information Technology Essay" https://www.ukessays.com/essays/information-technology/principles-of-pretty-good-privacy-information-technology-essay.php#citethis (Erişim: Tarihi: 27.03.2018)

[39] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, Samir Belfkih, "Big Data Technologies: A Survey" *Journal of King Saud University-Computer and Information Sciences* https://doi.org/10.1016/j.jksuci.2017.06.001 12Haziran2017

[40] "Cloud vs. In-house: Which Hadoop Option is Right for You?" https://datafloq.com/read/cloud-vs-house-hadoop-option-right/55 (Erişim: Tarihi: 27.03.2018)

[41] Luuk Vloet, Master tezi, Radboud University, "Influence of Big Data and Analytics On Management Control", Sayfa:18-26, Haziran 2016 http://theses.ubn.ru.nl/bitstream/handle/123456789/1745/1516.017%20luuk%20vloet%20mathesis.pdf?sequence=1 (Erişim: Tarihi: 30.03.2018)

[42] Marcos D.Assunção, Rodrigo N. Calheiros, Silvia Bianchi, Marco A.S. Netto, Rajkumar Buyya, "Big Data Computing And Clouds: Trends And Future Directions", Journal of Parallel and Distributed Computing 79–80: 3-15, 2015

[43] Thomas Davenport, "Big Data At Work Dispelling The Myths, Uncovering The Opportunities" https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/hbr-big-data-at-work-107035.pdf (Erişim Tarihi: 19.04.2018)

[44] A.V. Den Boer, "Dynamic Pricing And Learning: Historical Origins, Current Research, And New Directions" Surveys İn Operations Research And Management Science, 20(1): 1-18, 2015

[45] Juliefrizzo-Barker, Peter A.Chow-White, Maryam Mozafari, Dung Ha, "An Empirical Study Of The Rise Of Big Data in Business Scholarship" International Journal Of Information Management 36(3): 403-413, 2016

[46] Andrew Mcafee, Erik Brynjolfsson, " Big Data: The Management Revolution" Harvard Business Review 90(10):60-6, 68, 128 https://hbr.org/2012/10/big-data-the-management-revolution (Erişim Tarihi: 30.03.2018)

[47] Ray M. Chang, Robert J. Kauffman, Young Ok Kwon, "Understanding The Paradigm Shift To Computational Social Science in The Presence Of Big Data" Published in Decision Support Systems, 63: 67-80, 2014

[48] Kaile Zhou, Chao Fu, Shanlin Yang, "Big Data Driven Smart Energy Management: From Big Data To Big İnsights" Renewable And Sustainable Energy Reviews 56: 215-225, 2016

[49] Nitin Agrawal, Aprna Tripathi, Nitin Kr, Agrawal "Big Data Security And Privacy Issues: A Review" International Journal Of Innovative Computer Science & Engineering 2(4):12-15, 2015

[50] Aisling O'Driscoll, Jurate Daugelaite, Roy D. Sleator, "'Big Data', Hadoop And Cloud Computing In Genomics", Journal of Biomedical Informatics 46(5): 774-781, 2013

[51] "A View Of Cloud Computing", Communications of the ACM, doi:10.1145/1721654.1721672. 53(4):50-58, 2010

[52] Liu Huan, "Bigdata Drives Cloud Adoption In Enterprise", IEEE internet Computing 17(4): 68–71, 2013

[53] Suraj Pandey, Surya Nepal, "Cloud Computing and Scientific Applications - Big Data, Scalable Analytics, and Beyond Preface", Future Generation Computer Systems DOI:10.1016/j.future.2013.04.026 29(7):1774–1776, 2013

[54] Daniel Warneke, Odej Kao, "Nephele: Efficient Parallel Data Processing In The Cloud" doi:10.1145/1646468.1646476 2009 http://stratosphere.eu/assets/papers/Nephele_09.pdf (Erişim Tarihi: 20.06.2018)

[55] Thilina Gunarathne, Bingjing Zhang, Tak-Lon Wu, Judy Qiu, "Scalable Parallel Computing On Clouds Using Twister4Azure Iterative MapReduce", Future Generation Computer Systems 29(4): 1035-1048, 2013

[56] Abdul Nasir Khan, M. L. Mat Kiah, Mazhar Ali, Sajjad A. Madani, Atta ur Rehman Khan, Shahaboddin Shamshirband "BSS: Block-Based Sharing

Scheme For Secure Datastorage Services In Mobile Cloud Environment", *The Journal of Supercomputing* 70(2): 946–976, 2014

[57] Y, Amanatullah, Ipung H.P., Juliandri A, Lim C. "Toward Cloud Computing Reference Architecture: Cloud Service Management Perspective." DOI**:** 10.1109/ICTSS.2013.6588059 Published 2013 in International Conference on ICT for Smart Society

[58] Qi Zhang, Lu Cheng, Raouf Boutaba, "Cloud Computing: State-Of-The-Art And Research Challenges", *Journal of Internet Services and Applications* 1(1): 7–18, 2010

[59] Ohnmar Aung, Thandar Thein, "Enhancing Namenode Fault Tolerance İn Hadoop Distributed File System", *International Journal of Computer Applications* 87(12):41-47, 2014

[60] Can Uzunkaya, Tolga Ensari, Yusuf Kavurucu, "Hadoop Ecosystem And Its Analysis On Tweets", Procedia - Social and Behavioral Sciences 195: 1890-1897 , 2015

[61] Apache Hadoop. https://hadoop.apache.org/ (Erişim Tarihi: 30.03.2018)

[62] Basit Qureshi, Yasir Javed, Anis Kouba, Mohamed-Foued Sriti, Maram Alajlan, "Performance of a Low Cost Hadoop Cluster for Image Analysis in Cloud Robotics Environment", Symposium on Data Mining Applications, 82: 90-98 2016

[63] "What is Hadoop?: Hadoop in Action", https://livebook.manning.com/#!/book/hadoop-in-action/chapter-1/14 (Erişim Tarihi: 30.03.2018)

[64] A.Elsayed, O. Ismail, M.E. El-Sharkawi, "MapReduce: State-Of-The-Art And Research Directions", International Journal Of Computer And Electrical Engineering, 6(1):34-39, 2014

[65] Subramaniyaswamy, V. Vijayakumar, R. Logesha V. Indragandhi, "Unstructured Data Analysis On Big Data Using MapReduce", Procedia Computer Science 50: 456-465, 2015

[66] S. Vijaykumar, S. G. Saravanakumar, M. Balamurugan Dr., "Unique Sense: Smart Computing Prototype", Procedia Computer Science 50: 223-228, 2015

[67] Mohd Rehan Ghazi, Durgaprasad Gangodkar, "Hadoop, MapReduce And Hdfs: A Developers Perspective", Procedia Computer Science 48: 45-50, 2015

[68] Rakesh Kumar, Bhanu Bhushan Parashar, Sakshi Gupta, Yougeshwary Sharma,Neha Gupta, " Apache Hadoop, NoSQL and NewSQL Solutions of Big Data", International Journal of Advance Foundation and Research in Science & Engineering (IJAFRSE) 1(6):28-36, 2014

[69] Best Practices Series, "The Evolution Of Big Data: Nosql, Hadoop, Spark & Beyond", Big Data Quarterly | Bahar 2016 https://www.datastax.com/wp-content/uploads/resources/whitepaper/Big-Data-Quarterly-The-Evolution-of-Big-Data.pdf (Erişim Tarihi:20.04.2018)

[70] Sachin Bende, Rajashree Shedge, "Dealing With Small Files Problem In Hadoop Distributed File System", Procedia Computer Science 79: 1001-1012, 2016

[71] Vaibhav Gopal Korat, Kumar Swamy Pamu, "Reduction Of Data At Namenode İn Hdfs Using Harballing Technique", International Journal of Advanced Research in Computer Engineering & Technology 1(4):635-642, 2012

[72] Priya Deshpande, Darshan Bora, "The Recovery System For Hadoop Cluster" https://ksiresearchorg.ipage.com/seke/dms14paper/paper76.pdf (Erişim Tarihi: 20.04.2018)

[73] Florin Dinu, T. S. Eugene Ng, "Analysis Of Hadoop's Performance Under Failures", *Computer Science Technical Reports* 2011 https://pdfs.semanticscholar.org/e72b/9c83fa5ff3f876b3ccec0484b0d02b569418.pdf (Erişim Tarihi:26.03.2018)

[74] Chuck Lam, "Hadoop in Action", Kitap, Sayfa:181-184, 2011

[75] Tom White, "Hadoop: The Definitive Guide", O'reilly Media, Inc., Sayfa:44, Mayıs 2012.

[76] Sriramoju Ajay Babu, Namavaram Vijay, Ramesh Gadde "An Overview Of Big Data Challenges, Tools And Techniques" International Journal Of Research And Applications Transactions 4(16): 596-601, 2017

[77] Jens Dittrich, Jorge-Arnulfo Quiane-Ruiz, "Efficient Big Data Processing İn Hadoop MapReduce" Proceedings of the VLDB Endowment, 5(12): 2014-2015, 2012

[78] Katarina Grolinger, Michael Hayes, Wilson A. Higashino, Alexandra L'Heureux, David S. Allison, Miriam A.M. Capretz, "Challenges For MapReduce İn Big Data", *Proc. of the IEEE 10th 2014 World Congress on Services*, 182-189, 2014

[79] Manolis Gergatsoulis, Christos Nomikos, Eleftherios Kalogeros, And Matthew Damigos "An Algorithm For Querying Linked Data Using Map-Reduce", International Conference On Data Management İn Cloud, Grid And P2p Systems 8059:51-62, 2013

[80] Subramaniyaswamy V, Vijayakumar V, Logesh R, Indragandhi V, "Unstructured Data Analysis On Big Data Using MapReduce" 2nd International Symposium On Big Data And Cloud Computing (Isbcc'15) Procedia Computer Science 50(2015):456 – 465, 2015

[81] Jacopo Urbani, Jason Maassen, Niels Drost, Frank Seinstra, Henri Bal, "Scalable RDF Data Compression With MapReduce", Concurrency and Computation Practice and Experience 25(1):24-39, 2013

[82] Kim, J. S., Whang, K. Y., Kwon, H. Y., Song, I. Y., "PARADISE: Big Data Analytics Using The Dbms Tightly İntegrated With The Distributed File System", 19(3): 299-322, 2016

[83] B. Saraladevi, N. Pazhaniraja, Victer Paul, M.S. Saleem Basha, Dhavachelvan, "Big Data And Hadoop-A Study İn Security Perspective", Procedia Computer Science 50:596-601, 2015

[84] C.L.Philip Chen, Chun-Yang Zhang, "Data-İntensive Applications, Challenges, Techniques", Information Sciences 275: 314-347, 2014

[85] Ruchi Mittal, Bagga Ruhi. "Performance Analysis Of Multi-Node Hadoop Clusters Using Amazon EC2 Instances." International Journal Of Science And Research (IJSR) 4(10): 1646-1650, 2013

[86] Parth Gohil, Dweepna Garg, Bakul Panchal. "A Performance Analysis Of MapReduce Applications on Big Data in Cloud Based Hadoop." Information Communication And Embedded Systems (ICICES), International Conference On. IEEE, DOI: 10.1109/ICICES.2014.7033791. 2014

[87] Devadiga, Arun S., P. R. Shalini, And Aditya Kumar Sinha. "Virtual Hadoop: The Study And Implementation Of Hadoop in Virtual Environment Using Cloudstack KVM." International Journal Of Engineering Development And Research. 2(2) : 1899-1906, 2014

[88] Pratiksha D. Mandal, Madhuri S. Kadam, Sayali R. Kakade, Monali J. Reddy, Prof. Amar More Rehberliğinde, "Study Of Elastic Hadoop On Private Cloud" International Journal of Scientific and Research Publications, 6(1):321-323, 2016

[89] Weiyi Shang, Zhen Ming Jiang, Hadi Hemmati, Bram Adams, Ahmed E. Hassan, Patrick Martin, "Assisting Developers Of Big Data Analytics Applications When Deploying On Hadoop Clouds" ICSE 2013, San Francisco, CA, USA DOI: 10.1109/ICSE.2013.6606586. 402-411

[90] Atilla Ergüzen, Erdal Erdal, "Medical Image Archiving System Implementation with Lossless Region of Interest and Optical Character Recognition", Journal of Medical Imaging and Health Informatics 7(6): 1246–1252, 2017

[91] Atilla Ergüzen, Erdal Erdal, " An Efficient Middle Layer Platform for Medical Imaging Archives", Journal of Healthcare Engineering, https://www.hindawi.com/journals/jhe/aip/3984061/ (Erişim Tarihi: 9.6.2018)

[92] Atilla Ergüzen, Mahmut Ünver, " Developing a File System Structure to Solve Healthy Big Data Storage and Archiving Problems Using a Distributed File System", Applied Sciences 8(6):913, 2018

EKLER

EK1. Sunucu Kısmında Kodlama

```
namespace server_side
{
      public partial class Service1 : ServiceBase
    {
        private static long[] fileSize = { 0, 4096000 + 1, 10240000 + 4096000 + 2,
79872000 };
        private static int[] blockSize = { 4 * 1024, 10 * 1024, 64 * 1024 };
        private static int[] bitSay = { 0, 1000, 2000, 999, 1999, 3000 };
        private static int blockInf = 16;  //16B block bilgileri
        private static int blockTip;
        private static string inputPath;
        private static string outputPath = @"C:\Users\dell\Documents\Visual Studio
2013\Projects\ServerService_Fileio\ServerService_Fileio\75mb.dat";
        private static string textPath = @"C:\Users\dell\Documents\Visual Studio
2013\Projects\ServerService_Fileio\ServerService_Fileio\TextFile1.txt";
        Stream outputFile = File.Open(outputPath, FileMode.Open,
FileAccess.ReadWrite);
        static int BitSet(int value, int position)
{  return value | (1 << position); }
        static int BitClear(int value, int position)
{  return value & ~(1 << position); }
        private static LinkedList<int> pos = new LinkedList<int>();
        private static List<int> bit = new List<int>();
        private static Object thisLock = new Object();
        private static int startBlock; // = bit[0]

        private static string[] kullanici = { "a", "1" };
        private static string[] ipList = { "192.168.1.x", "192.168.1.y",
"192.168.1.z" };  // server ip liste   0.kendi_ip 1.gönderilen_ip 2.replica_ip
        public static string receivedPath = @"C:\Users\dell\Desktop";
        public static string fileName;
        private static String ip = "192.168.1.t";   //client ip
        public static string jsonPath = @"C:\Users\dell\Documents\Visual Studio
2013\Projects\ServerService_Fileio\ServerService_Fileio\JsonFiles\EncryptData_json5.j
son";
        IPEndPoint ipEnd;
        Socket sock;
        Socket clientSock;

        public Service1()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
            ipEnd = new IPEndPoint(IPAddress.Any, 443);
            sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.IP);
            sock.Bind(ipEnd);
            this.run();
        }
```

```
        private void run()
        {
            lock (thisLock)
            {
                start_server();
                try
                {
                    PgpDecrypt.Decrypt(receivedPath + "/" + fileName,
      desktop+@"\server_side\server_side\Keys\PGPPrivateKey.asc", "P@ll@m@lli",
      receivedPath + "/" + "D" + fileName);
                    WebClient c = new WebClient();
                    var data = c.DownloadString(receivedPath + "/" + "D" + fileName);

                    JObject o = JObject.Parse(data);
                    if (kullanici[0].Equals(o["name"].ToString()) &
      kullanici[1].Equals(o["sifre"].ToString()))
                    {
//client ip ekle jsonlara                          ip = o["clientIP"].ToString();
                        if(o["islem_kodu"].ToString()=="1")
                        {//ip gönder
                            SendIP(ipList[1]);
                        }
                        if(o["islem_kodu"].ToString()=="2")
                        {//dosya al-yaz ve dosyayı replikalara gönder
                            start_server();
                            yaz();
  //                          SendFile(jsonPath,ipList[2]);
                        }
                        if(o["islem_kodu"].ToString()=="3")
                        {//dosya sil
                            //client inf dosya al
                            readJson(); // aldığı dsya oku
                            deleteFile(client_sendstartblock, outputFile,
      client_sendblocktip);
                        }
                        if(o["islem_kodu"].ToString()=="4")
                        {//client'a gönder - read
                            readFile(outputFile, 14336377);
                            SendFile(textPath,ip);
                        }
                        if(o["islem_kodu"].ToString()=="5")
                        {//replika
                            start_server();
                            yaz();
                        }
                    }
                    else
                    {
                        start_server();
                    }
                    run();
                }
                catch (Exception ex) { MessageBox.Show(ex.ToString()); sock.Close();
      }
            }
        }

        private static void deleteFile(int start_bunch, Stream outputFile, int
      block_Tip) // p start bunch
        {
            blockTip = block_Tip;
```

```
            int a = 0;
            int bit_pos;

            //   outputFile.Position = bit_pos / 8;
            do
            {
                bit_pos = bit_pointer(start_bunch);
                a = getNxtBlock(outputFile, start_bunch);

                outputFile.Position = bit_pos / 8;
                int value = outputFile.ReadByte();
                value = BitSet(value, 7 - (bit_pos % 8));
                outputFile.Position = bit_pos / 8;
                outputFile.WriteByte((byte)value);

                //  outputFile.Position = a;
                start_bunch = a;
            } while ((outputFile.Position = a) != (int)0);
        }

        private void start_server()
        {
            try
            {
                sock.Listen(100);
                clientSock = sock.Accept();

                byte[] serverData = new byte[1024 * 5000];
                int receivedBytesLen = clientSock.Receive(serverData);
                int fileNameLen = BitConverter.ToInt32(serverData, 0);
                fileName = Encoding.ASCII.GetString(serverData, 4, fileNameLen);

                BinaryWriter bWrite = new BinaryWriter(File.Open(receivedPath + "/" +
fileName, FileMode.Append)); ;
                bWrite.Write(serverData, 4 + fileNameLen, receivedBytesLen - 4 -
fileNameLen);
                inputPath = receivedPath + "/" + fileName;
                bWrite.Close();
                clientSock.Close();
            }
            catch (Exception ex) { }
        }

        public static void SendFile(string fileName, string aliciIP)
        {
            try
            {
                IPAddress[] ipAddress = Dns.GetHostAddresses(aliciIP);
                IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], 443);
                Socket ServerSock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.IP);

                string filePath = "";
                fileName = fileName.Replace("\\", "/");
                while (fileName.IndexOf("/") > -1)
                {
                    filePath += fileName.Substring(0, fileName.IndexOf("/") + 1);
                    fileName = fileName.Substring(fileName.IndexOf("/") + 1);
                }

                byte[] fileNameByte = Encoding.ASCII.GetBytes(fileName);
```

```
            byte[] fileData = File.ReadAllBytes(filePath + fileName);
            byte[] clientData = new byte[4 + fileNameByte.Length +
fileData.Length];
            byte[] fileNameLen = BitConverter.GetBytes(fileNameByte.Length);

            fileNameLen.CopyTo(clientData, 0);
            fileNameByte.CopyTo(clientData, 4);
            fileData.CopyTo(clientData, 4 + fileNameByte.Length);

            ServerSock.Connect(ipEnd);
            ServerSock.Send(clientData);
            ServerSock.Close();
        }
        catch (Exception ex) { }
    }

    public static void SendIP(string IP)
    {
        try
        {
            IPAddress[] ipAddress = Dns.GetHostAddresses(ip);
            IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], 443);
            Socket ServerSock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.IP);

            byte[] serverData = System.Text.Encoding.ASCII.GetBytes(IP);
            ServerSock.Connect(ipEnd);
            ServerSock.Send(serverData);
            ServerSock.Close();
        }
        catch (Exception ex) { }
    }

    private static void getfileInf(Stream p1, Stream p2)
    {
        BinaryWriter writer = new BinaryWriter(p2);
        LinkedListNode<int> node;
        pos.AddFirst(block_pointer(bit[0]));
        node = pos.Find(block_pointer(bit[0]));

        DateTime dt = DateTime.Today;
        int a = (int)p1.Length;
        int c = blockSay(inputPath);
        int b = a - (c - 1) * blockSize[blockTip];
        int date = dt.Year;
        long nBlock;
        for (int j = 1; j < bit.Count; j++)
        {
            pos.AddAfter(pos.Find(block_pointer(bit[j - 1])),
block_pointer(bit[j]));
            nBlock = pos.Find(block_pointer(bit[j - 1])).Next.Value;
            p2.Position = block_pointer(bit[j - 1]);
            writer.Write((int)nBlock);    //nextblockId yaz
            writer.Write((int)(blockTip - 16));    //fileSize yaz
            writer.Write((int)date);           //tarih yaz
            writer.Write(ip);  // client ip yaz
        }
        p2.Position = pos.Last.Value;
        writer.Write((int)0);
        writer.Write((int)b);
        writer.Write((int)date);
```

```
            writer.Write(ip);
        } //next block adres + dosya boyutu + tarih yaz (16B)

        private static int getNxtBlock(Stream outputFile, long position)
        {
            byte[] b = new byte[4];
            outputFile.Position = position;
            int numBytes = 0;
            outputFile.Read(b, 0, b.Length);
            numBytes = BitConverter.ToInt32(b, 0);
            return numBytes;// next block adres
        }

        private static void readFile( Stream outputFile, long position)
        {
            TextWriter dosya = new StreamWriter(textPath);
            blockSay(inputPath);
            byte[] block = new byte[blockSize[blockTip]];
            outputFile.Position = position;
            int a = 0;
            do
            {
                a = getNxtBlock(outputFile, outputFile.Position);
                while (outputFile.Read(block, 0, block.Length) > 0)
                {
                    for (int i = 16; i < block.Length; i++)
                    {
                        dosya.Write((char)block[i]);
                    }
                    break;
                }
            } while ((outputFile.Position = a) != (int)0);  //next block 0 olana
        kadar oku
        }

        private static void bitListe(Stream p1, Stream p2) //block sayısınca false
        bit listele
        {
            int a = blockSay(inputPath);
            int fileBytes;
            StringBuilder sb = new StringBuilder();

            for (int i = bitSay[blockTip] / 8; i < bitSay[blockTip + 3] / 8; i++)
        //blocksayısı/8
            {
                p2.Position = i;
                fileBytes = p2.ReadByte();
                sb.Append(Convert.ToString(fileBytes, 2).PadLeft(8, '0')); //8
        karakter tamamla soldan 0 yaz
            }
            bit.Clear();
            lock (thisLock)
            {
                for (int i = 0; i < sb.Length; i++) // blocksayısı
                {
                    if (sb[i] == '1')
                    {
                        bit.Add(i + bitSay[blockTip]); a--;
                    }
                    if (a == 0) break;
                }
```

```
                startBit = bit[0];
            }
        }

        private static void write(Stream p1, Stream p2)
        {
            bitListe(p1, p2);
            byte[] block = new byte[blockSize[blockTip] - blockInf];
            for (int i = 0; i < bit.Count; i++)
            {//dosya yaz
                p1.Position = i * (blockSize[blockTip] - blockInf);
                p1.Read(block, 0, block.Length);
                p2.Position = block_pointer(bit[i]) + blockInf;
                p2.Write(block, 0, block.Length);
                //bitmap
                p2.Position = bit[i] / 8;
                int value = p2.ReadByte();
                value = BitClear(value, 7 - (bit[i] % 8));
                p2.Position = bit[i] / 8;
                p2.WriteByte((byte)value);
            }
            getfileInf(p1, p2);
        }  //dosya yaz

        private static int block_pointer(int bit_index)
        {
            int bitPosition = bitSay[5]/8 + blockSize[blockTip] * (bit_index -
bitSay[blockTip]) + (int)fileSize[blockTip];
            return bitPosition; //bytePosition
        } //bit -> block

        private static int bit_pointer(int blockPosition)
        {
            int bit_index = (blockPosition - (bitSay[5] / 8) -
(int)fileSize[blockTip]) / blockSize[blockTip] + bitSay[blockTip];
            return bit_index; //bytePosition
        }

        private static int blockSay(string p1)
        {
            int min_blockSize, b_say;
            int i = 1, j = 0;
            FileInfo fi = new FileInfo(p1);
            long dosyaBoyutu = fi.Length;
            min_blockSize = (int)dosyaBoyutu / blockSize[j];
            for (int x = 1; x < blockSize.Count(); x++)
            {
                b_say = (int)dosyaBoyutu / blockSize[x];
                if (min_blockSize > b_say)
                    if (b_say > 0)
                    { min_blockSize = b_say; j = x; }
            }
            blockTip = j;
            while (dosyaBoyutu > blockSize[j])
            { dosyaBoyutu = dosyaBoyutu - blockSize[j]; i++; }
            return i;
        }//input dosyası kaç block

        private void yaz()
        {
```

```
            Stream inputFile = File.Open(inputPath, FileMode.Open,
       FileAccess.ReadWrite);
            write(inputFile, outputFile);

            create_json(ip,block_pointer(startBit),blockTip);
 //           SendFile(desktop +
       @"\server_side\server_side\JsonFiles\client_inf.json",ip);
           // inputFile.Close();
          //   outputFile.Close();
        }

        private void create_json(string client_ip, int start_block, int block_tip)
        {
            JObject clientInf = new JObject(
              new JProperty("client_ip", ip),
    new JProperty("start_block", start_block),
    new JProperty("block_tip", block_tip));

            using (StreamWriter file = File.AppendText(desktop +
       @"\server_side\server_side\JsonFiles\client_inf.json"))
            using (JsonTextWriter writer = new JsonTextWriter(file))
           {
                clientInf.WriteTo(writer);
           }
        } //dosya bilgilerini yaz

        private int client_sendstartblock;
        private int client_sendblocktip;
        private void readJson()
        {
            WebClient c1 = new WebClient();
            var data1 = c1.DownloadString(desktop +
       @"\server_side\server_side\JsonFiles\client_inf.json");

            JObject o1 = JObject.Parse(data1);
            client_sendstartblock = Convert.ToInt32(o1["start_block"]);
            client_sendblocktip = Convert.ToInt32(o1["block_tip"]);
        }
    }
}
```

```
namespace server_side
{
    class PgpDecrypt
    {
        private const int BufferSize = 0x10000; // should always be power of 2

        #region Encrypt

        /*
         * Encrypt the file.
         */

        public static void EncryptFile(string inputFile, string outputFile, string
publicKeyFile, bool armor, bool withIntegrityCheck)
        {
            try
            {
                using (Stream publicKeyStream = File.OpenRead(publicKeyFile))
                {
                    PgpPublicKey encKey = ReadPublicKey(publicKeyStream);

                    using (MemoryStream bOut = new MemoryStream())
                    {
                        PgpCompressedDataGenerator comData = new
PgpCompressedDataGenerator(CompressionAlgorithmTag.Zip);
                        PgpUtilities.WriteFileToLiteralData(comData.Open(bOut),
PgpLiteralData.Binary, new FileInfo(inputFile));

                        comData.Close();
                        PgpEncryptedDataGenerator cPk = new
PgpEncryptedDataGenerator(SymmetricKeyAlgorithmTag.Cast5, withIntegrityCheck, new
SecureRandom());

                        cPk.AddMethod(encKey);
                        byte[] bytes = bOut.ToArray();

                        using (Stream outputStream = File.Create(outputFile))
                        {
                            if (armor)
                            {
                                using (ArmoredOutputStream armoredStream = new
ArmoredOutputStream(outputStream))
                                {
                                    using (Stream cOut = cPk.Open(armoredStream,
bytes.Length))
                                    {
                                        cOut.Write(bytes, 0, bytes.Length);
                                    }
                                }
                            }
                            else
                            {
                                using (Stream cOut = cPk.Open(outputStream,
bytes.Length))
                                {
                                    cOut.Write(bytes, 0, bytes.Length);
                                }
                            }
                        }
                    }
                }
```

```
            }
            catch (PgpException e)
            {
                throw;
            }
        }

        #endregion Encrypt

        #region Encrypt and Sign

        /*
         * Encrypt and sign the file pointed to by unencryptedFileInfo and
         */

        public static void EncryptAndSign(string inputFile, string outputFile,
string publicKeyFile, string privateKeyFile, string passPhrase, bool armor)
        {
            PgpEncryptionKeys encryptionKeys = new PgpEncryptionKeys(publicKeyFile,
privateKeyFile, passPhrase);

            if (!File.Exists(inputFile))
                throw new FileNotFoundException(String.Format("Input file [{0}] does
not exist.", inputFile));

            if (!File.Exists(publicKeyFile))
                throw new FileNotFoundException(String.Format("Public Key file [{0}]
does not exist.", publicKeyFile));

            if (!File.Exists(privateKeyFile))
                throw new FileNotFoundException(String.Format("Private Key file
[{0}] does not exist.", privateKeyFile));

            if (String.IsNullOrEmpty(passPhrase))
                throw new ArgumentNullException("Invalid Pass Phrase.");

            if (encryptionKeys == null)
                throw new ArgumentNullException("Encryption Key not found.");

            using (Stream outputStream = File.Create(outputFile))
            {
                if (armor)
                    using (ArmoredOutputStream armoredOutputStream = new
ArmoredOutputStream(outputStream))
                    {
                        OutputEncrypted(inputFile, armoredOutputStream,
encryptionKeys);
                    }
                else
                    OutputEncrypted(inputFile, outputStream, encryptionKeys);
            }
        }

        private static void OutputEncrypted(string inputFile, Stream outputStream,
PgpEncryptionKeys encryptionKeys)
        {
            using (Stream encryptedOut = ChainEncryptedOut(outputStream,
encryptionKeys))
            {
                FileInfo unencryptedFileInfo = new FileInfo(inputFile);
                using (Stream compressedOut = ChainCompressedOut(encryptedOut))
```

```
                {
                    PgpSignatureGenerator signatureGenerator =
InitSignatureGenerator(compressedOut, encryptionKeys);
                    using (Stream literalOut = ChainLiteralOut(compressedOut,
unencryptedFileInfo))
                    {
                        using (FileStream inputFileStream =
unencryptedFileInfo.OpenRead())
                        {
                            WriteOutputAndSign(compressedOut, literalOut,
inputFileStream, signatureGenerator);
                            inputFileStream.Close();
                        }
                    }
                }
            }
        }

        private static void WriteOutputAndSign(Stream compressedOut, Stream
literalOut, FileStream inputFile, PgpSignatureGenerator signatureGenerator)
        {
            int length = 0;
            byte[] buf = new byte[BufferSize];
            while ((length = inputFile.Read(buf, 0, buf.Length)) > 0)
            {
                literalOut.Write(buf, 0, length);
                signatureGenerator.Update(buf, 0, length);
            }
            signatureGenerator.Generate().Encode(compressedOut);
        }

        private static Stream ChainEncryptedOut(Stream outputStream,
PgpEncryptionKeys m_encryptionKeys)
        {
            PgpEncryptedDataGenerator encryptedDataGenerator;
            encryptedDataGenerator = new
PgpEncryptedDataGenerator(SymmetricKeyAlgorithmTag.TripleDes, new SecureRandom());
            encryptedDataGenerator.AddMethod(m_encryptionKeys.PublicKey);
            return encryptedDataGenerator.Open(outputStream, new byte[BufferSize]);
        }

        private static Stream ChainCompressedOut(Stream encryptedOut)
        {
            PgpCompressedDataGenerator compressedDataGenerator = new
PgpCompressedDataGenerator(CompressionAlgorithmTag.Zip);
            return compressedDataGenerator.Open(encryptedOut);
        }

        private static Stream ChainLiteralOut(Stream compressedOut, FileInfo file)
        {
            PgpLiteralDataGenerator pgpLiteralDataGenerator = new
PgpLiteralDataGenerator();
            return pgpLiteralDataGenerator.Open(compressedOut,
PgpLiteralData.Binary, file);
        }

        private static PgpSignatureGenerator InitSignatureGenerator(Stream
compressedOut, PgpEncryptionKeys m_encryptionKeys)
        {
            const bool IsCritical = false;
            const bool IsNested = false;
```

```
            PublicKeyAlgorithmTag tag =
m_encryptionKeys.SecretKey.PublicKey.Algorithm;
            PgpSignatureGenerator pgpSignatureGenerator = new
PgpSignatureGenerator(tag, HashAlgorithmTag.Sha1);
            pgpSignatureGenerator.InitSign(PgpSignature.BinaryDocument,
m_encryptionKeys.PrivateKey);
            foreach (string userId in
m_encryptionKeys.SecretKey.PublicKey.GetUserIds())
            {
                PgpSignatureSubpacketGenerator subPacketGenerator = new
PgpSignatureSubpacketGenerator();
                subPacketGenerator.SetSignerUserId(IsCritical, userId);

pgpSignatureGenerator.SetHashedSubpackets(subPacketGenerator.Generate());
                // Just the first one!
                break;
            }

pgpSignatureGenerator.GenerateOnePassVersion(IsNested).Encode(compressedOut);
            return pgpSignatureGenerator;
        }

        #endregion Encrypt and Sign

        #region Decrypt

        /*
       * decrypt a given stream.
       */

        public static void Decrypt(string inputfile, string privateKeyFile, string
passPhrase, string outputFile)
        {
            if (!File.Exists(inputfile))
                throw new FileNotFoundException(String.Format("Encrypted File [{0}]
not found.", inputfile));

            if (!File.Exists(privateKeyFile))
                throw new FileNotFoundException(String.Format("Private Key File
[{0}] not found.", privateKeyFile));

            if (String.IsNullOrEmpty(outputFile))
                throw new ArgumentNullException("Invalid Output file path.");

            using (Stream inputStream = File.OpenRead(inputfile))
            {
                using (Stream keyIn = File.OpenRead(privateKeyFile))
                {
                    Decrypt(inputStream, keyIn, passPhrase, outputFile);
                }
            }
        }

        /*
        * decrypt a given stream.
        */

        public static void Decrypt(Stream inputStream, Stream privateKeyStream,
string passPhrase, string outputFile)
        {
            try
```

```
            {
                PgpObjectFactory pgpF = null;
                PgpEncryptedDataList enc = null;
                PgpObject o = null;
                PgpPrivateKey sKey = null;
                PgpPublicKeyEncryptedData pbe = null;
                PgpSecretKeyRingBundle pgpSec = null;

                pgpF = new
PgpObjectFactory(PgpUtilities.GetDecoderStream(inputStream));
                // find secret key
                pgpSec = new
PgpSecretKeyRingBundle(PgpUtilities.GetDecoderStream(privateKeyStream));

                if (pgpF != null)
                    o = pgpF.NextPgpObject();

                // the first object might be a PGP marker packet.
                if (o is PgpEncryptedDataList)
                    enc = (PgpEncryptedDataList)o;
                else
                    enc = (PgpEncryptedDataList)pgpF.NextPgpObject();

                // decrypt
                foreach (PgpPublicKeyEncryptedData pked in
enc.GetEncryptedDataObjects())
                {
                    sKey = FindSecretKey(pgpSec, pked.KeyId,
passPhrase.ToCharArray());

                    if (sKey != null)
                    {
                        pbe = pked;
                        break;
                    }
                }

                if (sKey == null)
                    throw new ArgumentException("Secret key for message not
found.");

                PgpObjectFactory plainFact = null;

                using (Stream clear = pbe.GetDataStream(sKey))
                {
                    plainFact = new PgpObjectFactory(clear);
                }

                PgpObject message = plainFact.NextPgpObject();

                if (message is PgpCompressedData)
                {
                    PgpCompressedData cData = (PgpCompressedData)message;
                    PgpObjectFactory of = null;

                    using (Stream compDataIn = cData.GetDataStream())
                    {
                        of = new PgpObjectFactory(compDataIn);
                    }

                    message = of.NextPgpObject();
```

```
                    if (message is PgpOnePassSignatureList)
                    {
                        message = of.NextPgpObject();
                        PgpLiteralData Ld = null;
                        Ld = (PgpLiteralData)message;
                        using (Stream output = File.Create(outputFile))
                        {
                            Stream unc = Ld.GetInputStream();
                            Streams.PipeAll(unc, output);
                        }
                    }
                    else
                    {
                        PgpLiteralData Ld = null;
                        Ld = (PgpLiteralData)message;
                        using (Stream output = File.Create(outputFile))
                        {
                            Stream unc = Ld.GetInputStream();
                            Streams.PipeAll(unc, output);
                        }
                    }
                }
                else if (message is PgpLiteralData)
                {
                    PgpLiteralData ld = (PgpLiteralData)message;
                    string outFileName = ld.FileName;

                    using (Stream fOut = File.Create(outputFile))
                    {
                        Stream unc = ld.GetInputStream();
                        Streams.PipeAll(unc, fOut);
                    }
                }
                else if (message is PgpOnePassSignatureList)
                    throw new PgpException("Encrypted message contains a signed
message - not literal data.");
                else
                    throw new PgpException("Message is not a simple encrypted file -
type unknown.");

                #region commented code

                //if (pbe.IsIntegrityProtected())
                //{
                //    if (!pbe.Verify())
                //        msg = "message failed integrity check.";
                //    //Console.Error.WriteLine("message failed integrity check");
                //    else
                //        msg = "message integrity check passed.";
                //    //Console.Error.WriteLine("message integrity check passed");
                //}
                //else
                //{
                //    msg = "no message integrity check.";
                //    //Console.Error.WriteLine("no message integrity check");
                //}

                #endregion commented code
            }
            catch (PgpException ex)
            {
```

```
                throw;
            }
        }

        #endregion Decrypt

        #region Private helpers

        /*
        * A simple routine that opens a key ring file and loads the first available
key suitable for encryption.
        */

        private static PgpPublicKey ReadPublicKey(Stream inputStream)
        {
            inputStream = PgpUtilities.GetDecoderStream(inputStream);

            PgpPublicKeyRingBundle pgpPub = new PgpPublicKeyRingBundle(inputStream);

            // we just loop through the collection till we find a key suitable for
encryption, in the real
            // world you would probably want to be a bit smarter about this.
            // iterate through the key rings.
            foreach (PgpPublicKeyRing kRing in pgpPub.GetKeyRings())
            {
                foreach (PgpPublicKey k in kRing.GetPublicKeys())
                {
                    if (k.IsEncryptionKey)
                        return k;
                }
            }

            throw new ArgumentException("Can't find encryption key in key ring.");
        }

        /*
        * Search a secret key ring collection for a secret key corresponding to
keyId if it exists.
        */

        private static PgpPrivateKey FindSecretKey(PgpSecretKeyRingBundle pgpSec,
long keyId, char[] pass)
        {
            PgpSecretKey pgpSecKey = pgpSec.GetSecretKey(keyId);

            if (pgpSecKey == null)
                return null;

            return pgpSecKey.ExtractPrivateKey(pass);
        }

        #endregion Private helpers
    }
}
```

```
namespace server_side
{
 public class PgpEncryptionKeys
    {
        public PgpPublicKey PublicKey { get; private set; }

        public PgpPrivateKey PrivateKey { get; private set; }

        public PgpSecretKey SecretKey { get; private set; }

        public PgpEncryptionKeys(string publicKeyPath, string privateKeyPath, string
passPhrase)
        {

            if (!File.Exists(publicKeyPath))

                throw new ArgumentException("Public key file not found",
"publicKeyPath");

            if (!File.Exists(privateKeyPath))

                throw new ArgumentException("Private key file not found",
"privateKeyPath");

            if (String.IsNullOrEmpty(passPhrase))

                throw new ArgumentException("passPhrase is null or empty.",
"passPhrase");

            PublicKey = ReadPublicKey(publicKeyPath);

            SecretKey = ReadSecretKey(privateKeyPath);

            PrivateKey = ReadPrivateKey(passPhrase);

        }

        #region Secret Key

        private PgpSecretKey ReadSecretKey(string privateKeyPath)
        {

            using (Stream keyIn = File.OpenRead(privateKeyPath))

            using (Stream inputStream = PgpUtilities.GetDecoderStream(keyIn))
            {

                PgpSecretKeyRingBundle secretKeyRingBundle = new
PgpSecretKeyRingBundle(inputStream);

                PgpSecretKey foundKey = GetFirstSecretKey(secretKeyRingBundle);

                if (foundKey != null)

                    return foundKey;

            }

            throw new ArgumentException("Can't find signing key in key ring.");

        }
```

```
        private PgpSecretKey GetFirstSecretKey(PgpSecretKeyRingBundle
secretKeyRingBundle)
        {

            foreach (PgpSecretKeyRing kRing in secretKeyRingBundle.GetKeyRings())
            {

                PgpSecretKey key = kRing.GetSecretKeys()

                    .Cast<PgpSecretKey>()

                    .Where(k => k.IsSigningKey)

                    .FirstOrDefault();

                if (key != null)

                    return key;

            }

            return null;

        }

        #endregion

        #region Public Key

        private PgpPublicKey ReadPublicKey(string publicKeyPath)
        {

            using (Stream keyIn = File.OpenRead(publicKeyPath))

            using (Stream inputStream = PgpUtilities.GetDecoderStream(keyIn))
            {

                PgpPublicKeyRingBundle publicKeyRingBundle = new
PgpPublicKeyRingBundle(inputStream);

                PgpPublicKey foundKey = GetFirstPublicKey(publicKeyRingBundle);

                if (foundKey != null)

                    return foundKey;

            }

            throw new ArgumentException("No encryption key found in public key
ring.");

        }

        private PgpPublicKey GetFirstPublicKey(PgpPublicKeyRingBundle
publicKeyRingBundle)
        {

            foreach (PgpPublicKeyRing kRing in publicKeyRingBundle.GetKeyRings())
            {
```

```
                PgpPublicKey key = kRing.GetPublicKeys()
                    .Cast<PgpPublicKey>()
                    .Where(k => k.IsEncryptionKey)
                    .FirstOrDefault();
                if (key != null)
                    return key;
            }
            return null;
        }
        #endregion
        #region Private Key
        private PgpPrivateKey ReadPrivateKey(string passPhrase)
        {
            PgpPrivateKey privateKey =
SecretKey.ExtractPrivateKey(passPhrase.ToCharArray());
            if (privateKey != null)
                return privateKey;
            throw new ArgumentException("No private key found in secret key.");
        }
        #endregion
    }
}
```

EK2. İstemci Kısmında Kodlama

```
namespace ftp_client
{
    public partial class Form1 : Form
    {
        public static string fileName;
        private static String ip;
        private static String path;
        private static IPEndPoint ipEnd;
        private static Socket sock;
        private static Socket clientSock;
        private static string curMsg = "Idle";
        private static string
       desktop=Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
        private static string[] jsonFiles = {
       desktop+@"\ftp_client\JsonFiles\json1.json",

       desktop+@"\ftp_client\JsonFiles\json2.json",

       desktop+@"\ftp_client\JsonFiles\json3.json",

       desktop+@"\ftp_client\JsonFiles\json4.json",

       desktop+@"\ftp_client\JsonFiles\json5.json"};

        public Form1()
        {
            InitializeComponent();

        }

        private static void SendFile(string fileName)
        {
            try
            {
                curMsg = "";
                IPAddress ipa = IPAddress.Parse(ip);
                IPEndPoint endPoint = new IPEndPoint(ipa, 443);

                //  IPAddress[] ipAddress = Dns.GetHostAddresses("192.168.1.119");
                //   IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], 5656);
                Socket clientSock = new Socket(AddressFamily.InterNetwork,
       SocketType.Stream, ProtocolType.IP);

                string filePath = "";

                fileName = fileName.Replace("\\", "/");
                while (fileName.IndexOf("/") > -1)
                {
                    filePath += fileName.Substring(0, fileName.IndexOf("/") + 1);
                    fileName = fileName.Substring(fileName.IndexOf("/") + 1);
                }

                byte[] fileNameByte = Encoding.ASCII.GetBytes(fileName);
                if (fileNameByte.Length > 850 * 1024)
                {
                    curMsg = "File size is more than 850kb, please try with small
       file.";
                    return;  }
```

```
            curMsg = "Buffering ...";
            byte[] fileData = File.ReadAllBytes(filePath + fileName);
            byte[] clientData = new byte[4 + fileNameByte.Length +
fileData.Length];
            byte[] fileNameLen = BitConverter.GetBytes(fileNameByte.Length);

            fileNameLen.CopyTo(clientData, 0);
            fileNameByte.CopyTo(clientData, 4);
            fileData.CopyTo(clientData, 4 + fileNameByte.Length);

            curMsg = "Connection to server ...";

            clientSock.Connect(endPoint);

            curMsg = "File sending...";
            clientSock.Send(clientData);

            //    curMsg = "Disconnecting...";
            clientSock.Close();
            curMsg = "File transferred.";

        }
        catch (Exception ex)
        {
            if (ex.Message == "No connection could be made because the target
machine actively refused it")
                curMsg = "File Sending fail. Because server not running.";
            else
                curMsg = "File Sending fail." + ex.Message;
        }
    }

    private static string ReceiveIP()
    {
        string stringData = "";
        ipEnd = new IPEndPoint(IPAddress.Any, 443);
        sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.IP);
        sock.Bind(ipEnd);
        try
        {
            sock.Listen(100);
            clientSock = sock.Accept();

            byte[] serverData = new byte[1024];
            int receivedBytesLen = clientSock.Receive(serverData);
            stringData = Encoding.ASCII.GetString(serverData, 0,
receivedBytesLen);

            clientSock.Close();
        }
        catch (Exception ex) { }
        return stringData;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        FileDialog fDg = new OpenFileDialog();
        if (fDg.ShowDialog() == DialogResult.OK)
        {
```

```
            SendFile(desktop + @"\ftp_client\JsonFiles\EncryptData_json2.json");
            SendFile(fDg.FileName);
        }
        ReceiveFile();
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        ip = textBox1.Text;
        label1.Text = curMsg;
    }

    private void button3_Click(object sender, EventArgs e) //ip iste
    {
        string fName = jsonFiles[0];
        if (fName != null)
        {
            encryptJsonFile(fName);
            SendFile(path);
        }
        label2.Text = ReceiveIP();
    }
    private void encryptJsonFile(string fName)
    {
        FileInfo fInfo = new FileInfo(fName);
        // Pass the file name without the path.
        string name = fInfo.FullName;
        // Encrypt the file using the public key from the certificate.

        PgpEncryptionKeys encryptionKeys = new PgpEncryptionKeys(desktop +
@"\ftp_client\Keys\PGPPublicKey.asc", desktop +
@"\ftp_client\Keys\PGPPrivateKey.asc", "P@ll@m@lli");
        PgpEncrypt encrypter = new PgpEncrypt(encryptionKeys);
        using (Stream outputStream = File.Create("C:\\Keys\\EncryptData_" +
fInfo.Name))
        {
            encrypter.EncryptAndSign(outputStream, new FileInfo(name));
        }
        path = @"C:\\Keys\\EncryptData_" + fInfo.Name;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string fName = jsonFiles[2];
        if (fName != null)
        {
            encryptJsonFile(fName);
            SendFile(path);
        }

        //hangi dosya silinecek id gönder
      //
SendFile(@"C:\Users\dell\Desktop\ftp_client\JsonFiles\client_inf.json");
    }

    private void button4_Click(object sender, EventArgs e)
    {
        string fName = jsonFiles[3];
        if (fName != null)
        {
            encryptJsonFile(fName);
```

```
                SendFile(path);
            }
           // ReceiveFile();   // gönderilen dosyanın bilgilerini al
        }
        string receivedPath = desktop +@"\ftp_client";
        private void ReceiveFile()
        {
            ipEnd = new IPEndPoint(IPAddress.Any, 443);
            sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
      ProtocolType.IP);
            sock.Bind(ipEnd);
            sock.Listen(100);
            clientSock = sock.Accept();
            byte[] cData = new byte[1024 * 5000];
            int receivedBytesLen = clientSock.Receive(cData);
            int fileNameLen = BitConverter.ToInt32(cData, 0);
            fileName = Encoding.ASCII.GetString(cData, 4, fileNameLen);

            BinaryWriter bWrite = new BinaryWriter(File.Open(receivedPath + "/" +
      fileName, FileMode.Append)); ;
            bWrite.Write(cData, 4 + fileNameLen, receivedBytesLen - 4 - fileNameLen);

        }
    }
}
```

```
namespace PGPSnippet.PGPEncryption
{
    public class PgpEncrypt
    {

        private PgpEncryptionKeys m_encryptionKeys;

        private const int BufferSize = 0x10000; // should always be power of 2

        /// <summary>
        /// Instantiate a new PgpEncrypt class with initialized PgpEncryptionKeys.
        /// </summary>
        /// <param name="encryptionKeys"></param>
        /// <exception cref="ArgumentNullException">encryptionKeys is
null</exception>

        public PgpEncrypt(PgpEncryptionKeys encryptionKeys)
        {

            if (encryptionKeys == null)

                throw new ArgumentNullException("encryptionKeys", "encryptionKeys is
null.");

            m_encryptionKeys = encryptionKeys;

        }

        /// <summary>

        /// Encrypt and sign the file pointed to by unencryptedFileInfo and

        /// write the encrypted content to outputStream.

        /// </summary>

        /// <param name="outputStream">The stream that will contain the

        /// encrypted data when this method returns.</param>

        /// <param name="fileName">FileInfo of the file to encrypt</param>

        public void EncryptAndSign(Stream outputStream, FileInfo
unencryptedFileInfo)
        {

            if (outputStream == null)

                throw new ArgumentNullException("outputStream", "outputStream is
null.");

            if (unencryptedFileInfo == null)

                throw new ArgumentNullException("unencryptedFileInfo",
"unencryptedFileInfo is null.");

            if (!File.Exists(unencryptedFileInfo.FullName))

                throw new ArgumentException("File to encrypt not found.");

            using (Stream encryptedOut = ChainEncryptedOut(outputStream))
```

```
            using (Stream compressedOut = ChainCompressedOut(encryptedOut))
            {

                PgpSignatureGenerator signatureGenerator =
InitSignatureGenerator(compressedOut);

                using (Stream literalOut = ChainLiteralOut(compressedOut,
unencryptedFileInfo))

                using (FileStream inputFile = unencryptedFileInfo.OpenRead())
                {

                    WriteOutputAndSign(compressedOut, literalOut, inputFile,
signatureGenerator);

                }
            }
        }

        private static void WriteOutputAndSign(Stream compressedOut,

            Stream literalOut,

            FileStream inputFile,

            PgpSignatureGenerator signatureGenerator)
        {

            int length = 0;

            byte[] buf = new byte[BufferSize];

            while ((length = inputFile.Read(buf, 0, buf.Length)) > 0)
            {

                literalOut.Write(buf, 0, length);

                signatureGenerator.Update(buf, 0, length);
            }

            signatureGenerator.Generate().Encode(compressedOut);
        }

        private Stream ChainEncryptedOut(Stream outputStream)
        {

            PgpEncryptedDataGenerator encryptedDataGenerator;

            encryptedDataGenerator =

                new PgpEncryptedDataGenerator(SymmetricKeyAlgorithmTag.TripleDes,
                                              new SecureRandom());

            encryptedDataGenerator.AddMethod(m_encryptionKeys.PublicKey);

            return encryptedDataGenerator.Open(outputStream, new byte[BufferSize]);

        }

        private static Stream ChainCompressedOut(Stream encryptedOut)
```

```
        {

            PgpCompressedDataGenerator compressedDataGenerator =

                new PgpCompressedDataGenerator(CompressionAlgorithmTag.Zip);

            return compressedDataGenerator.Open(encryptedOut);

        }

        private static Stream ChainLiteralOut(Stream compressedOut, FileInfo file)
        {

            PgpLiteralDataGenerator pgpLiteralDataGenerator = new
PgpLiteralDataGenerator();

            return pgpLiteralDataGenerator.Open(compressedOut,
PgpLiteralData.Binary, file);

        }

        private PgpSignatureGenerator InitSignatureGenerator(Stream compressedOut)
        {

            const bool IsCritical = false;

            const bool IsNested = false;

            PublicKeyAlgorithmTag tag =
m_encryptionKeys.SecretKey.PublicKey.Algorithm;

            PgpSignatureGenerator pgpSignatureGenerator =

                new PgpSignatureGenerator(tag, HashAlgorithmTag.Sha1);

            pgpSignatureGenerator.InitSign(PgpSignature.BinaryDocument,
m_encryptionKeys.PrivateKey);

            foreach (string userId in
m_encryptionKeys.SecretKey.PublicKey.GetUserIds())
            {

                PgpSignatureSubpacketGenerator subPacketGenerator =
                   new PgpSignatureSubpacketGenerator();

                subPacketGenerator.SetSignerUserId(IsCritical, userId);

pgpSignatureGenerator.SetHashedSubpackets(subPacketGenerator.Generate());

                // Just the first one!
                break;
            }

pgpSignatureGenerator.GenerateOnePassVersion(IsNested).Encode(compressedOut);

            return pgpSignatureGenerator;

        }
    }
}
```

```
namespace PGPSnippet.Keys
{
    public class PgpEncryptionKeys
    {

        public PgpPublicKey PublicKey { get; private set; }

        public PgpPrivateKey PrivateKey { get; private set; }

        public PgpSecretKey SecretKey { get; private set; }

        public PgpEncryptionKeys(string publicKeyPath, string privateKeyPath, string
passPhrase)
        {

            if (!File.Exists(publicKeyPath))

                throw new ArgumentException("Public key file not found",
"publicKeyPath");

            if (!File.Exists(privateKeyPath))

                throw new ArgumentException("Private key file not found",
"privateKeyPath");

            if (String.IsNullOrEmpty(passPhrase))

                throw new ArgumentException("passPhrase is null or empty.",
"passPhrase");

            PublicKey = ReadPublicKey(publicKeyPath);

            SecretKey = ReadSecretKey(privateKeyPath);

            PrivateKey = ReadPrivateKey(passPhrase);

        }

        #region Secret Key

        private PgpSecretKey ReadSecretKey(string privateKeyPath)
        {

            using (Stream keyIn = File.OpenRead(privateKeyPath))

            using (Stream inputStream = PgpUtilities.GetDecoderStream(keyIn))
            {

                PgpSecretKeyRingBundle secretKeyRingBundle = new
PgpSecretKeyRingBundle(inputStream);

                PgpSecretKey foundKey = GetFirstSecretKey(secretKeyRingBundle);

                if (foundKey != null)

                    return foundKey;

            }

            throw new ArgumentException("Can't find signing key in key ring.");
```

```
        }


        private PgpSecretKey GetFirstSecretKey(PgpSecretKeyRingBundle
secretKeyRingBundle)
        {

            foreach (PgpSecretKeyRing kRing in secretKeyRingBundle.GetKeyRings())
            {

                PgpSecretKey key = kRing.GetSecretKeys()

                    .Cast<PgpSecretKey>()

                    .Where(k => k.IsSigningKey)

                    .FirstOrDefault();

                if (key != null)

                    return key;

            }

            return null;

        }

        #endregion

        #region Public Key

        private PgpPublicKey ReadPublicKey(string publicKeyPath)
        {

            using (Stream keyIn = File.OpenRead(publicKeyPath))

            using (Stream inputStream = PgpUtilities.GetDecoderStream(keyIn))
            {

                PgpPublicKeyRingBundle publicKeyRingBundle = new
PgpPublicKeyRingBundle(inputStream);

                PgpPublicKey foundKey = GetFirstPublicKey(publicKeyRingBundle);

                if (foundKey != null)

                    return foundKey;

            }

            throw new ArgumentException("No encryption key found in public key
ring.");

        }

        private PgpPublicKey GetFirstPublicKey(PgpPublicKeyRingBundle
publicKeyRingBundle)
        {

            foreach (PgpPublicKeyRing kRing in publicKeyRingBundle.GetKeyRings())
```

```
            {
                PgpPublicKey key = kRing.GetPublicKeys()
                    .Cast<PgpPublicKey>()
                    .Where(k => k.IsEncryptionKey)
                    .FirstOrDefault();
                if (key != null)
                    return key;
            }
            return null;
        }
        #endregion
        #region Private Key
        private PgpPrivateKey ReadPrivateKey(string passPhrase)
        {
            PgpPrivateKey privateKey =
SecretKey.ExtractPrivateKey(passPhrase.ToCharArray());
            if (privateKey != null)
                return privateKey;
            throw new ArgumentException("No private key found in secret key.");
        }
        #endregion
    }
}
```