DAĞITIK GÖRÜNTÜ İŞLEME

Murat TEZGİDER

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Galip AYDIN

OCAK - 2017

T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DAĞITIK GÖRÜNTÜ İŞLEME

YÜKSEK LİSANS TEZİ

Murat TEZGİDER

(121129116)

Tezin Enstitüye Verildiği Tarih: 20 ARALIK 2016

Tezin Savunulduğu Tarih : 04 OCAK 2017

Tez Danışmanı : Doç. Dr. Galip AYDIN (F.Ü.)

Diğer Jüri Üyeleri : Prof. Dr. Ali KARCI (İ.Ü.)

Yrd. Doç. Dr. Ahmet ÇINAR (F.Ü.)

OCAK - 2017

ÖNSÖZ

Teknolojinin gelişmesi ile fotoğraf, video gibi çoklu medya dosyalarının miktarı, boyutu ve kalitesi giderek artmaktadır. Bunun sonucu olarak da elde edilen bu verilerin depolanabilmesi ve işlenebilmesi için gerekli çözümlerin geliştirilmesi önemli sorunlar olarak ortaya çıkmaktadır. Görüntü ve video dosyalarının boyutunun, sayısının ve kalitesinin artması geleneksel yöntemler ile bu verilerin depolanması ve işlenmesini zorlaştırmıştır. Büyük bir hızla ve farklı formatlarda üretilen yüksek hacimli bu verilerin depolanması ve işlenmesi için büyük veri teknolojilerinden yararlanılabilmektedir.

MapReduce teknolojisi dağıtık dosya sistemi üzerinde çalışmaktadır. Dağıtık dosya sistemi çok sayıda görüntünün veya büyük boyutlardaki çoklu medya verilerinin depolanması problemine çözüm sağlayabilmektedir. Bu çalışmada, yüksek maliyetli büyük depolama donanımlarına ve yüksek işlem gücüne sahip bilgisayarı kullanmak yerine, pahalı olmayan sunucuları veya bilgisayarları kullanarak oluşturulabilen bir küme (Cluster) üzerinde görüntü işleme için dağıtık bir sistem kurulup çalıştırılması ile ilgili çalışmalar gerçekleştirilmiştir. Bu dağıtık sistem üzerinde görüntü işleme işlemleri gerçekleştirilerek performansı analizi yapılmıştır.

Bu tez çalışmasında büyük hacimli görüntü ve video dosyalarının depolanması için Hadoop Dağıtık Dosya Sistemi (HDFS) kullanılmış ve büyük veri içinde önemli bir yeri olan görüntü ve video dosyalarının paralel olarak işlenmesi için dağıtık bir sistem geliştirilmiştir. Bu sistemde görüntü işleme kütüphanesi olarak OpenIMAJ kütüphanesi kullanılmıştır. HDFS ortamında depolanan görüntü ve video verilerinin MapReduce tekniği kullanılarak dağıtık olarak işlenmesi için gerekli olan programlar yazılmış ve yatay ölçekleme yapılarak performans değerlendirmesi gerçekleştirilmiştir.

TEŞEKKÜR

Bu tez çalışmasında engin bilgi ve tecrübeleriyle desteğini esirgemeyen değerli tez danışmanım **Doç. Dr. Galip AYDIN**'a ve hayatım boyunca her konuda, hep yanımda ve destek olan değerli aileme, tez yazma sürecinde motivasyon desteği sağlayan değerli arkadaşlarıma çok teşekkür ederim.

TÜBİTAK BİLGEM Bilişim Teknolojileri Enstitüsü (BTE) tarafından Kalkınma Bakanlığı Yatırım Programı Desteği ile yürütülen Bulut Bilişim ve Büyük Veri Araştırma Laboratuvarı (B3LAB)'na tez çalışmam boyunca sağlamış olduğu altyapı desteği için teşekkür ederim.

Murat TEZGİDER

ELAZIĞ-2017

İÇİNDEKİLER

Sayfa No

ÖNSÖZ ……II
İÇİNDEKİLER……IV
ÖZET……..VI
SUMMARY……VII
ŞEKİLLER LİSTESİ……VIII
TABLOLAR LİSTESİ……X
KISALTMALAR LİSTESİ……XI
1. GİRİŞ……1
2. LİTERATÜR ÖZETİ VE İLGİLİ TEKNOJİLER……3
2.1. Paralel Hesaplama……5
2.2. Dağıtık Hesaplama……6
2.3. Bulut Bilişim……7
2.3.1. Bulut Bilişimin Temel Özellikleri……9
2.3.2. Bulut Bilişim Hizmet Modelleri……10
2.3.2.1. Hizmet Olarak Yazılım (Software as a Service - SaaS)……11
2.3.2.2. Hizmet Olarak Platform (Platform as a Service - PaaS)……12
2.3.2.3. Hizmet Olarak Altyapı (Infrastructure as a Service - IaaS)……12
2.3.3. Bulut Bilişim Kurulum Modelleri……12
2.3.3.1. Genel Bulut……13
2.3.3.2. Özel Bulut……14
2.3.3.3. Topluluk Bulut……14
2.3.3.4. Melez Bulut……15
2.4. Büyük Veri……15
2.5. Büyük Veri Teknolojileri……18
2.5.1. MapReduce……18
2.5.2. Dağıtık Dosya Sistemleri……20
2.5.2.1. Google Dosya Sistemi……21
2.5.3. Hadoop……23

2.5.3.1. Hadoop'un Mimarisi 24
2.5.3.2. Hadoop 1 25
2.5.3.3. Hadoop Dağıtık Dosya Sistemi (HDFS) 27
2.5.3.4. Hadoop 2 30
3. GÖRÜNTÜ İŞLEME VE DAĞITIK GÖRÜNTÜ İŞLEME 33
3.1. Görüntü Oluşturma 33
3.2. Görüntünün Temsili 36
3.3. Görüntü İşleme 39
3.4. Görüntü İşleme Uygulama Alanları 40
3.5. OpenIMAJ 41
3.5.1. OpenIMAJ Kütüphanesin Bileşenleri 42
3.6. Dağıtık Görüntü İşleme 45
4. BÜYÜK VERİ TEKNOLOJİLERİ İLE DAĞITIK GÖRÜNTÜ İŞLEME SİSTEMİ TASARIMI 48
4.1. OpenStack ile Sanal Makine ve Hadoop Kümesi Oluşturma 48
4.2. Oluşturulan Hadoop Kümesindeki Sanal Makinelere Erişim 63
5. UYGULAMALAR VE TESTLER 67
5.1. Dağıtık Görüntü İşleme Uygulaması (Görüntü Dosyalarından Dağıtık Görüntü İşleme ile Yüz Tespiti) 68
5.2. Dağıtık Görüntü İşleme Sistemi Değerlendirme Sonuçları 76
5.3. Dağıtık Video İşleme Uygulaması (Video Dosyalarında Sahne Değişim Tespiti) 81
6. SONUÇLAR 93
KAYNAKLAR 94
ÖZGEÇMİŞ 99

ÖZET

Bu tez çalışmasında, büyük veri alanında önemli bir yeri olan görüntü ve video dosyalarının dağıtık olarak işlenmesi konusu incelenmiştir. Görüntü ve video dosyalarının işlenmesinde kullanılmak üzere OpenStack bulut sistemi ile farklı ölçeklerde Hadoop kümeleri oluşturulmuştur. Dağıtık görüntü ve video işlemeyi gerçekleştirmek için Hadoop MapReduce programlama modeli ve OpenIMAJ kütüphanesi kullanılarak iki uygulama geliştirilmiştir. Bu uygulamalar geliştirilirken video ve görüntü dosyalarının Hadoop Dağıtık Dosya Sistemi (HDFS)'inden okunması, işlenmesi ve elde edilen sonuçların yazılması için gerekli olan ve Hadoop kütüphanesi tarafından sağlanmayan sınıflar yazılmıştır.

Geliştirilen uygulamalar farklı ölçeklerdeki Hadoop kümeleri üzerinde çalıştırılarak dağıtık görüntü ve dağıtık video işlemede Hadoop MapReduce programlama modeli test edilmiştir. Geliştirilen bu sistem ile hızlı bir şeklide artan görüntü ve video dosyalarının daha kısa sürede işlenmesi ve analiz edilmesi mümkün olmaktadır. Bu sisteminin OpenIMAJ kütüphanesi ile uyumlu geliştirilmesi OpenIMAJ kütüphanesinin sağladığı bir çok algoritmanın dağıtık olarak görüntü ve video işlemede kullanılabilmesini sağlayacaktır.

Anahtar Kelimeler: Dağıtık görüntü işleme, Paralel görüntü işleme, Büyük veri, Hadoop, MapReduce, YARN, Bulut bilişim, OpenIMAJ

SUMMARY

Distributed Image Processing

This thesis is about the distributed processing of image and video files, which have an important place in the field of big data. Hadoop clusters at different scales have been created with OpenStack cloud system to be used for processing image and video files. With using Hadoop MapReduce programming model and OpenIMAJ library, two applications have been developed to perform distributed image and video processing. Required classes for these applications to read and process the video and image files from Hadoop Distributed File System (HDFS) and write the obtained results, which are not provided by Hadoop framework, have been coded.

With the developed system, processing and analyzing image and video files is possible in shorter times. Development of this system in compatible with the OpenIMAJ library will make it possible to use many algorithms provided by OpenIMAJ in distributed processing of image and video files.

Key Words : Distributed image processing, Parallel image processing, Big data, Hadoop, MapReduce, YARN, Cloud computing, OpenIMAJ

ŞEKİLLER LİSTESİ

Sayfa No

Şekil 2.1. Mikroişlemcilerin zaman içindeki gelişimi [2]. 4
Şekil 2.2. Dağıtık sistem mimarisi [1]. 7
Şekil 2.3. Bulut bilişim metaforu [30]. 8
Şekil 2.4. Bulut bilişim servis modelleri 11
Şekil 2.5. Bulut kurulum modelleri 13
Şekil 2.6. Kelime sayma uygulamasının MapReduce aşamaları 19
Şekil 2.7. Google Dosya Sistemi (GFS) mimarisi [45]. 22
Şekil 2.8. GSF yazma kontrolü ve veri akışı [45]. 22
Şekil 2.9. GFS veri okuma işlemi [55]. 23
Şekil 2.10. Hadoop 1 ve Hadoop 2 bileşenleri 24
Şekil 2.11. Hadop 1 Mimarisi [59]. 25
Şekil 2.12. Hadoop 1 sürümünde işlerin yürütülmesi [62]. 27
Şekil 2.13. HDFS mimarisi [65] 28
Şekil 2.14. HDFS ortamına dosya yazma işlemi [2]. 29
Şekil 2.15. HDFS ortamından dosya okuma [2]. 29
Şekil 2.16. Hadoop 2'nin mimarisi [75]. 31
Şekil 3.1. Sayısal görüntünün elde edilmesi [76]. 34
Şekil 3.2. Elektromanyetik tayf ve görünür ışık [79]. 35
Şekil 3.3. Analog görüntünün (solda) örneklenme ve niceleme sonucu sayısallaştırma işlemi (sağda) [76]. 36
Şekil 3.4. Görüntüyü oluşturan pikseller 37
Şekil 3.5. 300x 300 piksele sahip (sol üst), 200 x 200 piksele sahip (sağ üst), 100 x 100 piksele sahip (sol alt) ve 50 x 50 piksele (sağ alt) sahip aynı boyutta olan görüntüler 38
Şekil 3.6. a) 128, b)64, c) 32, d)16, e) 8 parlaklık çözünürlüğüne sahip görüntüler [82]. . 39
Şekil 3.7. Görüntü işleme kullanım alanları [85]. 41
Şekil 3.8. OpenIMAJ kütüphanesiyle yapılabilen bazı işlemler 42
Şekil 3.9. Hadoop MapReduce ile dağıtık görüntü işleme 47
Şekil 4.1. OpenStack giriş ekranı 49
Şekil 4.2. Proje ön izlemesi 50
Şekil 4.3. OpenStack ile oluşturulan sanal makine örnekleri 50
Şekil 4.4. Küme şablonu oluşturma 1. adım 51
Şekil 4.5. Küme şablonu oluşturma 2. adım 51
Şekil 4.6. Küme şablonu oluşturma 3. adım 52
Şekil 4.7. Oluşturulan küme şablonu 52
Şekil 4.8. Anahtar çiftlerini listeleme ve yeni anahtar çifti oluşturma 53
Şekil 4.9. Ağ topolojisi oluşturma 53
Şekil 4.10. Küme şablonundan küme oluşturma 54
Şekil 4.11. Cluster1M2WTemplate şablonundan oluşturulan Hadoop kümesi 55

Şekil 4.12. Hadoop kümesinin ağ topolojisi 56
Şekil 4.13. Oluşturulan Hadoop kümesine ait genel bilgiler..... 57
Şekil 4.14. Ambari giriş ekranı 58
Şekil 4.15. Ambari web arayüzü 59
Şekil 4.16. Ambari ara yüzü ile HDFS' in yapılandırma ayarlarının yapılması 60
Şekil 4.17. Ambari ara yüzü ile MapReduce2 yapılandırma ayarlarının yapılması..... 61
Şekil 4.18. Ambari ara yüzü ile YARN yapılandırma ayarlarının yapılması..... 62
Şekil 4.19. Putty Key Generator ile akademi.pem dosyasının açılması..... 64
Şekil 4.20. PuTTY Key Generator ile private anahtar oluşturma 65
Şekil 4.21. PuTTY ile sanal makiye erişme (1) 66
Şekil 4.22. PuTTY ile sanal makiye erişme (2) 66
Şekil 5.1. Yüz görüntü dosyalarının adlandırılması 68
Şekil 5.2. Dağıtık görüntü işleme adımları..... 71
Şekil 5.3. PuTTY ile usta sanal makinesi erişilmesi 72
Şekil 5.4. PIPA data setinde bulunan görüntü örnekleri 75
Şekil 5.5. Dağıtık görüntü işleme uygulaması tarafından oluşturulan yüz dosyalarının yer aldığı SequenceFile dosyaları..... 75
Şekil 5.6. Dağıtık görüntü işleme algoritması ile tespit edilen yüzler 76
Şekil 5.7. Görüntü işleme uygulaması çalışırken Ambari metrikleri 77
Şekil 5.8. Görüntü işleme uygulaması çalışırken CPU kullanım grafiği 78
Şekil 5.9. Görüntü işleme uygulaması çalışırken hafıza kullanım grafiği 79
Şekil 5.10. Dağıtık görüntü işleme uygulaması çalışırken takip edilmesi 79
Şekil 5.11. Dağıtık görüntü işleme uygulaması işçi sanal bilgisayar sayısına göre yürütme süresi..... 81
Şekil 5.12. Dağıtık video işleme uygulamasının mimarisi..... 84
Şekil 5.13. Dağıtık video işleme uygulaması ile elde edilen sahne görüntülerinin HDFS ortamında listelenmesi..... 87
Şekil 5.14. Dağıtık video işleme uygulaması ile elde edilen sahneler 88
Şekil 5.15. Dağıtık video işleme uygulamasının çalışırken izlenmesi..... 89
Şekil 5.16. Dağıtık video işleme uygulaması çalışırken Ambari metrikleri 89
Şekil 5.17. Dağıtık video işleme uygulaması işçi sanal bilgisayar sayısına göre yürütme süresi..... 92

TABLOLAR LİSTESİ

Sayfa No

Tablo 4.1. Hadoop kümesindeki makine özellikleri 55
Tablo 5.1. Görüntü işleme uygulaması için yapılan yapılandırma 77
Tablo 5.2. Dağıtık görüntü işleme uygulamasının farklı ölçeklerdeki Hadoop kümesi üzerinde yürütülmesi 80
Tablo 5.3. Dağıtık video işleme uygulama aşamalarında harcanan süreler..... 85
Tablo 5.4. Video işleme uygulaması için yapılan 1. yapılandırma..... 90
Tablo 5.5. Video işleme uygulaması için yapılan 2. yapılandırma..... 91
Tablo 5.6. Dağıtık video işleme uygulamasının farklı ölçeklerdeki Hadoop kümesini üzerinde yürütülmesi 91

KISALTMALAR LİSTESİ

HDFS : Hadoop Dağıtık Dosya Sistemi
NAS : Network Attached Storage
SLA : Hizmet düzeyi sözleşmesini
SISD : Tek Komut Tek Veri (Single Instruction, Single Data)
SIMD : Tek Komut Çoklu Veri (Single Instruction, Multiple Data)
MISD : Çoklu Komut Tek Veri (Multiple Instruction, Single Data)
MIMD : Çoklu Komut Çoklu Veri (Multiple Instruction, Multiple Data)
AFS : Andrew Dosya Sistemi
RM : Resource Manager
YARN : Yet Another Resource Negotiator
JNI : Java Native Interface
MPI : Message Passing Interface
NIST :Amerikan Ulusal Teknoloji ve Standartlar Enstitüsü

1. GİRİŞ

Teknolojinin gelişmesi ile birçok alanda veri üretim hızı, üretilen verinin hacmi ve çeşidi artmıştır. Veri üretim hızının artması, üretilen veri hacminin büyümesi, verinin çeşitlenmesi sonucu bu verinin depolanması ve işlenmesi zorlukları ortaya çıkmıştır. Web 2.0 ile internetin yaygınlaşması sonucu internet üzerinden özellikle de sosyal ağlar aracılığı ile sunulan veri miktarı hızlı bir şekilde artmış ve artmaya da devam etmektedir. Uydu teknolojileri, güvenlik, savunma sanayi, sağlık gibi birçok alanda da elde edilen veri giderek artmaktadır. Kısacası teknolojinin gelişmesiyle neredeyse hemen hemen her alanda veri çeşitleri ve miktarında önemli artışlar görülmüştür.

Görüntüleme araçlarının gelişmesi ve yaygınlaşması sonucunda bu araçların birçok alanda kullanılması ile beraber üretilen diğer tüm veriler gibi görüntü ve video verilerinin hacmi ve çeşitliği de artmıştır.

Yapay zekâ, makine öğrenmesi, veri madenciliği, istatistik ve analiz gibi verinin çok önemli olduğu alanlarda, var olan tekniklerin kullanımının yaygınlaşması ve yeni tekniklerin kullanılması sonucu veriden daha anlamlı ve değerli bilgi elde edilmesi verinin değerine değer katmıştır. Veri anlamlandırıldığı ve bilgiye dönüştürüldüğü ölçüde değer kazanmaktadır. Verinin anlamlandırılıp bilgiye dönüşmesinde depolama ve işleme aşamaları önemli bir yer tutmaktadır. Veri ham olarak depolandıktan sonra işlenebileceği gibi üretildikten hemen sonra işlenerek de depolanabilir. Depolama ve işleme süreçlerinin sırası uygulama alanına, veri üzerinde yapılacak işlemlere göre değişebilir.

Veri hacminin (volume), çeşitliliğinin (variety) ve üretim hızının (velocity) artmasıyla birlikte Büyük Veri kavramı kullanılmaya başlanmıştır. Hizmet düzeyi sözleşmesini (Service Level Agreement) sağlayacak şekilde tek bir bilgisayarın kaynakları ile işlenmesi veya depolanması mümkün olmayan veriyi ifade etmek için Büyük Veri (Big Data) kavramı kullanılmaktadır [2]. Tanımdaki hizmet düzeyi sözleşmesi (Service Level Agreement-SLA) önemli bir noktadır. Çünkü veri bir network attached storage (NAS) ortamında depolanarak bir bilgisayar kaynağı ile işlenebilir, fakat bu işlemin hizmet düzeyi sözleşmesinde belirtilen süreden uzun olmaması gerekmektedir. Hatta bu veri bir bilgisayar kaynağı ile birkaç saatte işlenebiliyorken hizmet düzeyi sözleşmesinde, üretilen verinin gerçek zaman veya gerçek zamana yakın bir sürede işlenmesi istenebilir. Bu ve benzer durumlarda hizmet düzeyi sözleşmesini sağlamak için bir bilgisayar kaynağından daha fazla kaynağa ihtiyaç duyulur.

Dağıtık veya paralel hesapla yöntemleri kullanılarak daha kısa sürede daha fazla veri işlenebilir, işleme ve okuma süreleri hizmet düzeyi sözleşmesine uygun hale getirilebilir [2]. Görüntü ve video gibi verilerin boyutlarının büyük olmasının bir sonucu olarak depolama ve hesaplama kaynağına olan ihtiyacı artırmasından dolayı klasik yöntemler ile Hizmet Düzeyi Sözleşmesini sağlayacak şekilde bu verileri işlemek daha da zordur. Bu ihtiyaçlar, pahalı donanımlara sahip bilgisayarlar ile karşılanabileceği gibi sıradan bilgisayarlardan oluşan bir veya birden fazla bilgisayar kümesi kullanılarak oluşturulan dağıtık bir sistem ile de karşılanabilir.

Bulut bilişim, ağ, depolama, CPU gibi depolama kaynaklarını, yazılım kaynaklarını ve altyapı bileşenlerini bir ara yüz ile hizmet olarak sunan modern bir yaklaşımdır [3]. Bulut bilişim sayesinde daha esnek, hızlı ölçeklenme sağlanabilmektedir. Bulut bilişim ile normal özelliklere sahip sunucu ve bilgisayarlar bir araya getirilerek daha iyi sanal sunucular oluşturulabilir. Veya yüksek kaynağa sahip sunuculardan istenilen özelliklerde birden fazla sanal sunucu oluşturulabilir. Bulut ile kaynakların daha verimli kullanılması sağlanabilir [4].

Bu tez çalışmasında, büyük veri alanında hem depolama hem de işleme açısından önemli bir yer tutan görüntü ve video dosyalarının dağıtık olarak işlenmesi için gerekli yazılımlar yapılmış, büyük veri teknolojileri yardımıyla bir dağıtık sistem geliştirilmiş ve bu sistem bulut üzerinde çalıştırılmıştır.

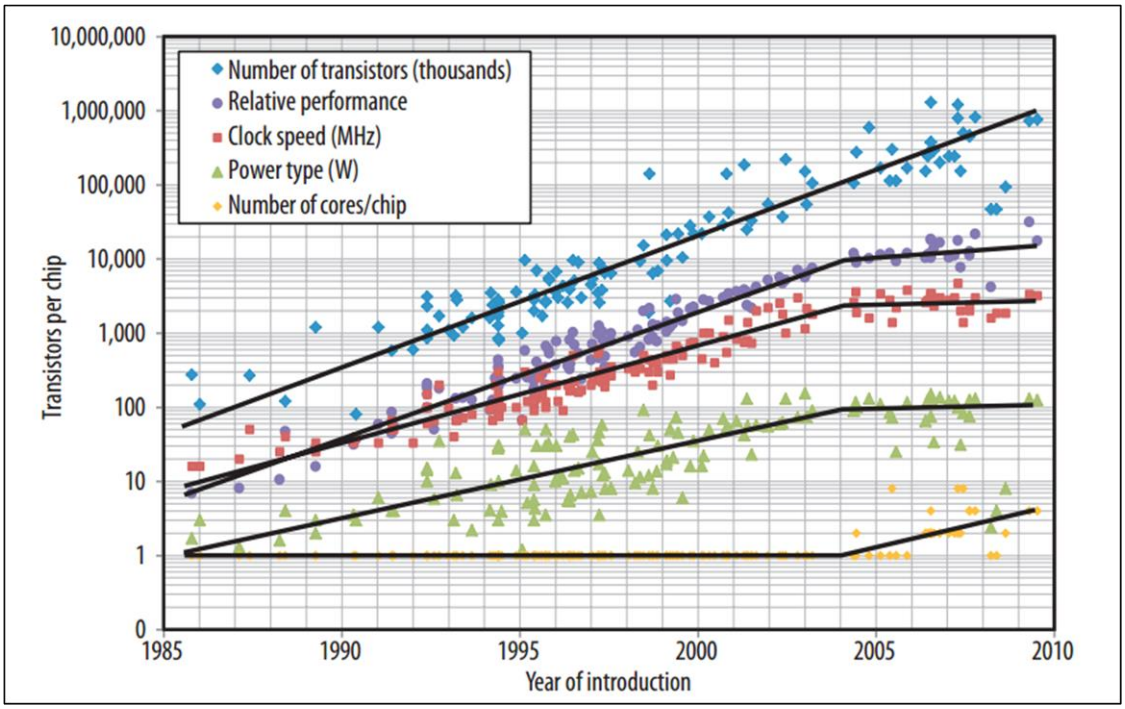
Tezin ikinci bölümde ilgili literatür özetlenmiş ve tezin arka planını oluşturan teorik konular ve ilgili teknolojiler anlatılmıştır. Üçüncü bölümde görüntü ve dağıtık görüntü işleme konuları özetlenerek kullanılan yaklaşımlar ve uygulama örnekleri özetlenmiştir. Beşinci bölümde tez kapsamında yapılan çalışmanın detayları, tasarlanan sistemin mimarisi, kullanılan algoritmalar ve veri setleri, uygulanan testler ve bu testlerin sonuçları paylaşılmıştır. Tezin altıncı ve son bölümünde ise tez çalışması sonucunda elde edilen kazanımlar ve literatür katkısı sunulmuştur.

2. LİTERATÜR ÖZETİ VE İLGİLİ TEKNOJİLER

Son yıllarda oldukça popüler hale gelen bulut bilişim, büyük veri, paralel hesaplama ve MapReduce teknolojileri kullanılarak, sıradan bilgisayarlardan oluşan bilgisayar kümeleri oluşturularak büyük boyutlu verilerin işlenmesi ve analiziyle alakalı birçok çalışma yapılmıştır [2, 5-18]. Mera, D. ve arkadaşları Hadoop ve Storm ile çoklu medya verilerinin özellik çıkarma işlemi için bir karşılaştırma yapmışlardır [6]. Sabarad, A.K, ve arkadaşları dağıtık sistemlerde renk ve doku özellik vektörlerini çıkarma işlemi ile alakalı bir çalışma yapmışlardır. Bu çalışmada sequencefile kullanılmadığında ve resimlerin boyutu 64 MB'dan küçük (10M) olduğu durumlarda node sayısının 3'ün üstüne çıkarılması ile işlem süresinin azalmayarak arttığı gösterilmiştir [7]. El-Kazzaz, S. ve El-Mahdy, A. Landsat uydusundan alınan görüntüleri Hadoop ile işleyerek mayınlı alanların tespitini yapmışlardır. Li ve arkadaşları 30 milyon görüntüden oluşan ve 2 milyonu coğrafik konum etiketli, 500'e yakın kategoriye sahip bir veri kümesi üzerinde SVM algoritması ve Hadoop kümesi kullanarak sınıflandırma işlemi gerçekleştirmişlerdir [11]. Yan ve arkadaşları 16 Hadoop düğümünden oluşan sistem üzerinde 250000 görüntüden oluşan veri ile test ettikleri görüntülerin anlamsal içeriğini bulmaya yönelik bir algoritma önermişlerdir [16]. Shi ve arkadaşları tarafından MapReduce programla modelini kullanarak içerik tabanlı görüntü geri alma yönetimi önerilmiştir [14]. Yang ve arkadaşları MIFAS adını verdikleri Hadoop sistemi üzerine kurdukları medikal görüntülerin depolanıp paylaşıldığı bir sistem tasarlamışlardır [17]. Bu sistemde Hadoop Dağıtık Dosya sistemini kullanmışlardır. Shelly ve Raghava Hadoop MapReduce programlama modelini kullanarak iris tanıma ile alakalı bir çalışma yapmışlardır [13]. Kocakulak ve Temizel Hadoop ile balistik görüntü tanıma işlemini gerçekleştirmişlerdir [10].

Bilgisayarın hesaplama gücü gün geçtikçe artmaktadır. Hesaplama gücündeki bu artış bireysel kullanıcılar için yeterli görünse de, teknolojinin gelişmesi ile ortaya çıkan ihtiyaçların karşılanabilmesi için yeterli olamamaktadır. Verinin sürekli artması, endüstri ihtiyaçları göz önüne alındığında bilgisayarların performans artışının sürekli olarak sürdürülmesi gerekir. Başta bilişim teknolojileri alanında faaliyet gösteren firmalar ve kurumlar olmak üzere eğitim, sağlık, bilim, savunma, finans gibi hemen hemen her alanda kalite ve verimliliğin artırılması, yeniliklerin sağlanması, araştırma geliştirme işlemlerinin yürütülmesi gibi birçok işlem için daha büyük hesaplama gücüne sahip bilgisayar ihtiyacı

giderek artmaktadır. Moore yasasına göre entegre devrelerindeki transistor sayısının her iki yılda iki katına çıkacağı öngörülmüştür [19]. Transistor sayısındaki artış bilgisayarın hızlanması dolayısı ile hesaplama gücünün artması anlamına gelmektedir. Bu yasa, günümüzde temel olarak geçerliğini sürdürse de, performans artış hızı bir yerden sonra yavaşlamaya başlamaktadır. Var olan donanımsal bazı kısıtları ortadan kaldıracak yeni teknolojik gelişmelerin gerçekleşmemesi durumunda, yasanın geçerliğini gelecekte sürdürmesi mümkün olmayabilir [20]. Şekil 2.1'de gösterilen grafikte mikroişlemcilerin zamanla gelişimi gösterilmiştir.



Şekil 2.1. Mikroişlemcilerin zaman içindeki gelişimi [2].

Bu grafikte transistor sayısı Moore yasasına göre sürekli olarak artmıştır. Performans 2004 yılından sonra daha yavaş artmaya başlamıştır. Bunun sebebi işlemci saat hızı ile gerekli olan güç tüketimi arasındaki ilişkidir. İşlemci güç tüketiminin çok artması istenen bir durum değildir. Daha büyük işlemci saat hızı için çok daha fazla güç harcamak gerektiğinden, 2004 yılından itibaren işlemci saat hızı fazla artmamıştır. Bunun yerine işlemci mimarisinde güncellemeler yapılarak aynı işlemci üzerinde birden fazla çekirdek yerleştirilmeye başlanmıştır [20]. İşlemci mimarisindeki bu güncelleme ile performans artışı sağlamak için yazılan yazılımların da bu mimariye uygun olarak geliştirilmesi gerekmektedir. Paralel programlama ile paralel olarak gerçekleştirilebilecek işlemler

parçalara ayrılır daha sonra bu parçalar işlemci çekirdeklerine paylaştırılır ve bunun sonucunda işlemler daha hızlı gerçekleştirilir.

2.1. Paralel Hesaplama

İşlemci mimarilerindeki güncelleme ile işlemciler, birden fazla işlemci çekirdeğinden oluşabilmektedir. Aynı anda birden fazla işlemci çekirdeği veya birden fazla işlemci kullanılarak işlemlerin daha hızlı bir şekilde yapılması fikri paralel hesaplama fikrinin doğmasında etkili olmuştur. İşlem hızının ve işlem süresinin önemli olduğu uygulamalarda hizmet düzeyi sözleşmesini (Service Level Agreement) sağlayacak şekilde işlemlerin gerçekleştirilmesinde paralel hesaplamanın önemli bir yeri vardır. Farklı uygulama mimarileri olan paralel hesaplama temelde işlemlerin veya verinin paralelleştirilmesi esasına dayanmaktadır [21]. Paralel hesaplama mimarilerinin daha kolay incelenebilmesi için Flynn taksonomisi [22] olarak bilinen sınıflandırmayı incelemekte fayda vardır. Bu sınıflandırmaya göre paralel bilgisayarlar, bir seferde işleyebildiği komut sayısına ve veriye göre 4 farklı sınıfa ayrılmıştır.

SISD – Tek Komut Tek Veri (Single Instruction, Single Data): Klasik Von Neumann [23] mimarisine sahip bilgisayarlardır. Aynı anda bir komut ve tek bir veriyi işleme yeteneğine sahiptir. Bu bilgisayarlar seri hesaplama yaparlar [22].

SIMD – Tek Komut Çoklu Veri (Single Instruction, Multiple Data): Çoklu işlemcili tek bir program belleğine sahip bilgisayarlardır. Her işlemci aynı anda aynı komutu farklı veriler için işleyebilir [22].

MISD – Çoklu Komut Tek Veri (Multiple Instruction, Single Data): Çoklu işlemcili çoklu program belleği ve tek bir veri hafızası olan bilgisayarlardır. Her işlemci kendisine gelen veri için aynı anda farklı komutlar çalıştırabilir. Bu mimariye göre bir paralel bilgisayar geliştirilmemiştir fakat modern grafik işlemciler için bu mimari kullanılmaktadır [24].

MIMD – Çoklu Komut Çoklu Veri (Multiple Instruction, Multiple Data): Bu mimari birden fazla işlemciden oluşur. İşlemciler aynı anda farklı veriler üzerinde farklı komutları çalıştırabilirler. Çok çekirdekli modern işlemciler bu mimariye sahiptirler [24].

Paralel hesaplama özel tasarım gerektiren çok işlemcili veya çok çekirdekli donanımlar gerektirdiği için yüksek hesaplama gücü gereken durumlarda maliyet daha da yükselmektedir. Bu yüzden daha az maliyetli dağıtık hesaplama sistemleri kullanılmaya başlanmıştır.

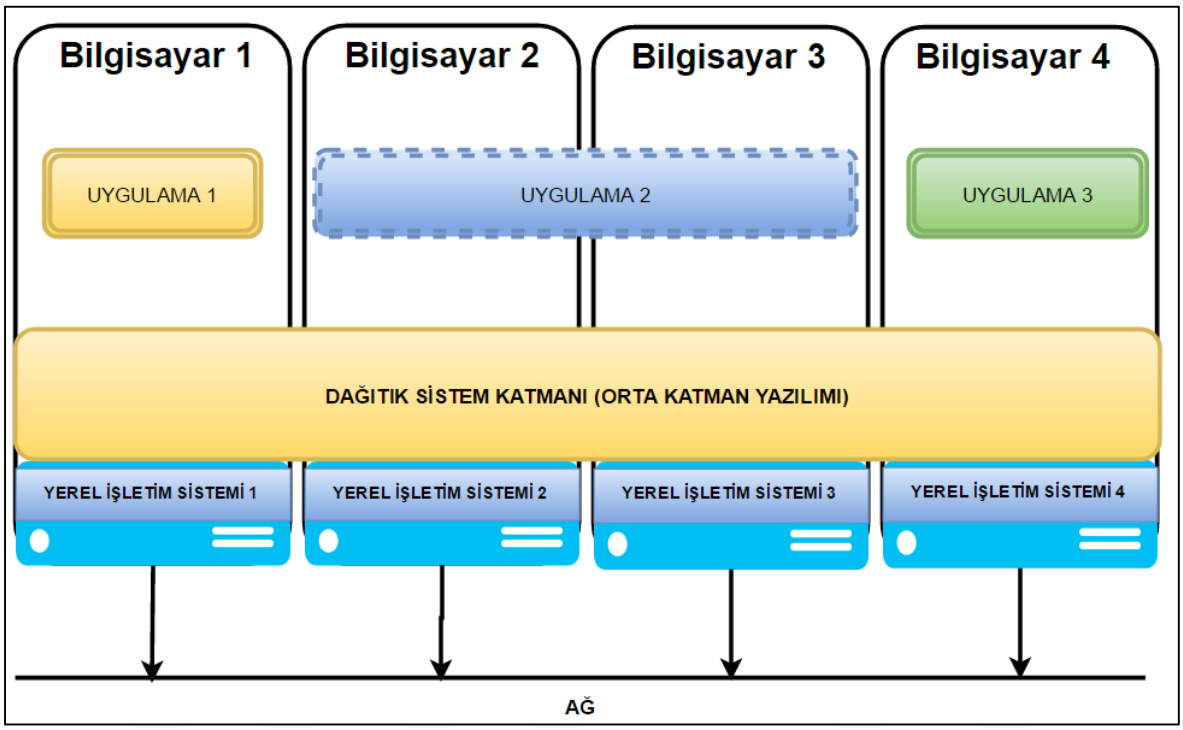
2.2. Dağıtık Hesaplama

Dağıtık sistemlerin literatürde birçok tanımı bulunmaktadır. En genel haliyle dağıtık sistem, bir birinden bağımsız bilgisayarlardan oluşan ve kullanıcılara tek bir bilgisayarmış gibi görünen bilgisayar topluluğundan oluşan sisteme denir [1]. Dağıtık hesaplama ise dağıtık sistemler aracılığı ile hesaplama, analiz, problem çözümü gibi işlemlerin gerçekleştirilmesidir. Dağıtık hesaplamada işler parçalara bölünür ve bu parçalar dağıtık sistemi oluşturan bilgisayarlara paylaştırılarak çözülür.

Bir dağıtık sistem; orta katman yazılımı sayesinde işletim sitemi, donamım veya haberleşme gibi alt seviyelerde gerçekleşen işlemleri, hata toleransını da sağlayacak şekilde kendisi sağlamalı ve kullanıcılardan bu işlemleri mümkün olduğunca gizleyerek kullanıcıları bu işlemlerden soyutlamalıdır. Dağıtık sistem kullanıcıya tek bir bilgisayarmış gibi görünmelidir. Sistem kaynaklarına nasıl erişeceği, bu kaynakların yerleri ve sistem kaynaklarının taşınması ve yedeklenmesi, kaynakların birbirlerinde soyutlanması gibi işlemler kullanıcıdan gizlenmelidir. Ayrıca bir dağıtık sistem oluşan hataları tolere etmelidir.

Dağıtık sistemlerin ortaya çıkmasını sağlayan en temel neden yüksek depolama ve hesaplama kaynağına olan ihtiyaçtır. Bu ihtiyacın pahalı ve çok yüksek özelliklere sahip süper bilgisayarlar ile karşılanması yerine daha az maliyetli sıradan bilgisayarlar ile karşılanabilmesi, hata toleranslı, ölçeklenebilir sistemler olmaları dağıtık sistemlerin popülerleşmesini sağlamıştır. Dağıtık sistemlerin kaynak ve veri paylaşımını sağlaması ve bunlara erişimi kolaylaştırması önemlerini daha da artırmıştır.

Dağıtık sistemlerde bir birinden bağımsız heterojen bilgisayarları kullanıcılara (kişi veya program) tek bir bilgisayarmış gibi sunmak için genellikle bir orta katman yazılımı kullanılır. Bu orta katman, heterojen bilgisayarlar arasında gerçekleşen haberleşme kaynak paylaşımı gibi alt seviyede gerçekleşen işleri yönetir. Şekil 2.2’ de gösterildiği gibi ağ ile birbirine bağlı 4 heterojen bilgisayar işletim sistemi üzerine kurulu bir dağıtık sistem gösterilmiştir. Burada dağıtık katman (orta katman yazılımı) heterojen sistem ile uygulamalar ve kullanıcılar birbirinden soyutlanmakta ve uygulama 1, uygulama 2 ve uygulama 3 uygulamalarına aynı ara yüzü sunulmaktadır. Bu uygulamalardan uygulama 2, bilgisayar 1 ve bilgisayar 2 üzerine dağıtık katman aracılığı ile dağıtılmıştır.



Şekil 2.2. Dağıtık sistem mimarisi [1].

2.3. Bulut Bilişim

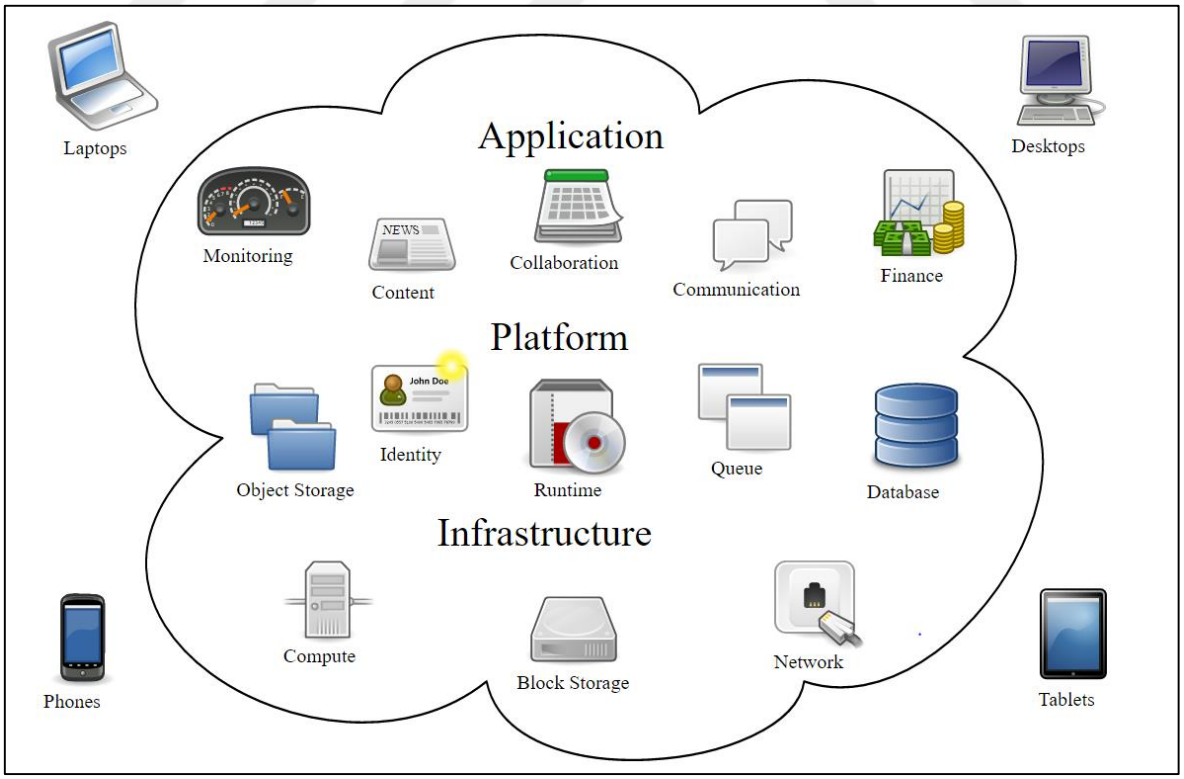
Bulut bilişim için yapılan farklı tanımlar bulunmaktadır [3, 25-28]. Bulut bilişim gerçek zamana yakın olarak organizasyonlara hızlı ve etkili bir şekilde kaynak ekleme ve çıkarmaya yardımcı olan yaklaşımlar dizisi olarak tanımlanabilir [3]. Amerikan Ulusal Teknoloji ve Standartlar Enstitüsü (NIST) bulut bilişimi, minimum hizmet sağlayıcı etkileşimi ve yönetim çabası ile hızlı bir şekilde alınıp salıverilebilen, yapılandırılabilir hesaplama kaynaklarının (ağlar, sunucular, veri depoları, uygulamalar, hizmetler vb.) paylaşıldığı havuza; istendiğinde uygun bir şekilde ağ bağlantısı sağlayan bir model olarak tanımlanmaktadır [25].

Bulut bilişimdeki bulut, hesaplama gücünden hesaplama alt yapısına, uygulamalardan iş süreçlerine kadar servis olarak sunulabilen her şeyin servis olarak sunulabilmesini sağlayan araçları tanımlamaktadır. Bulut; donanımları, ağı, depolama servislerini ve ara yüzleri içerir. Yeni kaynakların eklenmesi ile büyüyerek genişleyen veya kaynakların

çıkarılması küçülen bulut, kelimenin ifade ettiği gerçek anlamdaki bulutlara benzemektedir. Aslında bu özellik bulutun esnek ve ölçeklenebilir olmasını ifade etmektedir.

Bulut bilişim hem iş modeli hem de teknoloji ile alakalı olduğu için klasik veri merkezlerinden daha fazlasını ifade etmektedir. Bulut bilişim, internet evriminin bir sonraki aşaması olarak da görülebilir. Bulut servisleri sayesinde; yazılım, altyapı ve depolama hizmetleri ayrı ayrı veya bir platform olarak kullanıcının ihtiyaçlarına özel olarak internet aracılığı ile servis olarak sunulur. Ayrıca bulut bilişim için "kullandığın kadar öde" yaklaşımı benimsenmiştir. Bu yaklaşım sayesinde kullanıcılar, web üzerinden istediği hesaplama kaynaklarını kolaylıkla sistemlerine dahil edip çıkarmanın yanında bu kaynaklara da kullandıkları kadar ücret ödemektedirler.

Bulut bilişim yaklaşımı ile, dağıtık halde bulunan kaynaklar bir yazılım aracılığı ile bir araya getirilerek bir dağıtık sistem oluşturulabilir. Bu dağıtık sistem üzerinde sanal makineler başlatılarak kullanıcılara sunulabilir. Bunun yanında yüksek kapasiteli sunucu bilgisayarların kaynakları da bulut bilişim yaklaşımı ile sanallaştırma ve çoklu kiracılı (multi-tenant) mimari kullanılarak kullanıcılara dağıtılabilir. Sanal makine kaynakları elde olan kaynaklar dahilinde ihtiyaca göre artırılıp azaltılabilir [29].



Şekil 2.3. Bulut bilişim metaforu [30].

Bulut bilişim veriye, bilgiye, hesaplama kaynağına farklı seviyelerde erişim ve paylaşım alışkanlığına yeni bir yaklaşım getirmiştir. Bulut bilişim sayesinde alt seviyede birbirinden bağımsız olabilen kaynakları ayrı ayrı ayarlamaya ve yönetmeye gerek kalmaz. Kullanıcılar ihtiyaç duydukları kaynakları kaynak havuzundan kolay bir şekilde seçerek kendi hesaplama kaynaklarını kolaylıkla oluşturabilirler.

Genelde ticari olan ve bulut bilişim hizmet sağlayıcıları tarafından sağlanan bu hizmetler bilim dünyasının ilgisini çekmiş ve birçok projede kullanılmıştır [31]. OpenStack [32, 33] OpenNebula [34] ve Nimbus [35] gibi açık kaynak kodlu bulut bilişim yazılımları ile kullanıcılar veya kurumlar kendi bulut bilişim sistemlerini kurabilmektedirler.

2.3.1. Bulut Bilişimin Temel Özellikleri

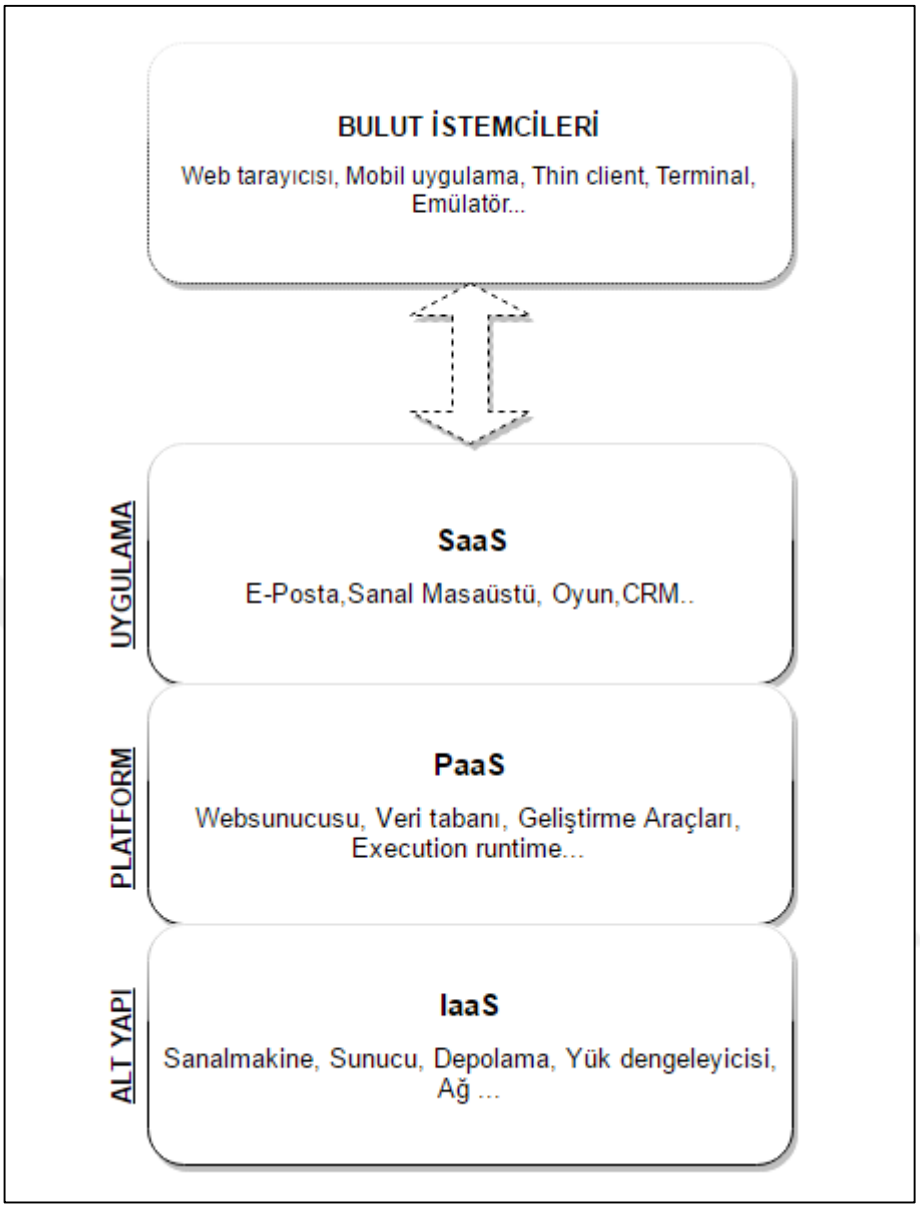
Amerikan Ulusal Teknoloji ve Standartlar Enstitüsü (NIST) bulut modelinin beş temel özellik, üç servis modeli ve dört kurulum modelinden oluştuğunu belirtmiştir [25].

- **Talep üzerine self servis (On-demand self-service) :** Kullanıcılar, servis sağlayıcının desteğine ihtiyaç duymadan sunucu zamanı ve depolama gibi kendi hesaplama kapasitesiyle alakalı özellikleri ihtiyaç halinde otomatik olarak değiştirebilmelidir**.**
- **Geniş ağ erişimi (Broad network access) :** Sunulan hizmetler, mobil cihazlar, iş istasyonları, tabletler, dizüstü bilgisayarlar gibi çeşitli istemciler vasıtası ile ağ üzerinden standart mekanizmalar ile erişilebilir olmalıdır.
- **Kaynak Havuzu (Resource pooling) :** Sağlayıcı, hesaplama kaynakları için havuz oluşturmalıdır. Sağlayıcı bu kaynakları çoklu kiracılı (multi-tanent) modele göre birden fazla kullanıcıya sunabilmelidir. Çeşitli fiziksel ve sanal kaynaklar kullanıcının isteğine uygun olarak atanabilmelidir. Kullanıcılardan bu kaynakların bulunduğu fiziksel yer gibi alt seviye bilgiler gizlenmelidir.
- **Hızlı Esneklik (Rapid elasticity) :** Bir bulut sisteminin kapasitesi esnek olmalıdır. İhtiyaçlara göre yeni kaynakların eklemesi veya kaynakların salıverilmesi ile kapasitenin esnek bir şekilde değiştirilmesi gerekebilir. Ölçeklenme mümkün olduğunca hızlı bir şekilde gerçekleştirilmelidir. Sitemin kaynakları kullanıcılara sınırsız olarak sunulmalı, kullanıcı istediği anda yeni kaynakları alabilmelidir.
- **Hizmetlerin Ölçülebilmesi (Measured service) :** Bir bulut sisteminde sunulan hizmetlerin ölçülebilir olması gerekmektedir. Depolama, işlem, bant genişliği, ağ trafiği, aktif kullanıcılar gibi farklı detay seviyelerinde sunulan hizmetlerin servis sağlayıcı ve

kullanıcılar için izlenebilir ve ölçülebilir olması gerekmektedir. Bulut bilişimde kullanıcı deneyimini artırmak için de bu özelliğin sağlanması oldukça önemlidir.

2.3.2. Bulut Bilişim Hizmet Modelleri

Bulut bilişimin hizmet modelleri olarak adlandırılan üç farklı temel hizmet modeli bulunmaktadır. Bu modellere göre bulut bilişim hizmetleri kullanıcılara sunulmaktadır. Bunlar Hizmet olarak Yazılım (Software as a Service – **SaaS**), Hizmet olarak Platform (Platform as a Service – **PaaS**) ve Hizmet olarak Altyapı'dır (Infrastructure as a Service – **IaaS)** [36]. Şekil 2.4'de Bulut bilişim hizmet modelleri gösterilmiştir. Bu modeller ayrı ayrı aşağıda açıklanmıştır.



Şekil 2.4. Bulut bilişim servis modelleri

2.3.2.1. Hizmet Olarak Yazılım (Software as a Service - SaaS)

Bu hizmet modelinde yazılımlar hizmet olarak sunulur. Hizmet sağlayıcısı tarafından bulut alt yapısı üzerine yazılımlar yüklenip ve kurulum ayarları yapıldıktan sonra kullanıcılara sunulur. Bulut kullanıcısı bulut hizmet sağlayıcısının sunduğu bu yazılımlara bilgisayar, tablet gibi çeşitli cihazlar kullanarak ağ veya internet üzerinden erişir ve sunulan yazılımı kullanır [28]. Kullanıcılar yazılımın çalıştığı platformlara müdahale edemezler. Bu şekilde kullanıcılar alt yapı, donanım, yazılım kurulumu, güvenlik gibi maliyetleri

karşılamak zorunda kalmaz. Microsoft Office 365 ve Google Drive bu modele örnek olarak verilebilir.

2.3.2.2. Hizmet Olarak Platform (Platform as a Service - PaaS)

İşletim sistemi, yazılım yürütme ortamı, veri tabanı gibi platformların hizmet olarak sunulduğu bulut hizmet modelidir. Bu modelde sistem yönetimi hizmet sağlayıcı tarafından gerçekleştirilir. Örneğin kullanıcı bir Java web projesini yayınlamak istediğinde bir Java uygulama sunucusuna ihtiyaç duyar. Kullanıcı, uygulama sunucusu kurulumu ve ayarları ile uğraşmadan bu platformu bulut sağlayıcılarından hizmet olarak alıp kullanabilir. Böylelikle uygulama sunucusu kurulumuna ve ayarlarına hem zaman harcamak hem de ücret ödemek zorunda kalmaz. Google App Engine ve Amazon Elastic Beanstalk bu hizmet modeline örnek verilebilir [36].

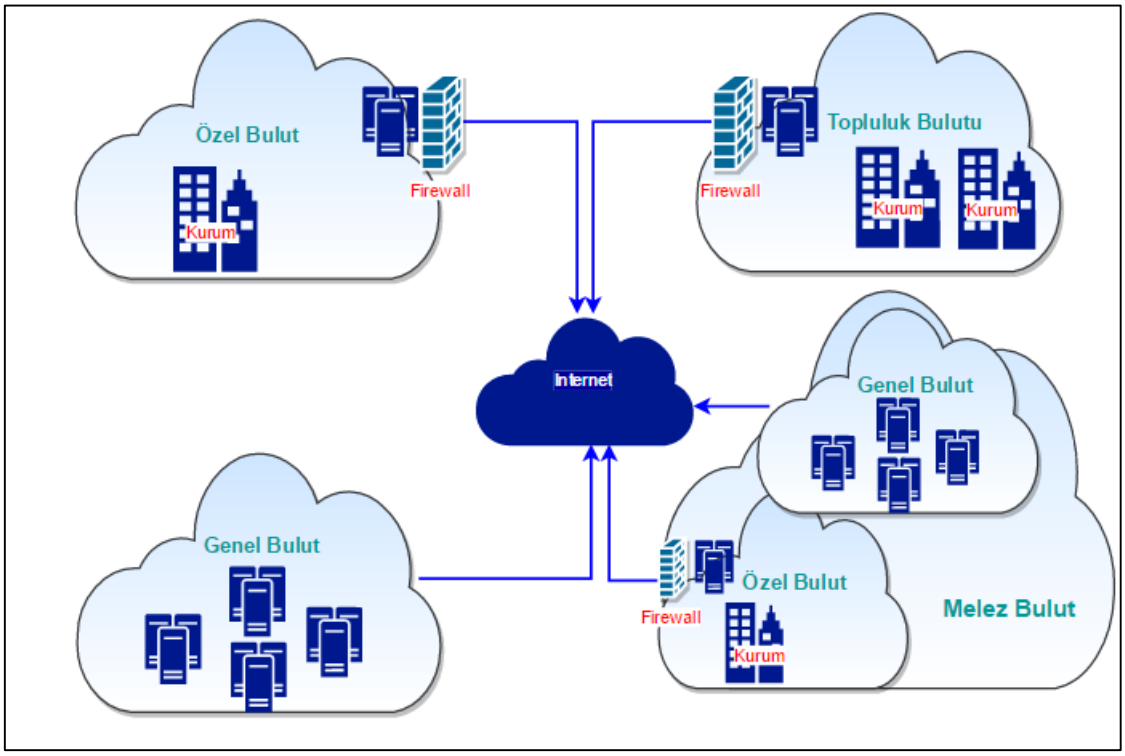
2.3.2.3. Hizmet Olarak Altyapı (Infrastructure as a Service - IaaS)

Ağ, depolama, hesaplama kaynaklarının sunucu ve sanallaştırma işlemleriyle alt yapı olarak sunulduğu hizmet modelidir. Kullanıcı bu alt yapıyı kullanarak kendi sanal makinelerini çalıştırıp, bu makinelere istediği işletim sitemini kurabilir. Bu modelde fiziksel alt yapı hizmet olarak sunulur. Sürekli kullanılmayan bir sistemin fiziksel alt yapısının oluşturulması o alt yapının kiralanmasına göre maliyetli olur. Bunun yerine kullanıcılar ihtiyaç duydukları fiziksel altyapıyı bulut hizmeti olarak alabilir. Böylelikle hızlı bir şekilde bu sistemleri kullanabilirler. Bu modelde kullanıcıların bulut alt yapısına müdahale yetkisi yoktur. İşletim sistemini, depolama kaynaklarını, işletim sistemi üzerine kurulacak yazılımları yönetebilir ve kontrol edebilir [28].

2.3.3. Bulut Bilişim Kurulum Modelleri

Bulut bilişimin NIST tarafından tanımlanan dört farklı kurulum modeli bulunmaktadır. Bunlar “Özel bulut”, “Genel bulut”, “Topluluk bulut”, “Melez bulut” modelleridir. Bulut bilişim alt yapısı kurulurken ihtiyaçlara göre yukardaki kurulum modellerinden biri esas alınır. Hangi modelin uygulanacağı kurum veya kuruluşun ihtiyaçlarına göre belirlenmelidir. Bir kurum, başka kurum ve kuruluşlar tarafından ortak olarak kullanılan genel bulutu seçebilirken; kendi bulut sistemini kurup sadece kendi kurumu içinde kullanmak isteyen

başka bir kurum ise özel bulut alt yapısını kurabilir. Hangi modelin seçileceği kurum ve kuruluşların bütçe ve ihtiyaçları ile alakalı bir tercihtir. Bulut kurulum modelleri Şekil 2.5' de gösterilmiştir. Yukarda isimleri verilen kurulum modelleri aşağıda alt başlıklar altında açıklanmıştır.



Şekil 2.5. Bulut kurulum modelleri

2.3.3.1. Genel Bulut

Genel bulut, bulut hizmetlerinin hizmet sağlayıcı tarafından kamuya açık olarak ücretli veya ücretsiz olarak sunulduğu bulut kurulum modelidir. Genel bulut, özel buluttun sağladığı güvenlik ve alt yapı hizmet seviyesinden daha az hizmet seviyesi ihtiyacına sahip bireysel kullanıcılar tarafından yaygın olarak kullanılmaktadır. Bununla birlikte kurum ve kuruluşlar içeriği hassas olmayan çevrimiçi doküman ve dosyaları depolama ve sunma gibi hizmetler için genel bulutları kullanabilirler. Hizmet Olarak Yazılım (SaaS) olan çevrimiçi ofis uygulamalarının bulut ortamında herkese sunulması genel buluta örnek olarak verilebilir.

Genel bulut halka açık olduğu, özel buluta göre daha geniş kaynak havuzuna sahip olduğu için kullanımdan dolayı kaynaklanan dalgalanmalara karşı daha hızlı ölçeklenebilir. Ayrıca genel bulut, büyük kaynak havuzuna ve geniş kullanıcı kitlesine sahip olmasından

dolayı maliyet avantajı sağlar. Bazı kurum ve kuruluşlar pazar elde etmek için daha uygun maliyete veya gelirlerini reklamlardan sağlayarak ücretsiz olarak genel bulut hizmeti sağlayabilmektedirler. Genel bulut hizmet sağlayıcıları genelde kullandığın kadar öde modelini kullandıkları için kullanıcılar kullanmadıkları kapasite için ödeme yapmak zorunda kalmazlar. Genel olarak genel bulut birden çok fiziksel sunucu ve birden fazla ağ gibi bileşenlerden oluştuğu için herhangi bir bileşenin fiziksel olarak arızalanması sonucunda oluşabilecek problemlerden daha az etkilenirler. Genel bulutun her yerden internet aracılığı ile erişilebilir olması kullanıcılar için büyük avantajlar oluşturmaktadır.

2.3.3.2. Özel Bulut

Kurum ve kuruluşların kendi özel ağı içinde bulunan bütün kullanıcılara veya özel bir kullanıcı kitlesine bulut hizmetini sağlamak için kendi bulut ortamını kurduğu bulut kurulum modelidir. Bulut bilişimde kurulum modelleri kullanıcı kitlesine göre sınıflandırılır. Özel bulut sadece belirli bir kullanıcı kitlesi için hizmete açıktır. Bulutun özel olması için kurumun binasında olma şartı yoktur. Özel bulut ayrı bir fiziksel kaynak havuzuna sahiptir. Bu kaynaklara güvenliği sağlanmış özel bir ağ veya şifrelenmiş bir bağlantı ile erişim sağlanır. Onun dışındaki kullanıcıların hizmetine kapalıdır. Özel bulut, genel buluta göre daha maliyetlidir. Fakat her ne kadar genel buluta göre maliyetli olsa da geleneksel yöntemlerle kaynak paylaşımına göre daha tasarrufludur. Sadece belli bir kullanıcı kitlesine erişim sağlaması ve firewall arkasına kurulabilmesi, fiziksel kaynak havuzunun ayrı olması gibi sağladığı avantajlardan dolayı genel buluta göre gizlilik ve güvenliği daha iyi sağlar. Özel bulut bir kuruma özel olarak kurulduğu için kurum, kendi özel ihtiyaç ve isteklerine göre bulut hizmetini özelleştirebilme ve daha fazla kontrol edebilme imkânına sahiptir.

2.3.3.3. Topluluk Bulut

Bu bulut modelinde belirli kurum ve kuruluşlardan oluşan belirli bir topluluğa hizmet verilir. Genel bulutta olduğu gibi herkese açık değildir. Özel bulutta olduğu gibi de sadece bir kurum ve kuruluşa ait değildir. Bu bulut modelinde bulut hizmeti genelde benzer ihtiyaç ve isteklere sahip bir topluluk tarafından ortak olarak kullanılır. Sağlık veya eğitim kurumları gibi benzer ihtiyaç ve isteklere sahip kurumlar için topluluk bulut modeli kullanılabilir. Ayrıca benzer ihtiyaç ve istekleri olan müşteriler için de topluluk bulutu üzerinden hizmet verilebilir

2.3.3.4. Melez Bulut

Yukarda belirtilen kurulum modellerinin beraber kullanıldığı bulut modelidir. Kurumlar hassas olan güvenlik ve gizlilik gerektiren işlemler için özel bulut kullanıp hassas olmayan ve gizlilik gerektirmeyen işler için daha az maliyeti olan ve daha ölçeklenebilir olan genel bulutu tercih edebilirler. Bu şekilde verimliklerini daha yükseltebilirler.

Melez bulut farklı şekillerde oluşturulabilir. Hizmet sağlayıcıları komple bir hizmet olarak melez bulut hizmetini sunabilecekleri gibi özel bulut hizmeti ve genel bulut hizmeti sağlayan hizmet sağlayıcıları bir araya gelerek melez buluttu sunabilirler. Ya da kurum kendi özel bulutunu kurup genel bulutunu bir hizmet sağlayıcıdan sağlayarak bunları entegre edebilir.

2.4. Büyük Veri

Bu tez, büyük veri içinde önemli bir yeri olan çoklu medya verilerinden görüntü ve video dosyalarının büyük veri teknikleri ile dağıtık olarak işlenmesini konu almaktadır. Bundan dolayı büyük veri kavramının, özelliklerinin ne olduğu ve niçin kullanılması gerektiği tezin anlaşılması için önemlidir.

Teknolojik gelişmeler sonucu uydu teknolojileri, güvenlik, savunma sanayi, sağlık gibi hemen hemen her alanda veri üretim hızı, üretilen verinin hacmi ve çeşidi artırmıştır. Web 2.0 ile internetin yaygınlaşması sonucu internet ortamıyla özellikle de sosyal ağlar aracılığı ile üretilen ve sunulan veri miktarı hızlı bir şekilde artmaktadır. Parçacık çarpıştırıcılarından elde edilen veriden DNA'nın içerdiği veriye, petrol arama firmalarının sismik verilerinden, sensörlerden alınan veriye, görüntü verilerinden konum verilerine kadar her gün birçok farklı kaynaktan farklı özelliklere sahip devasa boyutlarda birçok veri üretilmektedir.

Verinin hacim (volume), çeşitlilik (variety), üretim hızının (velocity) artmasıyla birlikte "Büyük Veri" kavramı kullanılmaya başlanmıştır [37, 38]. Hizmet düzeyi sözleşmesini (Service Level Agreement) sağlayacak şekilde bir bilgisayar kaynağı ile işlenmesi veya depolanması mümkün olmayan veriyi ifade etmek için "Büyük Veri" (Big Data) kavramı kullanılmaktadır [2]. Tanımdaki hizmet düzeyi sözleşmesi (Service Level Agreement) önemli bir noktadır. Çünkü veri bir network attached storage (NAS) ortamında depolanarak bir bilgisayar kaynağı ile işlenebilir, fakat bu işlemin hizmet düzeyi sözleşmesinde (Service Level Agreement) belirtilen süreden uzun olmaması gerekmektedir. Hatta bu veri bir

bilgisayar kaynağı ile birkaç saatte işlenebiliyorken hizmet düzeyi sözleşmesinde, üretilen verinin gerçek zaman veya gerçek zamana yakın bir sürede işlenmesi istenebilir. Bu ve benzer durumlarda hizmet düzeyi sözleşmesini sağlamak için bir bilgisayar kaynağından daha fazla kaynağa ihtiyaç duyulur.

Hizmet düzeyi sözleşmesi aslında büyük verinin V'leri olarak ifade edilen özelliklerin belirlenmesiyle yakından alakalıdır. Büyük veri özelliklerinin İngilizce "V" harfiyle başlayan kelimeler ile ifade edilmesinden dolayı büyük verinin V'leri olarak da anılan birkaç temel özelliği bulunmaktadır. Başlangıçta büyük verinin temel özellikleri olarak hacim (volume), çeşitlilik (variety), hız (velocity) özellikleri sayılıyorken [37, 38] daha sonrasında gerçeklik (veracity) [39], değer (value) özellikleri de eklenmiştir [40].

Hacim (volume), verinin hacminin büyük olmasını ifade etmektedir. Büyük hacimli verileri geleneksel yöntemler ile depolamak ve işlemek zordur. Bunun için büyük veri teknikleri kullanılır.

Çeşitlilik (variety), üretilen verinin birçok farklı form veya formatta olmasını ifade eder. Üretilen verilerin farklı form ve formatlarda olması verinin yapısal olarak depolanması ve işlenmesinde önemli problemler oluşturur.

Hız (velocity), verinin çok hızlı üretilmesini ifade eder. Devasa büyüklüklerde olmasa da çok hızlı üretilen verileri gerçek zamana yakın bir zamanda işlemek yine geleneksel yöntemler ile zordur. Bu özellikteki veriler büyük veri teknikleri kullanarak işlenebilir.

Dağıtık veya paralel hesapla yöntemleri kullanılarak daha kısa sürede daha fazla veri işlenebilir, işleme ve okuma süreleri hizmet düzeyi sözleşmesine uygun hale getirilebilir [2].

Görüntüleme araçlarının gelişmesi ve yaygınlaşması sonucu üretilen görüntü ve video verilerinin de hacmi ve çeşitliği artmıştır. Görüntü ve video gibi verilerin boyutlarının büyük olması hem depolama kaynağı hem de hesaplama kaynağına olan ihtiyacı daha da artırmıştır. Bu ihtiyaçları klasik yöntemler ile hizmet düzeyi sözleşmesini sağlayacak şekilde karşılamak daha zordur. Yüksek özellikli donanımlara sahip bilgisayarlar ile bu ihtiyacın karşılanması maliyetli bir çözümdür. Bundan dolayı sıradan bilgisayardan oluşan bilgisayar kümesi ile dağıtık bir sistem kurularak daha az maliyetle bu ihtiyaçları karşılama fikri ön plana çıkmıştır.

Bir bilgisayar kaynağı ile hizmet düzeyi sözleşmesinin karşılanmadığı durumlarda paralel ve dağıtık hesaplama tekniklerinin kullanımı ön plana çıkmaktadır. Dağıtık veya paralel hesaplama yöntemleri kullanılarak daha kısa sürede daha fazla veri işlenebilir, işleme ve okuma süreleri hizmet düzeyi sözleşmesine uygun hale getirilebilir. Bu durum büyük veri

ile paralel ve dağıtık sistem kavramlarının sık olarak beraber anılmasını ve kullanılmasını sağlamıştır. Büyük veri tekniklerinin ortaya çıkmasındaki temel fikir de verilerin çok hızlı olarak işlenebilmesini sağlamaktır.

Büyük veri teknikleri birçok alanda kullanılmaktadır. Petrol arama firmalarının terabaytlarca coğrafik verisinin analizinde, borsalarda dakikada üretilen milyonlarca işlem verisinin analizinde, market zincirlerinin müşteri verilerinin analizinde, sigorta şirketlerinin risk analizlerinde ve daha birçok alanda büyük veri tekniklerini kullanılmaktadır [39, 41, 42].

Büyük veri tekniklerinin ortak özellikleri aşağıda verilmiştir [2]:

Verinin Birden Fazla Düğüm Üzerine Dağıtılması: Büyük verinin işlenmesi ve analiz edilmesi için bir bilgisayarın kaynağı yeterli gelmemektedir. Büyük verinin işlenmesinde önemli avantajlardan biri genelde ortalama birkaç TB yerel depolama alanına sahip, sıradan, pahalı olmayan bilgisayarların kullanılmasıdır. Büyük veri genelde bir bilgisayarın yerel disk alanından daha büyük verilerden oluşur. Bu yüzden verinin birden fazla düğüm üzerine dağıtılması gerekmektedir. Veri bir bilgisayarın yerel diskine sığabilecek büyüklükte olsa bile ağ üzerinden verinin okunması yerel disklerden verinin okunmasına göre daha yavaş olduğundan bu veri düğümlere dağıtılır. Bu şekilde, veri farklı düğümler üzerinde yedeklenerek, herhangi bir düğümün arızalanması durumunda veri kaybı engellenmiş olur.

Uygulamaların Verinin Bulunduğu Düğümlere Taşınması: Büyük veri tekniklerinde veri, uygulamanın olduğu konuma getirilip orada işlenmez. Terabaytlarca verinin düğümler arasında taşınması yerine mümkünse verinin bulunduğu düğümlere uygulamalar taşınır ve veri orada işlenir. Uygulamalar düğümlere kopyalanırken uygulamanın bağımlı olduğu kütüphane ve paketlerin de taşınması gerekmektedir. Büyük veri sistemleri bu işlemlere job (iş) denir. Uygulama usta (master) düğüme kurulur, joblar bu uygulamanın diğer düğümlere taşınması işlemini yönetir.

Verilerin Rastgele Okunması (Random Reads) Yerine Ardışık Olarak Okunması: Hard disklerden veri okunurken rastgele erişim yöntemi kullanılır. Diskten bir veri okunmak istendiğinde, ilk olarak okuma kafası verinin olduğu yeri arar ve bulur. Bu işlem arama (seek) işlemi olarak adlandırılır. Bu işlem için belli bir süre harcanır. Verinin konumu bulunduktan sonra veri okunmaya başlanır. Bunun için de belli bir süre harcanır. Okuma işleminde arama zamanını azaltmak için, büyük veri sistemlerinde rastgele erişim yöntemleri yerine ardışık olarak okuma yöntemi tercih edilmektedir. Ardışık aramada verinin

bulunduğu konumu bulmaya harcanan toplam süre rastgele erişim yöntemine göre daha azdır, dolayısı ile ardışık okuma rastgele erişim yöntemi ile okumaya göre daha hızlıdır [43].

2.5. Büyük Veri Teknolojileri

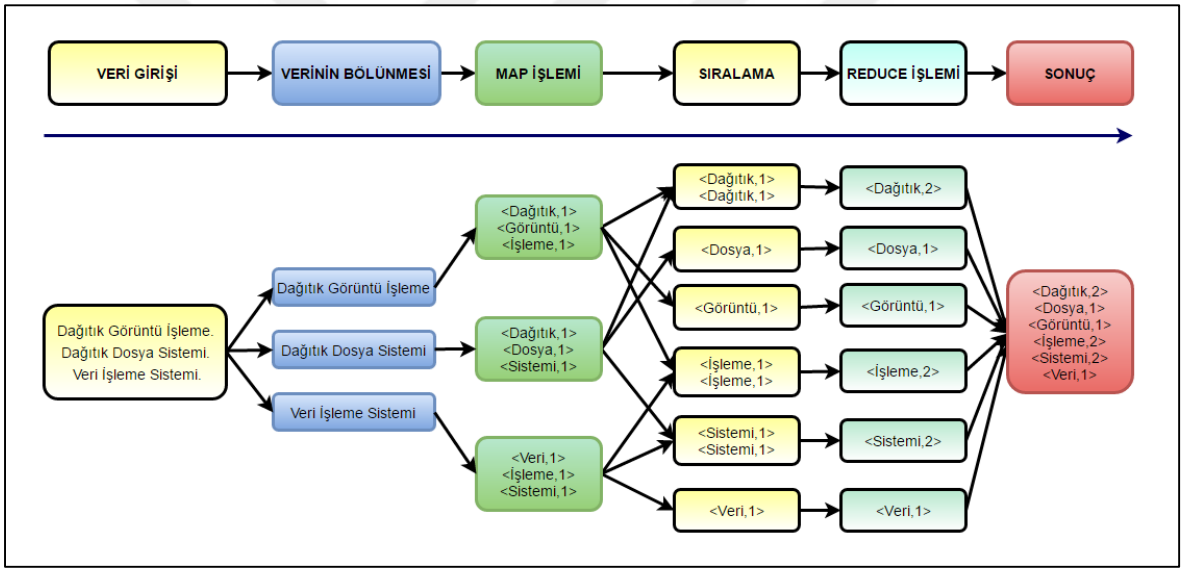
Veri üretim hızının, hacminin ve değerinin artması büyük veri kavramın ortaya çıkmasını sağlamıştır. Büyük verinin ortaya çıkması ile beraber bu verinin depolanması, işlenmesi ve analizi, bu veriden bilgi ve değer elde edilmesi gibi işlemler önem kazanmış ve bu işlemler için yeni teknolojilerin bulunması, geliştirilmesi ve etkin olarak kullanılması ile ilgili çalışmalar yapılmıştır. Çünkü ham veri işlenip bilgi elde edildiği oranda katma değer sağlar. Büyük verinin işlenmesi ve bu veriden katma değer elde etmek geleneksel yöntemler ile zor olduğundan büyük veri teknolojileri diye adlandırılan farklı yöntem ve teknolojiler geliştirilmiştir. Sadece bilim adamları değil büyük veriye sahip ve bu veriyle uğraşma durumunda olan Google, Yahoo, Facebook gibi firmalar da bu alanda çalışmalar gerçekleştirmişlerdir. Bu çalışmalardan biri olan MapReduce [44] 2004 yılında Google tarafından duyurulmuştur. MapReduce büyük verinin parçalara ayrılarak dağıtık olarak işlenmesini kolaylaştırmıştır. Büyük verinin depolanması problemine çözüm olarak da dağıtık dosya sistemleri kullanılmaktadır. Google bunun için Google Dosya Sistemi (GFS) adında ölçeklenebilir ve kontrol edilebilir bir dağıtık dosya sistemi geliştirmiştir [45]. Daha sonra 2004 yılında Google'ın yayınladığı MapReduce mimarisine dayanan açık kaynak kodlu Hadoop [46] kütüphanesi geliştirilmiştir. Hadoop kütüphanesi Hadoop Distributed File System (HDFS) adında bir dağıtık dosya sistemi ve MapReduce tekniği bileşenlerine sahiptir.

2.5.1. MapReduce

MapReduce büyük veri kümeleri oluşturmayı ve işlemeyi sağlayan bir programlama modelidir [44]. MapReduce programlama modelinin temelinde kullanıcı tanımlı map ve reduce adı verilen iki tane fonksiyon bulunmaktadır. Map fonksiyonu veriyi <anahtar, değer> (<key,value>) çifti olarak alır, işler ve ara <anahtar, değer> çiftlerini oluşturur. Reduce fonksiyonu ise map fonksiyonu tarafından üretilen bu ara <anahtar, değer> çiftlerini alır ve aynı anahtar bilgisine sahip değerlerin gruplama işlemini gerçekleştir [47]. Map ve reduce fonksiyonları tarafından kullanılan anahtar, değer parametreleri temel veri tipleri olabileceği gibi kullanıcı tarafından tanımlanan özel veri tipleri de olabilir.

MapReduce sayesinde büyük problemler daha küçük parçalara bölünerek dağıtık olarak çözümlenebilmektedir. MapReduce programlama modeli metin madenciliği, metin işleme, doğal dil işleme, makine öğrenmesi, görüntü işleme [48] gibi birçok alanda büyük veri problemlerinin çözümü için kullanılabilmektedir. MapReduce sayesinde işler sıradan bilgisayarlardan oluşan bilgisayar kümelerine dağıtılarak otomatik olarak paralelleştirme sağlanır.

Şekil 2.6' da bir MapReduce programı ile gerçekleştirilen örnek bir kelime sayma uygulamasının aşamaları gösterilmiştir. Şekilde görüldüğü gibi map işlemi ile kelimeler anahtar olarak, kelime sayısını ifade eden 1 sayısı ise değer olarak alınmış, <anahtar,değer> çifti oluşturulmuştur. Reduce işleminde ise aynı anahtar bilgisine sahip <anahtar,değer> çiftleri birleştirilmiştir.



Şekil 2.6. Kelime sayma uygulamasının MapReduce aşamaları

MapReduce tekniği ile işler daha küçük parçalara bölünerek paralel olarak işlenir. Yapılan işlem sonucunda elde edilen sonuçlar reduce aşamasında birleştirilir.

Bu tezde MapReduce tekniği kullanılmıştır. Görüntüler ve videolar MapReduce tekniği ile paralel olarak işlenmiştir. Bu bölümde MapReduce tekniği genel olarak tanıtılmıştır. Yapılan uygulama tezin sonraki bölümlerinde ayrıntılı olarak anlatılmıştır.

2.5.2. Dağıtık Dosya Sistemleri

Büyük verinin yönetiminde var olan genel dosya sistemlerinin yetersiz kalmasından dolayı dağıtık dosya sistemleri ön plana çıkmıştır. Dağıtık dosya sistemi, uzak dosya oluşturma, bu dosyalara erişme, okuma ve yazma işlemlerini yerel dosyalarda olduğu gibi sağlayan, kullanıcılara ağdaki herhangi bir bilgisayardan bu dosyalara erişmeye izin veren sistemdir [49].

Dağıtık dosya sistemlerinin ortaya çıkmasındaki temel fikir bilgisayar depolama kaynaklarının ortak olarak kullanılmasını sağlamaktır. Bunun yanında dağıtık dosya sisteminin hata toleranslı olarak çalışması, yedekleme işlemini sağlaması da bu dosya sistemlerini ön plana çıkarmıştır. Dağıtık dosya sistemleri, dağıtık sistemler ile beraber büyük veri problemlerinin çözümünde önemli bir yere sahiptir. Günümüzde kullanılan farklı Dağıtık dosya sistemleri bulunmaktadır. Sun Ağ Dosya Sistem (NFS) [50], Andrew Dosya Sistemi (AFS) [51], Google Dosya Sistemi (GFS) [45] , Hadoop Dağıtık Dosya Sistemi (HDFS) [52] bunlardan bazılarıdır. Bu tezde Google Dosya Sistemi (GFS) ve Hadoop Dağıtık Dosya Sistemi (HDFS) ileriki bölümlerde anlatılmıştır. Dağıtık dosya sisteminin aşağıda listelenen özellikleri sağlaması gerekmektedir [49, 53];

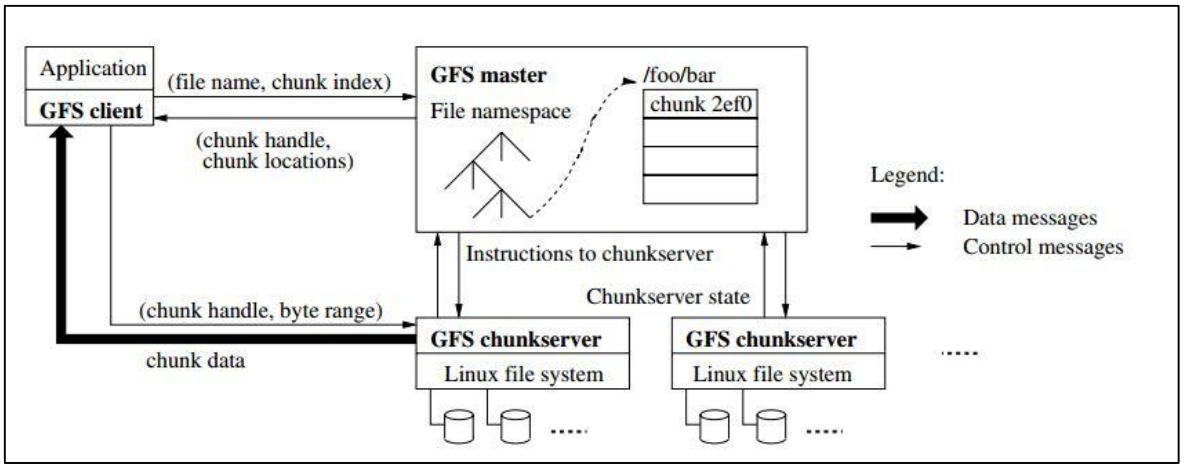
- **Saydamlık:** Bir dağıtık dosya sistemi, erişim, konum, eşzamanlılık, performans, ölçeklenebilirlik, taşınabilirlik gibi farklı seviyelerde saydamlığı sağlamalıdır [54]. Sağladığı ara yüz ve komut kümesi ile yerel ve uzak dosyalara erişimde bir farklılığın olmaması gerekmektedir. Yerel dosya sistemi için yazılan programlar uzak dosya sistemleri için de çalıştırılabilir olmalıdır. Depolama kaynaklarının ve dosyaların fiziksel konumları kullanıcı ve uygulamalardan gizlenmelidir, tek bir adres uzayı gösterilmelidir. Eş zamanlı olarak ortak bir veriye erişimde kullanıcı ve uygulamalar bir birinden etkilenmemelidir. Dağıtık sistemin performansının değişkenlik göstermesinden de kullanıcı ve uygulamalar etkilenmemelidir.
- **Yedekleme:** Verinin, dağıtık dosya sisteminde performans ve güvenirlik için yedeklenmesi gerekmektedir. Yedekleme sayesinde verinin bozulması durumunda daha önce oluşturulan yedekler kullanılarak sistem çalışması sürdürülür. Ayrıca veri farklı konumlara yedeklenerek veri yerelliği sağlanır. Veri yerelliği performans artışını sağlar.
- **Tutarlılık:** Verinin farklı yedeklerinin oluşturulması, tutarlılık probleminin ortaya çıkmasına neden olmaktadır. Verinin bir yedeğinde yapılan bir değişikliklerin diğer yedekler için de yapılması ve tutarlılığın sağlanması gerekmektedir.

- **Donanım ve İşletim Sistemi Farklılığı:** Farklı bilgisayar ve işletim sistemleri için yazılımların geliştirilebilmesi için bir ara yüz tanımlanması gerekmektedir.
- **Hata Toleransı:** Dağıtık dosya sistemi hata toleransını sağlaması gerekmektedir. Herhangi bir istemci veya sunucu hatası oluştuğunda sistem çalışmasını sürdürmelidir.
- **Güvenlik:** Dosya sistemi güvenli olmadır. Güvenlik için kimlik doğrulama, dijital imza ve şifreleme mekanizmaları kullanılarak yetkilendirme sağlanmalıdır. Yetkisiz erişimlere izin verilmemelidir.
- **Verimlilik:** Dağıtık dosya sistemi en az geleneksel dosya sistemi kadar performanslı olmalıdır.

2.5.2.1. Google Dosya Sistemi

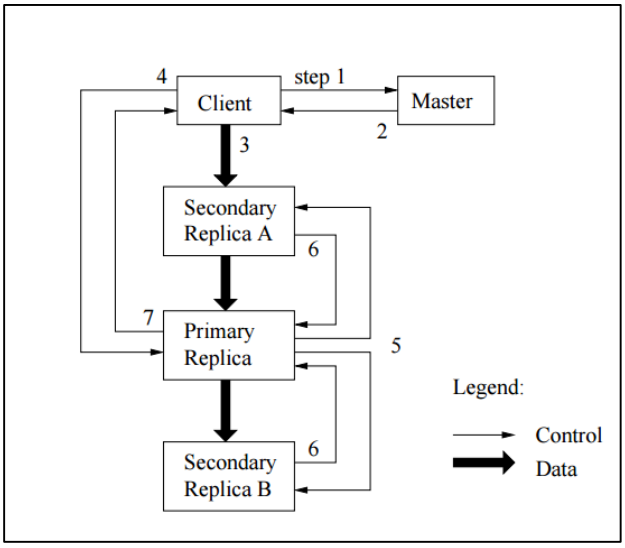
Google'ın kendi ihtiyaçları için geliştirdiği Google Dosya Sistemi (GFS) büyük, dağıtık, veri yoğun uygulamalar için geliştirilen ölçeklenebilir dağıtık bir dosya sistemidir [45]. GFS Google'ın artan veri işleme ihtiyacı göz önünde bulundurularak tasarlanmıştır. GFS yüzlerce veya binlerce sıradan bilgisayar üzerinde çalışabilir. GFS, ölçeklenebilir, performanslı, güvenilir ve kolay kullanılabilir bir dosya sistemidir. GFS kümesinin mimarisi Şekil 2.7'de gösterilmiştir.

Bir GFS kümesi 1 usta (master) sunucu, birden fazla da yığın (chunk) sunucudan oluşmaktadır. GFS kümesine birden fazla istemci bağlanabilmektedir. Ana sunucu sistemin bütün meta verisinin yönetir. Bu meta veri içerisinde, dosya adları, dosya yolları, dosyaların erişim kontrolleri, dosyaların fiziksel olarak saklandığı yığın sunucu bilgileri ve yığın sunucu ile ilgili meta veriler gibi bilgiler bulunmaktadır. Ana sunucu aynı zamanda yığın sunucuların yönetiminden de sorumlu olup yığın sunucuların durumlarını "Kalp Atışı" (HeartBeat) adını verilen mesajlarla kontrol eder. Verinin kendisi yığın sunucularda tutulur. Bu veriler varsayılan olarak 64 MB boyutunda yığın (chunk) adı verilen veri blokları şeklinde tutulmaktadır. Veri yığınların yığın sunucular üzerinde 3 adet yedeği oluşturulur. Ana sunucu yığın sunucular üzerinde verinin yedeğinin oluşturulmasını yönetir ve bu bilgileri tutar. Bu yedekler herhangi bir hata oluşması durumunda sistemin çalışmasının sürdürülmesi için önemlidir [45].



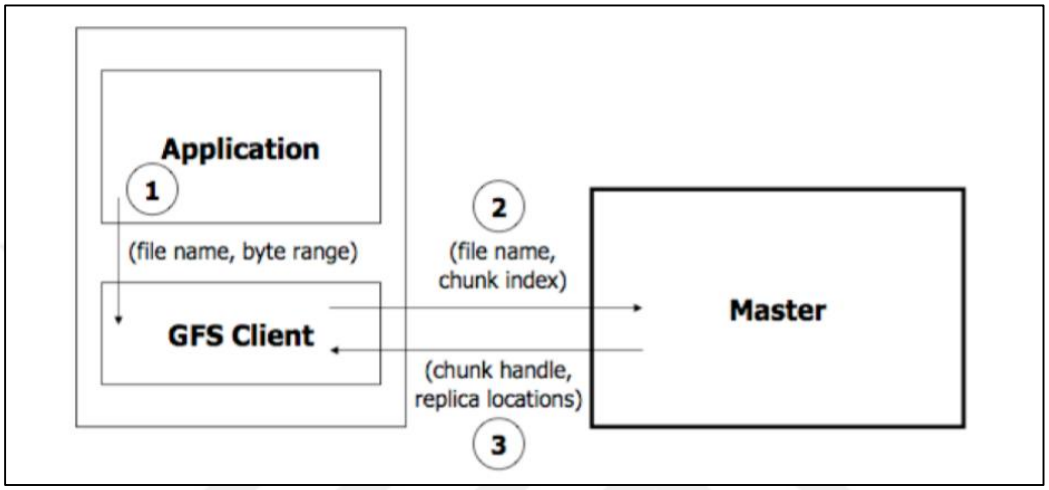
Şekil 2.7. Google Dosya Sistemi (GFS) mimarisi [45].

GFS veri yazma işlemi Şekil 2.8' de gösterilmiştir. Bir veri yazma işleminde istemci en yakın yığın sunucu ve onun yedeklerinin tutulduğu yığın sunucuların yerini sorar. Usta sunucu en yakın sunucunun adresini ve onun yedeklerinin tutulduğu yığın sunucu adreslerini istemciye iletir. İstemci usta sunucunun cevabında yer alan yığın sunuculara veriyi gönderme isteğinde bulunur. Bütün yığın sunucular kabul ettikten sonra veri ilk olarak birincil yığın sunucusuna yazılır. Birinci yığın sunucu bütün ikincil yedek sunuculara veri yazma isteğinde bulunur ve veriyi yazar. İkincil yığın sunuculardaki veri yazma işlemi tamamlandığında işlemin tamamlandığına dair bilgi birincil yığın sunucuya bildirilir. Birincil yığın sunucu yazma işleminin bittiğini istemciye iletir [45].



Şekil 2.8. GSF yazma kontrolü ve veri akışı [45].

Şekil 2.9’ da gösterildiği gibi okuma işlemi için uygulamalar GFS istemciden bir okuma talebinde bulunur. İstemci bu isteği ana sunucuya gönderir. Ana sunucu verinin bulunduğu yığın sunucu ve yedek yığın sunucu yerlerini istemciye iletir. İstemci yığın sunuculardan birini seçer ve istekte bulunur. Yığın sunucu veriyi istemciye gönderir. İstemci alınan veriyi uygulamaya iletir [55].



Şekil 2.9. GFS veri okuma işlemi [55].

2.5.3. Hadoop

Google, 2004 yılında yayınladığı MapReduce [44] makalesi ile açık kaynak olarak Hadoop’un geliştirilmesine ilham kaynağı olmuştur. İlk başlarda Hadoop, Nutch adı verilen açık kaynak kodlu bir web arama motorunu desteklemek için geliştirilmiştir [56]. Daha sonra Hadoop, Nutch’dan ayrılarak Apache Foundation altında ayrı bir proje haline gelmiştir.

Hadoop, Apache tarafından geliştirilmekte olan güvenilir, ölçeklenebilir dağıtık hesaplama için geliştirilen açık kaynak bir projedir [46]. Hadoop, Java [57] programlama dili ile geliştirilmiştir. MapReduce ve Hadoop Dağıtık Dosya Sistemi (HDFS) adı verilen Dağıtık Dosya sisteminin açık kaynak olarak geliştirildiği bir kütüphanedir. Büyük verinin işlenmesinde önemli yeri olan Apache Hadoop projesi Facebook, IBM, Yahoo, Amazon, Ebay gibi birçok kuruluş tarafından farklı amaçlar için yoğun olarak kullanılmaktadır [58]. Hadoop az maliyetle büyük veri problemlerinin çözümüne imkân sağlamıştır. Hadoop’un bulut bilişim desteği olan açık kaynak dağıtık dosya sistemi Hadoop Dağıtık Dosya Sistemi (HDFS)’e sahip olması büyük veriyi işlemede önemli bir avantaj oluşturmuştur.

2.5.3.1. Hadoop'un Mimarisi

Hadoop projesinin geliştirilme sürecinde farklı Hadoop sürümleri yayınlanmıştır. Hadoop 1 sürümünden Hadoop 2 sürümüne geçildiğinde önemli değişiklikler yapılmıştır. Bu nedenle Hadoop 1 sürümleri ile Hadoop 2 sürümleri arasında önemli farklılıklar bulunmaktadır. Hadoop 1'de bulunan JobTracker ve NameNode gibi bileşenler daha büyük Hadoop kümelerinde bir dar boğaza neden olmuştur [2]. Hadoop 1 sürümünde var olan ölçeklendirme, performans ve verimlilik alanlarında oluşan darboğazları aşmak için Hadoop 2 geliştirilmiştir. Şekil 2.10'da Hadoop 1 ve Hadoop 2'nin bileşenleri verilmiştir. Bu bileşenlerin kısaca açıklamaları aşağıda verilmiştir.

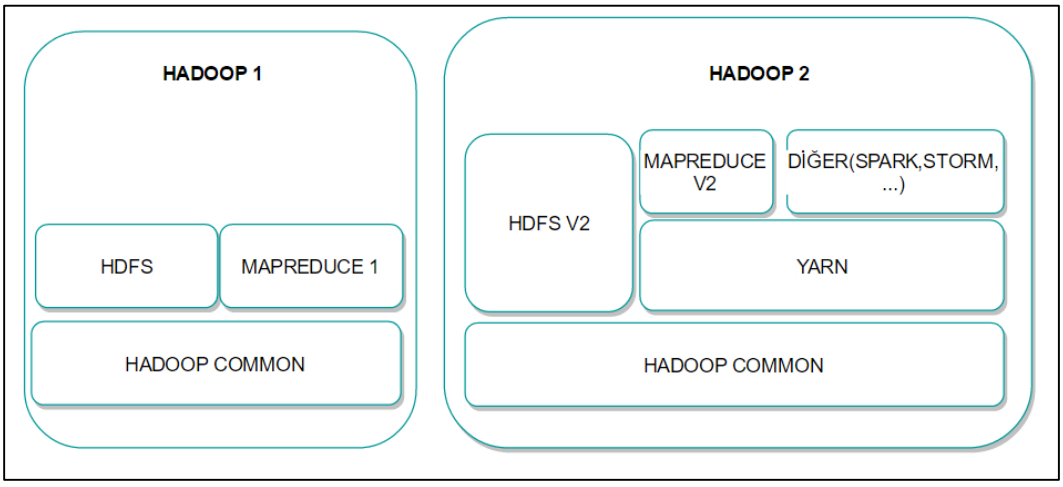
Hadoop Common: Diğer Hadoop modülleri için genel araçları içerir.

Hadoop Dağıtık Dosya Sistemi (HDFS) : Yüksek verimli, uygulama verilerinin tutulduğu ve erişildiği dağıtık dosya sistemidir

Hadoop YARN: İş planlamasını ve kaynak yönetimini sağlayan bir çatıdır.

Hadoop MapReduce: YARN tabanlı paralel büyük veri işleme sistemidir.

Aşağıda ilk olarak Hadoop 1 sürümünün bileşenleri tanıtılacaktır. Daha sonra bu tez çalışmasında kullanılan Hadoop Dağıtık Dosya Sistemi (HDFS) ve Hadoop 2 sürümü anlatılacaktır.



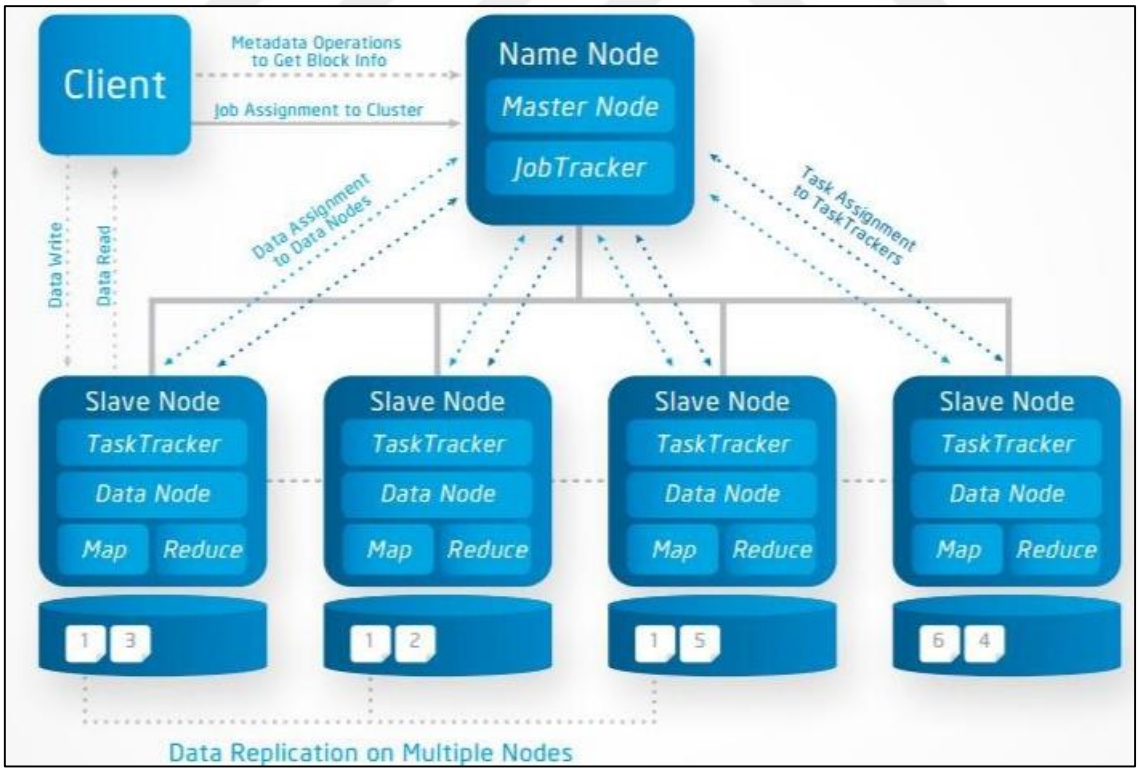
Şekil 2.10. Hadoop 1 ve Hadoop 2 bileşenleri

2.5.3.2. Hadoop 1

Hadoop 1 mimarisi NameNode, Secondary NameNode, Data Node, JobTracker, TaskTracker bileşenlerinden oluşmaktadır. Hadoop 1 sürümünün mimarisi Şekil 2.11'de gösterilmiştir. NameNode, Secondary NameNode, Data Node bileşenleri Hadoop Dağıtık Dosya Sistemi (HDFS) bileşenleri; JobTracker, TaskTracker bileşenleri de MapReduce bileşenleri olarak gruplandırılabilir.

Hadoop 1 kümelerinde usta (master) ve işçi (slave) olmak üzere iki tür düğüm bulunmaktadır. Usta düğümler NameNode, Secondary NameNode, JobTracker servislerini yürütür. İşçi düğümler DataNode ve TaskTracker servislerini yürütür.

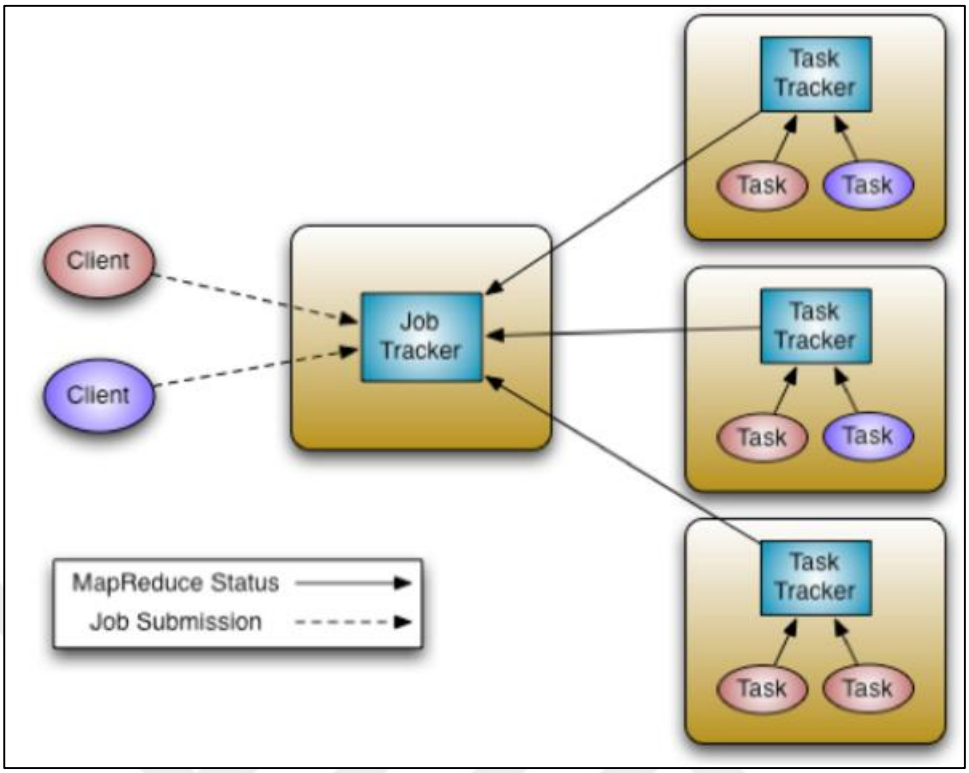
Bir kümede bir usta düğüm bulunur. Fakat birden fazla DataNode ve TaskTracker bulunabilir. Küçük ölçekli geliştirme ve test için oluşturulan kümelerde usta düğüm ve işçi düğüm genelde aynı bilgisayar üzerinde çalıştırılır. Ancak büyük ölçekli, üzerinde ürün çalıştırılacak kümelerde usta düğümler ile işçi düğümler ayrı düğüm olacak şekilde yapılandırılırlar.



Şekil 2.11. Hadop 1 Mimarisi [59].

- **NameNode:** Hadoop Dağıtık Dosya Sisteminde (HDFS) tutulan dosyaların meta verisini tutan bileşendir. Meta veriler, bloklarda bulunan dosyaların bilgisini ve DataNode üzerinde bu dosyaların bulunduğu yer bilgisini içerir [60].
- **Secondary NameNode:** Bu bileşen isminin çağrıştırdığı gibi Name Node bileşeninin bir yedeği değildir. Secondary NameNode dosya sisteminde kontrol noktaları oluşturarak sistemin düzgün çalışmasına yardımcı olur [61]. Küçük ölçekli kümeler için NameNode ile beraber aynı düğümde çalışacak şekilde yapılandırılabileceği gibi büyük ölçekli kümeler için ayrı düğümler üzerinde çalışacak şekilde de yapılandırılabilir.
- **DataNode:** HSFS üzerinde bulunan dosya bloklarının fiziksel olarak saklandığı bileşendir. DataNode' da veriler yerel disklerinde saklanır [56]. Dosya okuma ve yazma isteklerine cevap verir. Ayrıca veri blokların ve dosyaların yönetimi ile alakalı NameNode tarafından yapılan isteklere cevap verir.
- **JobTracker:** Ana bileşenlerden biridir, genel olarak işlerin (Job) yürütülmesini yönetmekle sorumludur [2]. Düğümlerin ve görevlerin düzgün çalışıp çalışmadığından sorumludur, başarısız olan görev ve düğümleri yeniden yapılandırır.
- **TaskTracker:** Bağımsız düğümlerde (Map ve Reduce) çalışır, map ve reduce görevlerini başlatmak ve yönetmekle sorumludur [2]. JobTracker ile ile haberleşir.

Hadoop 1 sürümü sadece MapReduce programlama modelini desteklemektedir. MapReduce programlama modeli Şekil 2.12' de gösterildiği gibi JobTracker ve TaskTracker tarafından yönetilir ve izlenir.

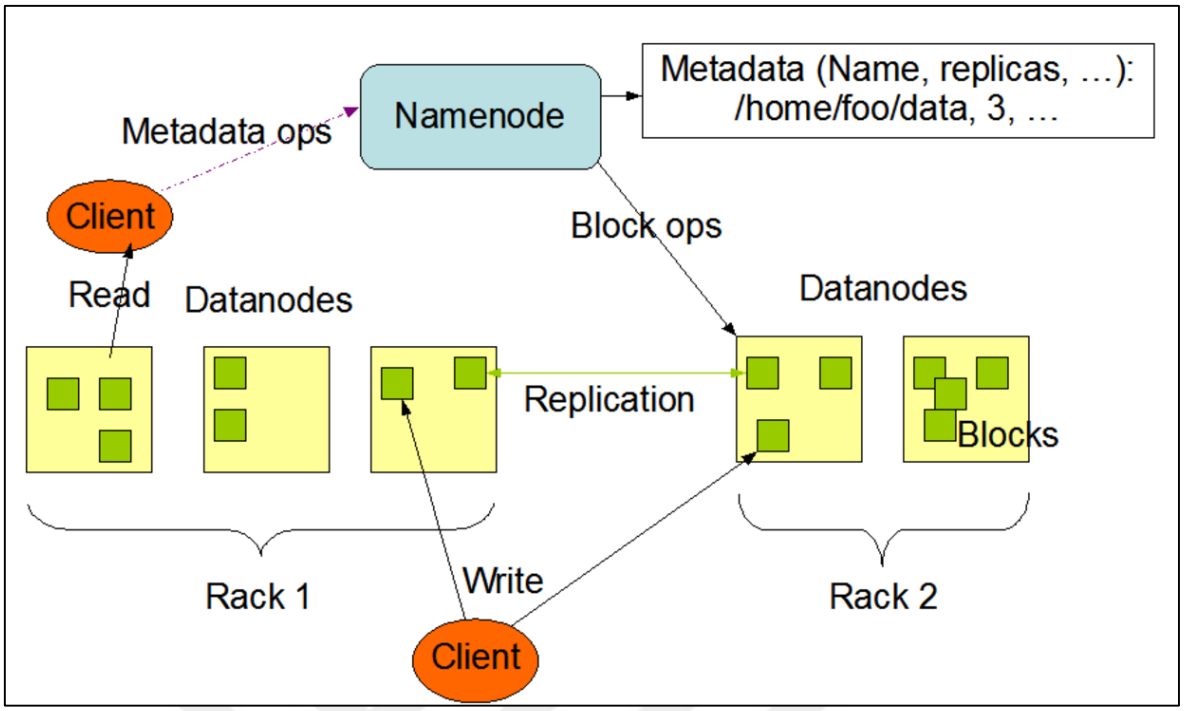


Şekil 2.12. Hadoop 1 sürümünde işlerin yürütülmesi [62].

2.5.3.3. Hadoop Dağıtık Dosya Sistemi (HDFS)

Büyük veriyi depolamak için bir bilgisayarın depolama kaynağı yeterli gelmediğinden, dağıtık dosya sistemleri kullanılmaya başlanmıştır. HDFS, Hadoop'un dağıtık dosya sistemi bileşenidir. Hadoop Dağıtık Dosya Sistemi (**HDFS**) [46, 63] uygulamaların büyük hacimli veriler ile çalışmasını sağlayan güvenilir, ölçeklenebilir dağıtık bir dosya sistemidir. Bu dağıtık dosya sistemi tasarlanırken verinin bir kere yazıldığı çok kere okunduğu durumlar göz önüne alınmıştır. Google Dosya Sistemi'nden (GFS) [45] esinlenerek geliştirilmiştir. HDFS, dağıtık halde bulunan depolama kaynaklarını bir araya getirerek kullanıcıya tek bir sistem gibi sunar. Bu sayede çok büyük hacimli verilerin depolanmasına imkan sağlar.

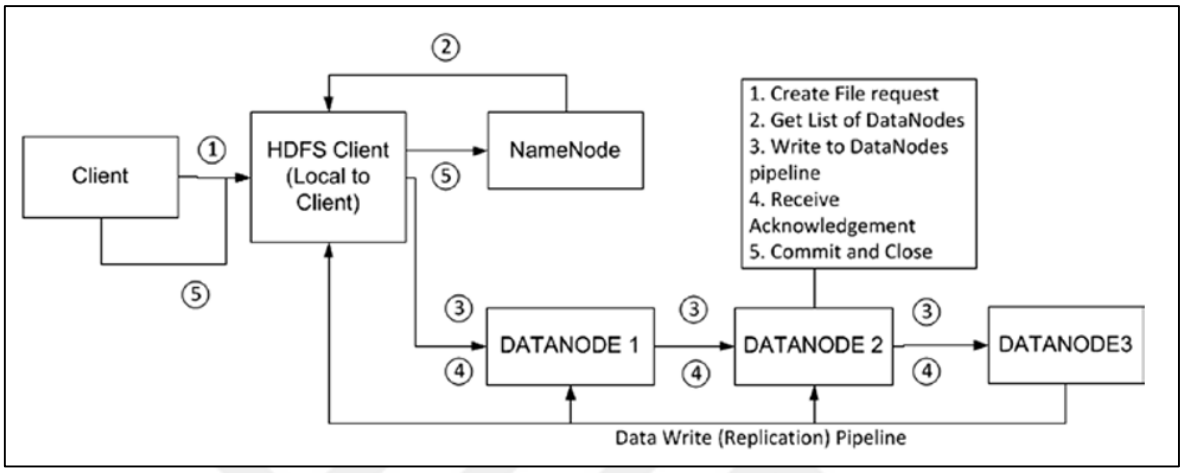
HDFS usta (master), işçi (slave) mimarisine göre çalışmaktadır. HDFS bileşenlerinden olan Name Node usta düğümde çalışır. Data Node'lar ise işçi düğümlerde çalışmaktadır. Şekil 2.13'de HDFS mimarisinde gösterildiği gibi Name Node dosya ve dizinlerin isim uzaylarını tutar. Ayrıca Name Node dosyalara ait veri bloklarının hangi Data Node düğümünde tutulduğunu bilir. Fakat dosyalara ait veri bloklarının yerini kalıcı olarak tutmaz, çünkü sistem başladığında Data Node'larda bu bilgiler yeniden yapılandırılır [64]. Data Node'lar ise dosyaların fiziksel olarak depolanmasını sağlar.



Şekil 2.13. HDFS mimarisi [65].

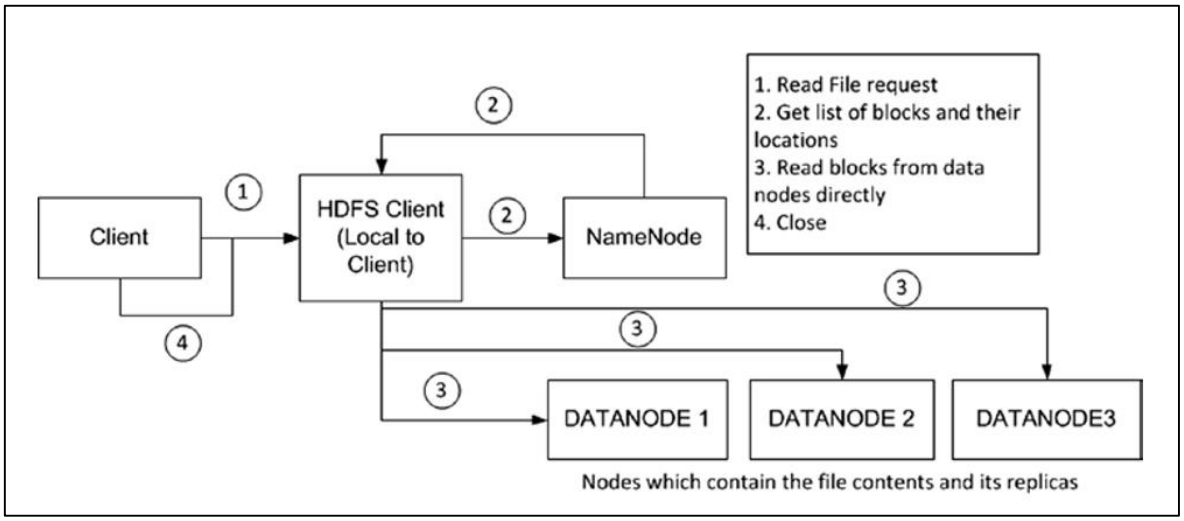
Veri bloklarının sunucu arızası sonucunda kaybolmaması için 3 farklı kopyası oluşturulur. Bu kopyalardan en az biri farklı sunucuda olmak üzere ayrı düğümlerde saklanır [66]. Verilerin kopyalarının oluşturulması aynı zamanda veri yerelliğini de sağlar. Veri yerelliği ise verilerin işlenmesi ve analiz edilmesinde daha iyi bir performans elde edilmesini sağlar. Varsayılan olarak dosya bloklarının boyutu 64 MB (Hadoop 1) veya 128 MB'dır (Hadoop 2). Dosya blok boyutu varsayılan değerlerden farklı bir değer olarak da ayarlanabilir. Her bir dosya için blok boyutu ayrı farklı değerlerde ayarlanabilir. Dosya boyutunun blok boyutundan büyük olması durumunda birden fazla bloğa yazılır. Dosya boyutu blok boyutunun tam katı olmadığı veya blok boyutundan küçük olduğu durumlarda bloklarda verinin yazılmadığı alan boşa gitmez ve blok boyutu boş alan kadar küçültülür [2]. HDFS büyük boyutlu dosyaların saklanması amacıyla tasarlandığı için çok sayıda küçük boyutlu dosyaların saklanmasında performans kaybı yaşanabilir [60, 67, 68]. Çünkü her bir dosya için ayrı ayrı meta veri tutulmaktadır. Dosya boyutunun çok küçük olması tutulan meta veri boyutunu artırmaktadır. Dosyalar okunurken her dosya için oluşturulan meta verilerinin ayrı ayrı okunması gerekmektedir. HDFS ortamında varsayılan dosya boyutundan daha büyük dosyalar saklanacaksa blok boyutunun artırılması daha az meta veri tutulmasını sağlayacaktır. Blok boyutu ne çok büyük ne de çok küçük seçilmelidir.

HDFS ortamına dosya yazma işlemi Şekil 2.14'de gösterilmiştir [2]. İstemci (client) HDFS ortamına bir dosya yazmak isteğinde, Name Node'dan dosyanın yazılacağı Data Node'ların listesini alır. Verinin yazılacağı bu düğümlere boru hattı (pipeline) mimarisi ile verileri yazar. Yazma işlemi bittiğinde Name Node'a yazma işleminin bittiği haber verilir.



Şekil 2.14. HDFS ortamına dosya yazma işlemi [2].

HDFS ortamından dosya okuma işlemi Şekil 2.15' de gösterilmiştir [2]. Okuma işlemi şu şekilde gerçekleştirilir. İstemci (client) dosya okuma isteği oluşturur. Name Node'dan dosya bloklarının bulunduğu Data Node konumu alınır. İstemci Data Node'a direk erişerek veriyi okur.



Şekil 2.15. HDFS ortamından dosya okuma [2].

2.5.3.4. Hadoop 2

Hadoop 1 toplu işler için MapReduce programlama modelinin yaygınlaşmasını sağlamıştır. Fakat ölçeklenme, kullanılabilirlik gibi bazı konularda Hadoop 1 mimarisindeki bazı kısıtlar, Hadoop 1 sürümünün yetersiz kalmasını ve Hadoop 2 sürümünün geliştirilmesini sağlamıştır. Hadoop 1 sürümü mimarisinden kaynaklanan kısıtlamalardan dolayı Hadoop 2'nin ana bileşenleri yeniden yazılmıştır. Hadoop 1 sürümünde MapReduce ile kısıtlı olan veri işleme Hadoop 2 ile daha geliştirilerek uzmanlaştırılmıştır. Hadoop 2'deki en önemli gelişme HDFS federasyonu ve YARN [69] kaynak yöneticisidir.

HDFS bir dağıtık dosya sistemidir. Bu dosya sisteminin iki ana bileşeni bulunmaktadır. Bunlar isim uzayları ve blok depolama servisleridir. İsim uzayı servisleri dosya ve dizinlerin oluşturulması, silinmesi, düzenlenmesi gibi işlemleri yönetir. NameNode bu işlemden sorumludur. Blok depolama servisleri ise DataNode düğümlerinde blokların depolanması ve yedeklerinin oluşturulması gibi blok işlemlerini yönetir.

Hadoop 1'de bir tane Name Node bütün Hadoop kümesi için bütün isim uzayını yönetmekteydi. NameNode'un düşmesi dosya ve dizin erişiminde problem yaşanmasına neden olmaktaydı. Hadoop 2 ile beraber birden fazla Name Node'un bu işlemi gerçekleştirebilmesi sağlanarak yatay ölçeklendirme ve performans iyileştirmesi sağlamıştır. Hadoop 2 ile beraber dosya sisteminin anlık görüntüleri alınabilir. Veri yedekleri oluşturularak bir felaketten dolayı veya yanlışlıkla silinen veya kaybolan verilere karşı önlem alınabilir. Bu yedekler tekrar yüklenerek veri kurtarılabilir.

Hadoop 2 sürümünde çalışan servisler iki gruba ayrılabilir. Bunlar Name Node, Secondary Name Node ve Data Nodes gibi HDFS 2 servisleri ve Resource Manager, Node Manager gibi MapRedue 2 (YARN) servislerdir.

HDFS 2 Servisleri

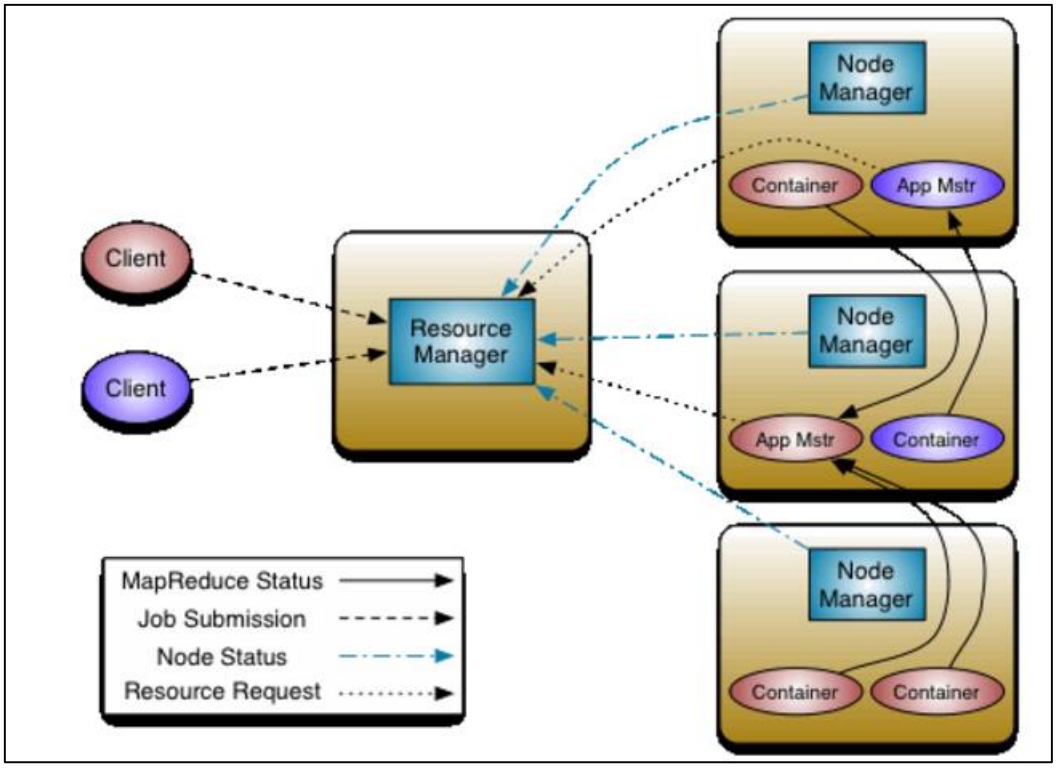
HDFS 2 servisleri HDFS 1 servislerinin bazı kısıtlamalarını ortadan kaldırmak için güncellenmiştir. HDFS 1 sürümünde bir tane Name Node ile bütün kümenin isim uzayı işlemleri gerçekleştirilmekteydi. Name Node'a erişilemediğinde bütün kümeye erişim sağlanamamaktaydı. Bu da sistemin sürekli kullanılabilir olmasında problem oluşturuyordu. HDFS 2 ile yüksek kullanılabilirliği sağlamak için aktif Name Node'un yanında beklemede olan StandBy Name Node adında ikinci bir Name Node kullanılmaktadır. Aktif Name Node düğümü erişilemez olduğunda beklemede olan StandBy Name Node devreye girerek işlemleri gerçekleştirir. Bu yedekli sistem sayesinde yüksek kullanılabilirlik sağlanmış olur.

HDFS 2 yüksek kullanılabilirlik modunda çalıştırılmadığında HDFS 1'de bulunan Name Node ve Secondary Name Node servisleri gibi çalışır.

MapRedue 2 (YARN)

YARN (Yet Another Resource Negotiator) [69] bir kaynak yöneticisidir. Hadoop 1'de veri işleme ve kaynak yönetimi işlemlerinden JobTracker sorumlu iken Hadoop 2 ile beraber veri işleme ve kaynak yönetimi ayrı birer servis haline gelmiştir. YARN kaynak yönetimi ve sorumluğu bakımından işletim sitemlerine benzediğinden Hadoop'un işletim sistemi olarak da adlandırılmaktadır. YARN çoklu kiracılı sistemlerde birden çok uygulamanın yürütülmesini sağlayacak şekilde geliştirilmiştir. Bu yapı ile MapReduce programlama modelinin yanında başka veri işleme modellerinin de kullanılabilirliği sağlanmıştır. YARN, Dryad [70], REEF [71] , Spark [72], Storm [73] ve Tez [74] gibi farklı programlama çatılarıyla kullanılabilmektedir [69]. YARN sayesinde kaynak yönetimi daha etkili bir şekilde sağlanmaktadır. YARN ile uygulamalar için özel kaynak yönetimi yapılabilmektedir. Uygulamanın önemine göre daha fazla veya daha az kaynak ayrılabilmektedir.

Şekil 2.16'da gösterilen Hadoop mimarisinin bileşenleri olan Resource Manager ile Node Manager, Hadoop 2 hesaplama çatısını oluşturmaktadır [75]. Hadoop 2 ile beraber gelen Resource Manager (RM) servisi kümenin kaynak yönetimini sağlar.



Şekil 2.16. Hadoop 2'nin mimarisi [75].

Scheduler ve ApplicationsManager, Resource Manager'in iki ana bileşenidir [75]. Application Master ise Resource Manager ve Node Manager ile beraber çalışarak uygulamaların yönetimini sağlar.

Scheduler çalışan uygulamalara kaynak tahsis etmekle sorumludur. Uygulamanın durumunu izleme ve takip etme işlemi Scheduler'in sorumluluğunda değildir. Scheduler uygulamaların CPU, hafıza, ağ gibi kaynak ihtiyacını konteynıra tahsis eder.

ApplicationsManager ise gelen işleri kabul eder. Uygulamaya özel olarak oluşturulan ApplicationMaster'ı oluşturmak ve çalıştırmak için ilk konteynır ile haberleşir. ApplicationMaster'in düşmesi durumunda ApplicationMaster'in çalıştığı konteynırı yeniden başlatır.

3. GÖRÜNTÜ İŞLEME VE DAĞITIK GÖRÜNTÜ İŞLEME

Teknolojik gelişmelerin en önemli çıktılarından biri üretilen veridir. Üretilen veri içerisinde video, görüntü gibi çoklu medya verilerinin miktarı giderek artmaktadır. Fotoğraf makineleri, video kayıt cihazlarının gelişmesi, kamera özelliğine sahip mobil cihazların yaygınlaşması gibi birçok etken çoklu medya verilerinin hızlı bir şekilde üretilmesinde ayrı bir ivme kazandırmıştır. Ayrıca güvenlik, sağlık, uydu teknolojileri, savunma sanayisi, endüstriyel uygulamalar, coğrafi bilgi sistemleri gibi birçok alanda da görüntü verilerin üretimi ve kullanımı da görüntü verilerinin miktarının ve boyutunun artmasına neden olmuştur. Hemen hemen her alanda üretilen görüntü, video gibi çoklu medya verilerinin artması bu verilerin depolanması, işlenmesi, analiz edilmesi için yeni tekniklerin geliştirilmesine neden olmuştur. Flickr, Instagram, Facebook, Youtube gibi sosyal paylaşım sitelerinin yaygınlaşması ile üretilen ve paylaşılan içerikler devasa boyutlara ulaşmıştır.

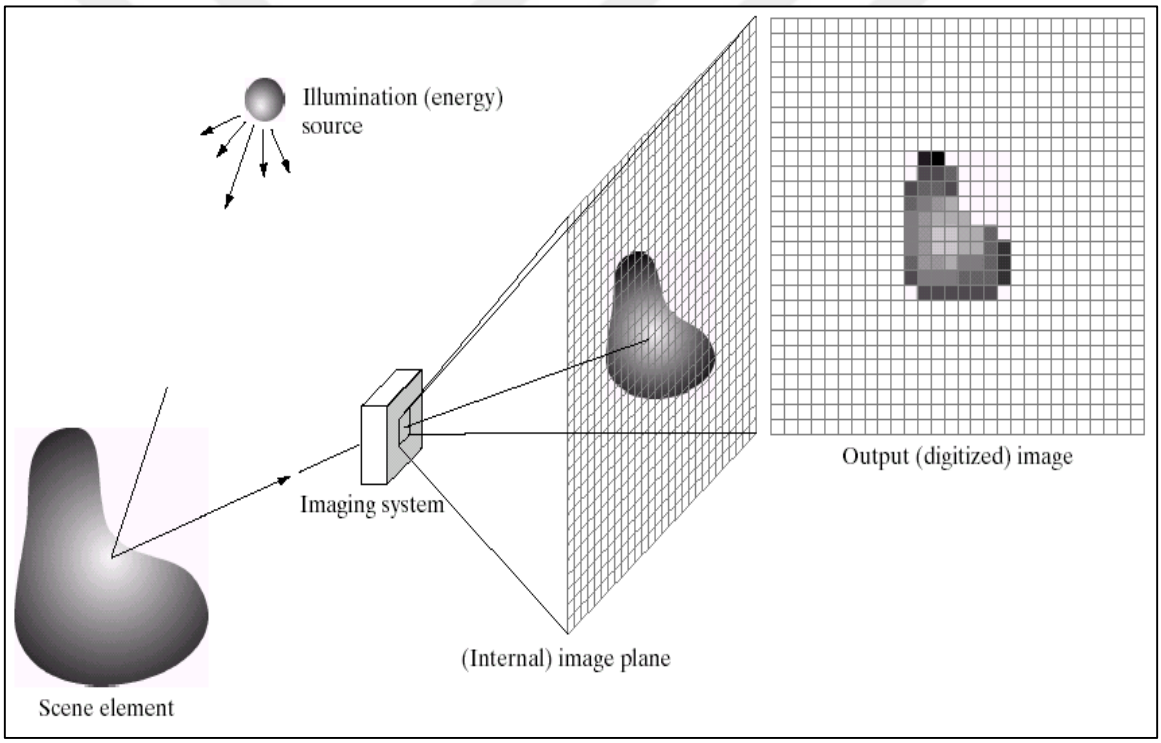
Hızlı bir şekilde üretilen ve devasa boyutlara ulaşan görüntü ve video verilerinin işlenmesi bir büyük veri problemidir. Bu tezde büyük veri problemlerinin çözümünde yoğun bir şekilde kullanılan Hadoop kütüphanesi ile görüntü ve video verilerinin dağıtık olarak işlenmesi konu alınmıştır. Görüntü işleme için geliştirilen birçok ücretli ve ücretsiz kütüphane bulunmaktadır. Bu kütüphaneler farklı diller ile geliştirilmiştir. Bu kütüphaneler görüntü işleme ile ilgili temel işlemleri ve hazır bazı algoritmaları içerir. Tezin bu bölümünde görüntü işleme için kullanılan kütüphane ve yöntemlerden bahsedilecektir.

3.1. Görüntü Oluşturma

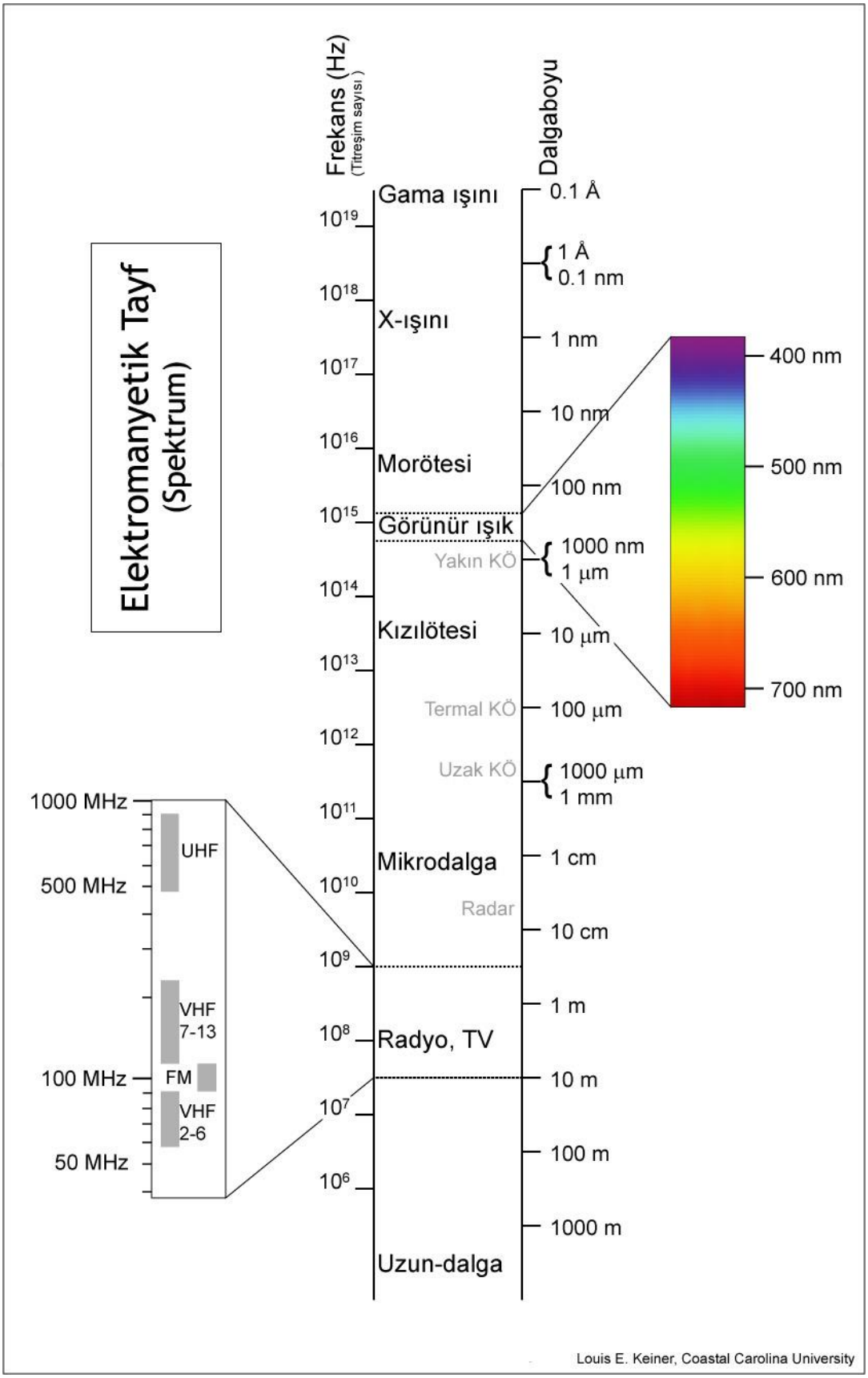
Görüntü iki boyutlu bir $f(x, y)$ fonksiyonu olarak tanımlanabilir [76]. Buradaki x, y görüntünün koordinat düzlemindeki yerini ifade eder. f fonksiyonun, (x, y) değerlerindeki genliği o noktadaki parlaklık veya renk gibi görüntüye ait bilgiyi ifade eder. Süreklilik özelliği gösteren f fonksiyonu ile analog görüntüler temsil edilebilir. f fonksiyonun ayrıklaştırılması, sonlu bir fonksiyon haline dönüştürülmesi analog görüntünün sayısal görüntüye dönüştürülmesine denk gelmektedir.

Şekil 3.1'de sayısal bir görüntünün nasıl elde edildiği gösterilmektedir. 3 boyutlu gerçek nesne uzayından yansıyan ışığın kamera gibi ışığa duyarlı cihazlarla 2 boyutlu görüntü uzayına aktarılmasıyla nesnelerin sayısal görüntüleri elde edilir [77].

Elektromanyetik tayfta Şekil 3.2'de gösterildiği gibi insan yaklaşık olarak 400 nm ile 700 nm arasında dalga boyuna sahip ışınımları algılayabilmektedir [78]. Bu aralıktaki dalga boyuna sahip ışınımlar görünür ışık olarak adlandırılmaktadır [79]. İnsan gözü görünür ışık altındaki nesneleri algılayabilmektedir. Fakat görünür ışık dışındaki ışınlar ile de görüntüleme yapılmaktadır. Örneğin, nükleer tıp ve astronomi gözlemlerinde gama, tıbbi tanıların yapılmasında ve endüstride X ray, mikroskobik canlıların görüntülenmesi için ultraviyole ışınları kullanılmaktadır [80]. Bu işlemler için CCD ve kızılötesi kameralar, X-ray, manyetik rezonans, ultrason cihazları gibi özel donanımlar kullanılmaktadır. Ayrıca gerçek dünyadan elde edilen görüntülerin yanında bilgisayar ortamında da görüntüler oluşturulabilir.

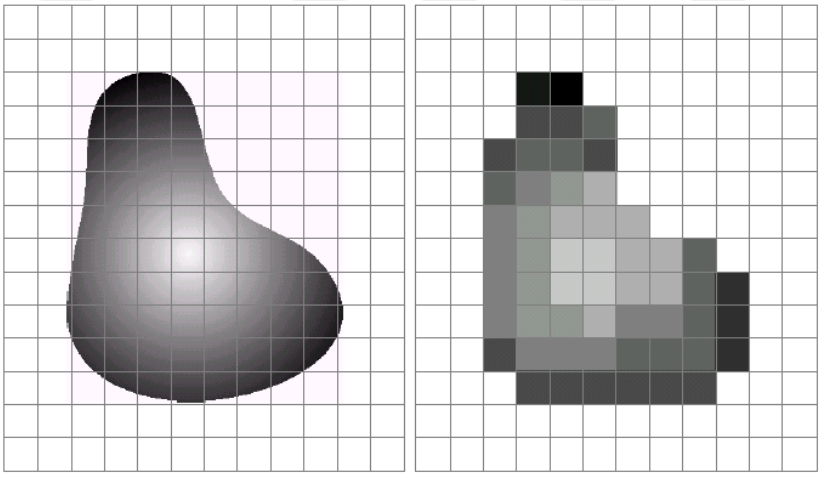


Şekil 3.1. Sayısal görüntünün elde edilmesi [76].



Şekil 3.2. Elektromanyetik tayf ve görünür ışık [79].

Bilgisayarlar ile görüntü işleme işlemlerinin uygulanabilmesi için analog görüntülerin örnekleme ve niceleme yapılarak sayısal görüntü haline dönüştürülmesi gerekir [81]. Analog görüntülerin hem koordinat olarak hem de genlik değerleri sürekli olabilmektedir. $f(x, y)$ sürekli fonksiyonu ile gösterilen analog görüntünün sürekli olan (x, y) koordinat değerlerinden örnekler alınarak sınırlı sayıda noktaların elde edilmesine örnekleme işlemi denir. $f(x, y)$ fonksiyonun (x, y) noktasındaki genlik değerlerinin sayısallaştırılmasına kuantalama (niceleme) denir [76]. Şekil 3.3'de analog görüntü bir görüntü ve bu görüntünün örnekleme ve niceleme işleminin sonucunda elde edilen sayısal görüntü gösterilmektedir.



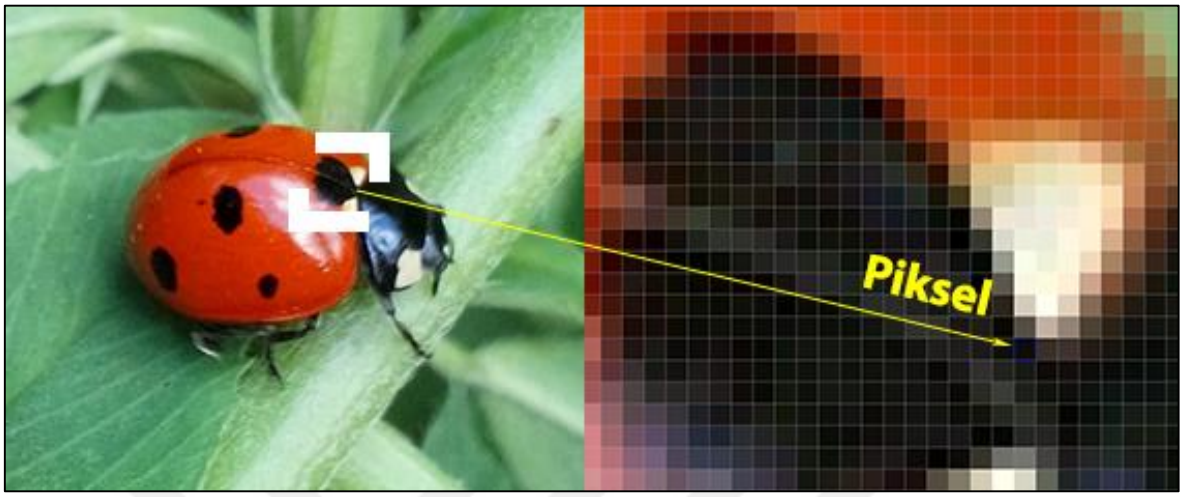
Şekil 3.3. Analog görüntünün (solda) örneklenme ve niceleme sonucu sayısallaştırma işlemi (sağda) [76].

3.2. Görüntünün Temsili

Sayısallaştırılmış görüntü ayrık ve sonlu $f[x, y]$ fonksiyonu olarak ve x= 0, 1, 2, 3, 4, …, M-1 ve y= 0, 1, 2, 3, 4, …, N-1 olacak şekilde düşünülürse, M satır N sütundan oluşan bir görüntü 2 boyutlu bir dizi ya da bir matris olarak tanımlanabilir [76]. Görüntülerin matris veya 2 boyutlu dizi olarak gösterimi yaygın bir yöntem olsa da bazen M x N elemana sahip 1 boyutlu dizi olarak da temsil edilebilir. Görüntü işleme için kullanılan kütüphanelerdeki uygulanma şekline göre bu tercih değişebilir.

Sayısal görüntülerin en küçük bileşeni yani matris veya dizinin her bir elemanı **piksel** olarak adlandırılır [82]. Her piksel ait olduğu görüntünün parlaklık ve renk bilgisine karşılık

gelen sayısal bir değer taşır [83]. Şekil 3.4'de bir görüntü büyütülerek görüntüyü oluşturan pikseller gösterilmiştir.



Şekil 3.4. Görüntüyü oluşturan pikseller

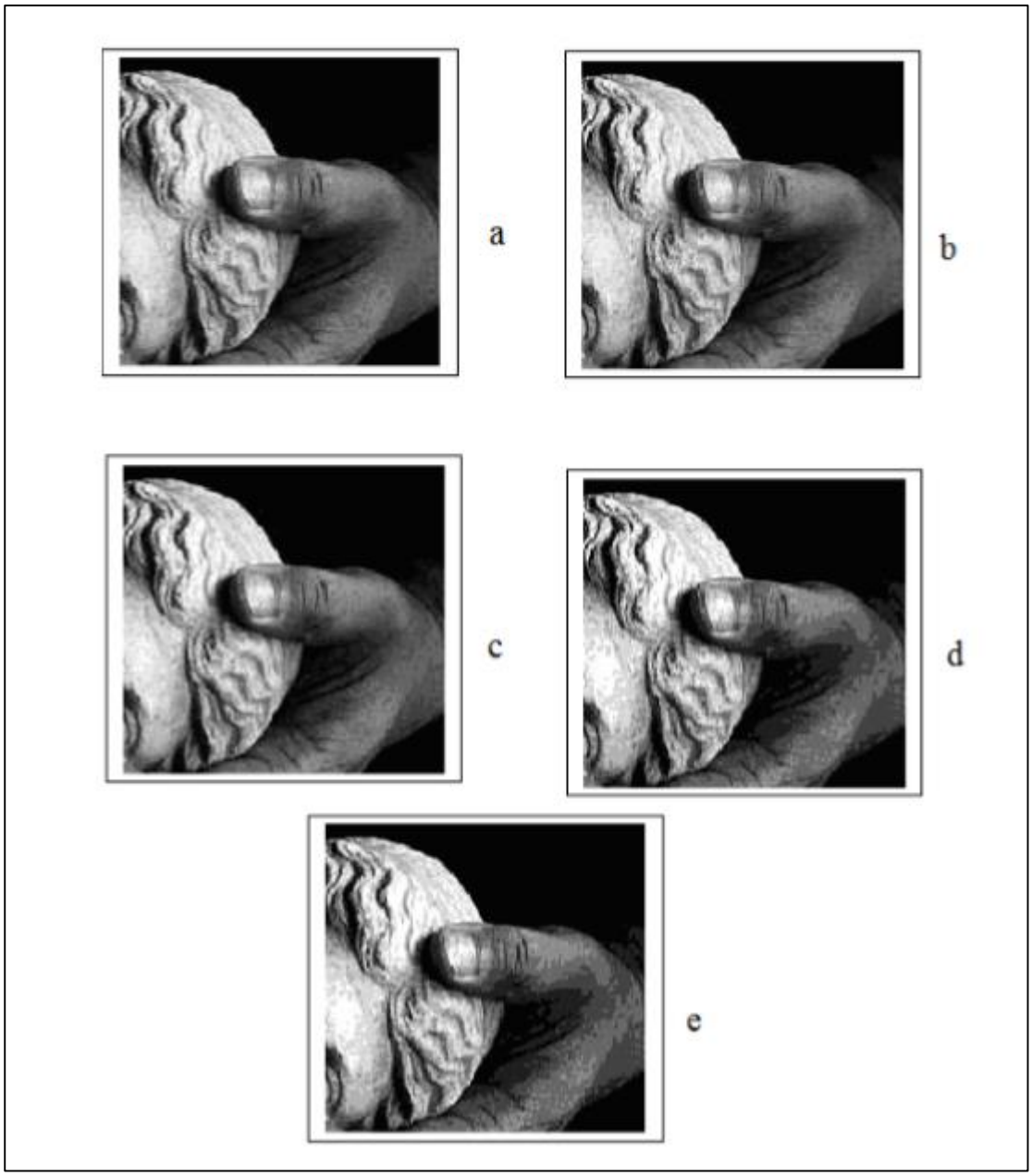
Piksel görüntünün en ve boy bilgisinden bağımsızdır. Enleri ve boyları eşit olan görüntülerin sahip olduğu piksel sayıları farklı olabilir. Tersten düşünülecek olunursa aynı piksel sayısına sahip bir görüntü farklı en ve boyda olabilir, gösterilebilir veya basılabilir. Şekil 3.5'de aynı boyda fakat farklı piksel sayısına sahip görüntüler gösterilmiştir. Birim alandaki piksel sayısı artınca görüntü kalitesi de artar.

Görüntü kalitesini ifade etmek için önemli olan çözünürlük teriminin farklı anlamları bulunmaktadır. Çözünürlük; uzamsal çözünürlük ve parlaklık çözünürlüğü olmak üzere görüntü kalitesini belirten önemli ölçütlerdir [84].

Uzamsal çözünürlük birim alandaki piksel sayılarını ifade eder. Bilgisayar ve mobil cihazların ekran boyutları genellikle inç (inch) olarak ifade edildiği için birim olarak genelde inç kullanılır. Uzunluk birimi olarak inç kullanıldığında uzamsal çözünürlük inç başına düzen piksel (ppi) olarak ifade edilir [84]. Birim alana düşen piksel sayısı artıkça hem görüntü kalitesi hem de görüntünün hafızada kapladığı alan artar. Bunun yanında birim alandaki piksel sayısının artması ile piksellerin kapladığı alan küçülür ve görüntülerdeki ayrıntılar daha iyi görünür hale gelir. Şekil 3.5'de farklı uzamsal çözünürlük değerlerine sahip görüntüler gösterilmektedir ve çözünürlük düştükçe görüntü ayrıntılarında kayıplar yaşanmıştır.

Şekil 3.5. 300x 300 piksele sahip (sol üst), 200 x 200 piksele sahip (sağ üst), 100 x 100 piksele sahip (sol alt) ve 50 x 50 piksele (sağ alt) sahip aynı boyutta olan görüntüler

Parlaklık çözünürlüğü niceleme yapılırken bir pikselin alabileceği farklı parlaklık veya renk seviye sayısını ifade eder [82]. Tek bir renkten oluşan ve 8 bit renk derinliğine sahip bir görüntü $2^8 = 256$ parlaklık ve renk seviyesine sahiptir. RGB (kırmızı, yeşil, mavi) renk modeli kullanıldığında her renk için 8 bit kullanıldığından toplamda 24 bit ile 16.777.216 farklı renk elde edilebilir. Parlaklık çözünürlüğünün artması görüntü kalitesini artması anlamına gelir. Ayrıca parlaklık çözünürlüğü de görüntünün hafızada kapladığı alanı etkiler. Şekil 3.6' da farklı parlaklık çözünürlüğüne sahip görüntüler gösterilmiştir.



Şekil 3.6. a) 128, b)64, c) 32, d)16, e) 8 parlaklık çözünürlüğüne sahip görüntüler [82].

3.3. Görüntü İşleme

Sayısal görüntü işleme, sayısal bilgisayarlar ile sayısal görüntülerin düzenlenmesi, iyileştirilmesi, analiz edilmesi gibi görüntü üzerinde yapılan işlemlere denir [76]. Görüntü analizi yani görüntüyü anlama ve bundan bilgi elde etme hem görüntü işleme ve hem de bilgisayarlı görü konusu içinde geçmektedir. Görüntü işlemenin nerede başladığı nerede bittiği konusunda belirli bir sınır yoktur. Sayısal görüntü işlemede giriş bir görüntü, çıkış bir görüntü olabileceği gibi giriş görüntü çıkış görüntüye ait özellikler veya görüntüden

çıkarılan bir bilgi olabilir. Görüntü işleme işlemleri yapılan işlemlerin karmaşıklığına göre aşağıdaki şekilde gruplandırılabilir [76].

Düşük seviye işleme: Görüntü ön işleme, gürültü azaltma, keskinleştirme, yumuşatma, zıtlık artırma gibi temel seviyede yapılan görüntü işleme işlemlerdir.

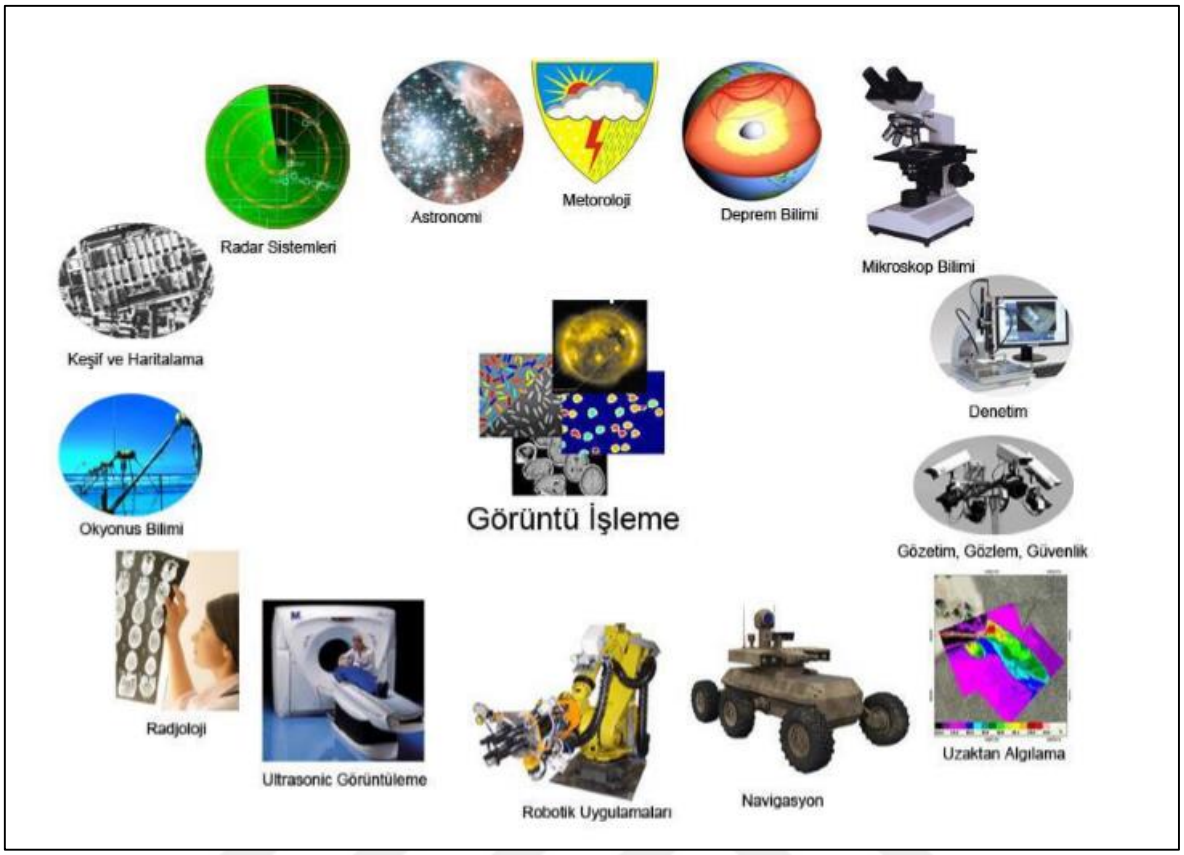
Orta seviye işleme: Görüntüyü nesnelere ve alanlara bölme, nesne ve alanları tanımlama, sınıflandırma gibi işlemler orta seviye işleme olarak kabul edilebilir. Orta seviye işlemede giriş genelde bir görüntü çıkış ise kenar, köşe veya tanımlanan nesneler gibi görüntüden elde edilen özellikler olur.

Yüksek seviye işleme: Görüntüden bilgi çıkarma işlemleri, nesne tanıma, görüntü analizi sonucu bir yargıya varma gibi işlemler yüksek seviye işleme olarak adlandırılabilir.

Yukarıda bahsedilen farklı seviyelerde görüntü işleme teknikleri bulunmaktadır. Görüntü işleme temelde görüntünün elde edilmesinden başlayarak, sayısallaştırılması, iyileştirilmesi, analiz edilmesine ve saklanmasına kadar birçok tekniği kapsamaktadır. Bu teknikler genelde pikseller üzerinde yapılan matematiksel hesaplamaları içermektedir.

3.4. Görüntü İşleme Uygulama Alanları

Görüntü işleme hemen hemen her alanda kullanım imkânı bulmuştur ve gün geçtikçe kullanımı giderek artmaktadır. Astronomi, tip, uzaktan algılama, savunma sanayi, arama motorları, meteoroloji bu alanlardan bazılarıdır. Şekil 3.7’ de görüntü işlemenin bazı kullanım alanları verilmiştir



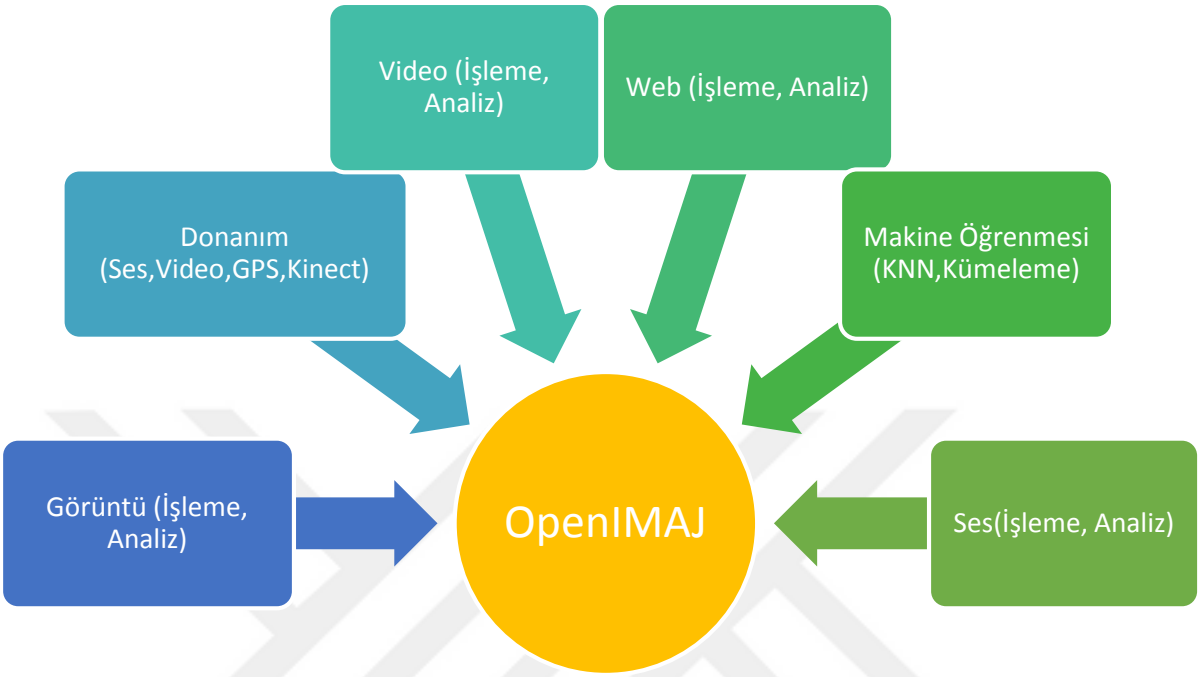
Şekil 3.7. Görüntü işleme kullanım alanları [85].

3.5. OpenIMAJ

Görüntü işlemenin yaygınlaşmasıyla karşılaşılan problemlerin çözümü için farklı birçok çözüm geliştirilmiştir. Bu çözümler çeşitli lisanslar altında yazılım kütüphaneleri haline getirilmişlerdir [86, 87].

OpenIMAJ, çoklu ortam içerik üretimini ve analizini sağlamak amacıyla oluşturulmuş kütüphane ve araçlarından oluşan açık kaynak kodlu bir Java kütüphanesidir [86]. Southampton Üniversitesi Elektronik ve Bilgisayar Bilimleri bölümünde geliştirilmiş olan OpenIMAJ, Avrupa birliği projesi kapsamında da destek almıştır [86]. OpenIMAJ, ACM Multimedia 2011 OSSC Awards- "*Best Open Source Software Competition Awards"* en iyi açık kaynak yazılım yarışması ödüllünün sahibi olmuştur [88]. Bilgisayar görmesi ve görüntü işleme ile alakalı birçok modern algoritmayı içeren kütüphane, ileri veri kümelemeden içerik analizine kadar birçok araç ve kütüphaneden oluşur. Şekil 3.8'de OpenIMAJ kütüphanesi ile yapılabilen bazı işlemler gösterilmiştir. OpenIMAJ Java diliyle yazıldığından platform bağımsızdır. Kütüphane Groovy, Jython, JRuby ve Scala gibi dilleri desteklemektedir. Kütüphane çok az native kod içermesinin yanında tamamen Java dili ile

yazılmıştır. Java dili ile yazılmış olmasından dolayı bu dili kullanan sistemlerde çoklu medya işleme ve analizi alanındaki ihtiyaçlara cevap vermesi açısından önemli bir kütüphanedir.



Şekil 3.8. OpenIMAJ kütüphanesiyle yapılabilen bazı işlemler

3.5.1. OpenIMAJ Kütüphanesin Bileşenleri

OpenIMAJ'dan bir kütüphane olarak bahsedilse de modüler olarak birçok kütüphane ve araçtan oluşmaktadır. Bu bileşenler modüler yapıda olmasından dolayı ihtiyaca göre kütüphanenin sadece ihtiyaç duyulan modülleri kullanılabilir. OpenIMAJ kütüphanesinin web sayfasındaki dokümantasyon sayfasında [89] yer alan bazı temel bileşenleri aşağıda verilmiştir [86, 90].

- **core**: **Modüllerin genelinde kullanılan temel fonksiyonları içeren alt modüldür.**

core: Çoklu medya işleme ile ilgili problemlerden çok genel programlama problemleri ile ilgili fonksiyonları içeren temel kütüphanedir. I/O (Input/Output - Giriş/Çıkış) araçları, rassallık, özütleme, tip dönüşümleri gibi işlemleri gerçekleştirir.

core-image: Piksel veya bağlantılı bileşen olarak görüntüleri tanımlamayı, basit tiplerin işlemesi ile alakalı ara yüzleri, görüntülerin yüklenmesi, gösterilmesi ve saklanması işlemlerini içerir.

core-video: Video tipleri ile ilgili temel tanımları, videoları göstermek ve işlemek için fonksiyonları içerir.

core-audio: Ses ile alakalı temel tanımları ve basit tipler ile alakalı temel işlemleri içerir.

core-math: Geometrik, matris, istatistiksel operatörleri içeren matematiksel tanımları içerir.

core-feature: Özellikleri ifade eden temel kavramlar genellikle bir veri dizisi olarak belirtilir. Bütün ilkel veri tipleri için özellik tanımlama, özelliklerin disk veya hafızadaki yerleri ile özelliklerin listelenmesi gibi işlemleri gerçekleştirir.

core-experiment: Elde edilen sonuçların otomatik olarak değerlendirilmesi için deney ve sonuçların otomatik olarak uygun bir şekilde oluşturmasını sağlayan sınıfları içerir.

core-citation: Yayın referanslarını oluşturmak için araçlar sağlar.

core-aop-support: Aspect Oriented Programming ve Bytecode temel desteğini sunar.

- **image**: **Görüntü ile ilgili işlevleri içeren alt modüldür.**

image-processing: Görüntü, piksel ve bağlantılı birleşenlerin işlenmesi ile alakalı işleme işlemlerini içerir (yeniden boyutlandırma, kenar detektörü vb.).

image-local-features: Yerel özelliklerin çıkarılması ile alakalı metotları içerir. Yerel özellikler, Difference of Gaussian, Harris gibi detektörler ile görüntülerin belli bölgeleri için elde edilen (Sift vb.) tanımlayıcılardır.

image-feature-extraction: Düşük seviye görüntü özelliklerini çıkarmak için metotları içerir. Piksel/Patch sınıflama modellerini ve global görüntü özelliklerini içerir.

faces: Yüz tanıma işlemi için sınıfları içerir.

image-annotation: Otomatik olarak resimlere açıklama notu ekleme işlemi ile ilgili sınıfları içerir.

object-detection: Nesne tanıma işlemi için sınıflar ve Haar – Cascade detektör uygulamasını içerir.

image-indexing-retrieval: CBIR ile ilgili bileşenler için alt projedir.

vector-image: Batik SVG kütüphanesini kullanarak vektör görüntü desteğini sağlar.

camera-calibration: Kamera kalibrasyon teknikleri ve bununla alakalı sınıfları içerir.

multiview: OpenIMAJ 3 boyutlu görüntüleme kütüphanesidir.

- **video: Video işleme ve analizini sağlayan alt modülleri içerir.**

video-processing: Çeşitli video işleme algoritmalarını içerir.

xuggle-video: Xuggler kütüphanesini video kaynağı olarak kullanabilmek için geliştirilen bir eklentidir.

- **audio: Ses işleme ve analizi için fonksiyonları içerir.**

audio-processing: Ses değiştirme gibi çeşitli ses işleme işlemlerini içerir.

- **machine-learning: Makine öğrenmesi kütüphaneleri için alt kütüphanedir.**
- **text: Metin analizi için fonksiyonları içeren OpenIMAJ kütüphanesidir.**
- **tools: : Komut satırından girdi alabilen OpenImaj ile geliştirilen araçların bulunduğu alt projedir.**

core-tool: OpenIMAJ kütüphanesine araçlar geliştirmek için oluşturulan temel kütüphanedir.

GlobalFeaturesTool: Görüntülerden çeşitli global özelliklerin çıkarılmasını sağlayan araçtır.

LocalFeaturesTool: Görüntülerden yerel özelliklerin çıkarılmasını sağlayan araçtır.

openimaj-processing: OpenIMAJ kütüphanesine işleme işlemlerini entegre eden araçlardır.

- **hadoop: OpenIMAJ ile Apache Hadoop projesini birlikte kullanılmasını sağlayan alt modülerdir.**

core-hadoop: Map-reduce ve sequence-files için temel işlemleri sağlayan temel kütüphanedir**.**

Tools: Map-reduce olarak tanımlanan çoklu medya analizi algoritmalarının Hadoop kümeleri üzerinde koşturulmasını sağlayan araçlardır.

core-hadoop-tool: Hadoop kümesi üzerinde map-reduce işlemi ile kümeleme ve özelliklerin niceleme için araçları sağlar.

SequenceFileTool: Hadoop SequenceFiles oluşturma, okuma gibi işlemleri sağlayan araçtır.

Bu tez çalışmasında OpenIMAJ kütüphanesi kullanılmıştır. Yukarıda OpenIMAJ kütüphanesinin tez açısından önemli görülen bileşenleri verilmiştir. Kütüphanenin Java ile yazılmış olması, açık kaynak kodlu bir kütüphane olması, çoklu medya işleme ve analizi ile alakalı zengin bir içeriğe sahip olmasından dolayı bu tez çalışmasında tercih edilmiştir. Tez çalışmasında dağıtık görüntü işleme için Hadoop kütüphanesi kullanılmıştır. Hadoop kütüphanesinin Java ile geliştirilmiş olmasından dolayı ve görüntü işleme için kullanılan kütüphanelerin Java ile geliştirilmiş olması önemli bir avantaj sağlamıştır. Görüntü işleme alanında OpenCV [87] gibi başka açık kaynak kütüphaneler de mevcuttur. OpenCV kütüphanesi C++ diliyle geliştirilmiştir. OpenCV'yi Java dili ile kullanmak için Java Native Interface (JNI) [91] ara yüzünden yararlanılmaktadır. OpenCV kütüphanesinin kullanılabilmesi için işletim sistemine özel derlenip kurulması gerekmektedir. Dağıtık bir

sitemde kümeyi oluşturan her bilgisayar için bu işlemi yapmak zahmetli bir işlem olacaktır. Bu gibi durumlar göz önüne alınarak OpenIMAJ Kütüphanesi tercih edilmiştir.

3.6. Dağıtık Görüntü İşleme

Son yıllarda, internetin yaygınlaşması, görüntü algılama donanımlarının ucuzlaması ve yaygınlaşması, endüstri, güvenlik, tıp, uydu teknolojileri gibi birçok alanda görüntü verilerinin önem kazanması ve buna bağlı olarak üretilen görüntü verilerinin artması sonucunda görüntü verileri klasik yöntemler ile işlenemeyecek boyutlara ulaşmıştır. Görüntü işlemenin yoğun hesaplama ve hafıza gerektiren bir işlem olması bu işlemi daha da zorlaştırmıştır.

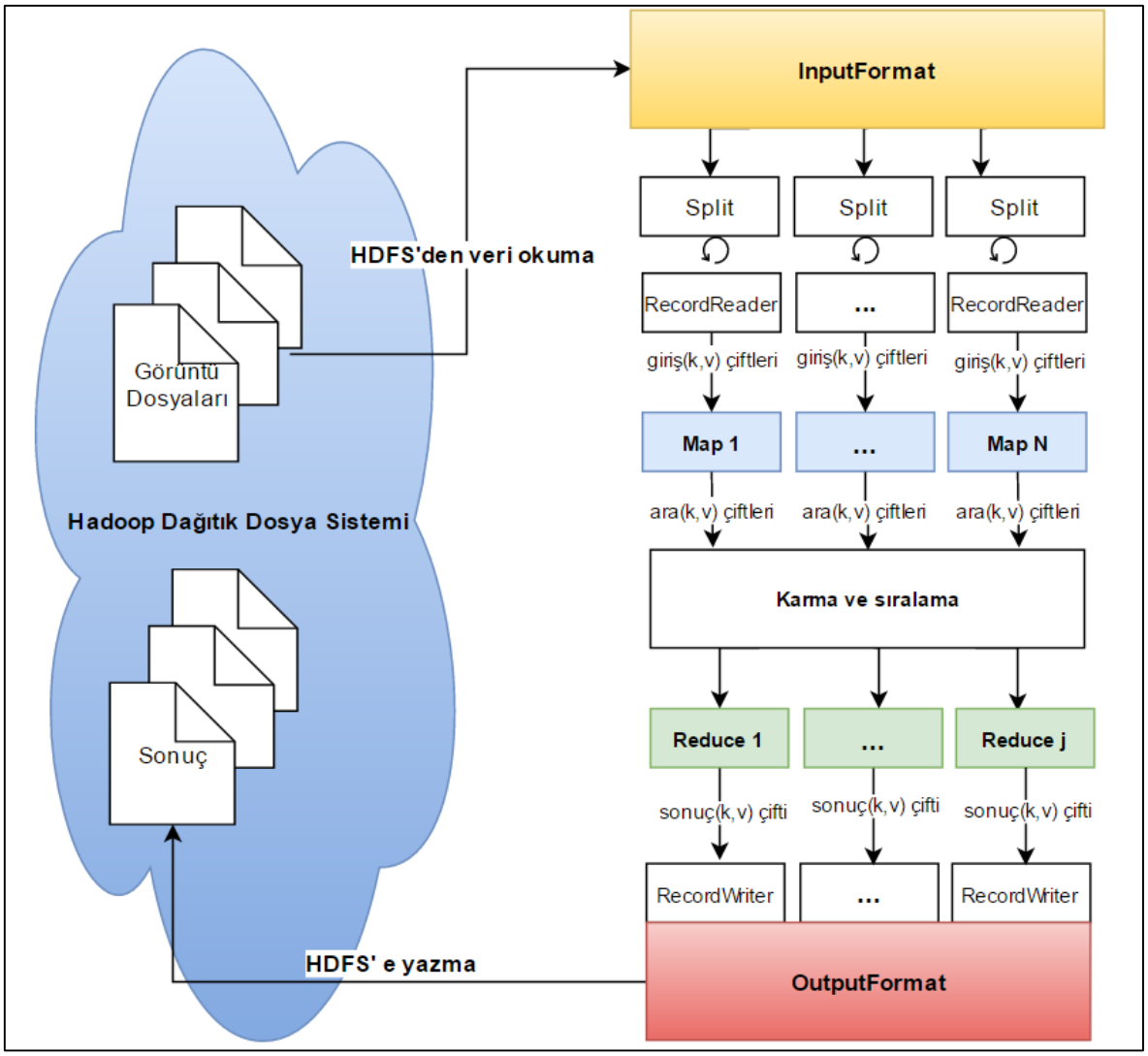
Büyük hacimli verilerin depolanması ve işlenmesi için farklı çözümler sunulmuştur. Bu verilerin işlenmesi için ilk akla gelen çözümlerden biri çok işlemcili ve büyük depolama kaynağına sahip süper bilgisayarları kullanmak olmuştur. Verinin sürekli ve hızlı bir şekilde artması göz önünde bulundurulursa bir bilgisayarın işlemci ve depolama kaynağının da bu veriyi işleyebilecek şekilde artırılması gerekmektedir. Büyük hacimli verileri işlemek için bu özelliklerdeki süper bilgisayarlara sahip olmak maliyetli bir çözüm olduğu için daha ekonomik ve daha sürdürülebilir çözümlerin araştırılması ihtiyacı doğmuştur.

İkinci bir yaklaşım ise büyük hacimli ve büyük veri olarak adlandırılan bu veri ve görüntüleri Message Passing Interface (MPI) ara yüzü ile dağıtık olarak işlemektir [8]. Bu yaklaşım ile veriler dağıtık hesaplama düğümleri üzerine dağıtılarak hesaplama işlemleri gerçekleştirilmektedir. Bu yaklaşımda alanında uzman sistem programcılarına ve geliştiricilerine ihtiyaç duyulmaktadır çünkü paralel programlamanın birçok aşamasında sistem programcılarının ve geliştiricilerinin kontrolü gerekmektedir [8]. Bu ayrıntı seviyesinde kontrolün ve sorumluluğun sistem programcılarında ve geliştiricilerinde olması sistem programcıların ve geliştiricilerin gereksiz bir karmaşıklıkla da uğraşmalarına neden olmaktadır [15].

Google'ın MapReduce programlama modelinin sunulmasıyla beraber bu programlama modeli ile büyük veri, özelde görüntü verileri işlenmeye başlanmıştır [8-18, 92]. MapReduce programlama modelinin açık kaynak uygulamasını içeren Hadoop projesi birçok araştırmacı ve firma tarafından kullanılmaktadır [58]. Hadoop, sağladığı basit MapReduce programlama modeli ve dağıtık dosya sistemi ile programcılara önemli kolaylıklar sağlamıştır.

Şekil 3.9'da Hadoop MapReduce ile örnek bir dağıtık görüntü işleme sistemi gösterilmiştir. Bu sistemde görüntü dosyaları Hadoop Dağıtık Dosya Sistemi'nde tutulmaktadır. Görüntü verilerinin HDFS'den okunup Hadoop sistemi tarafından kullanılabilmesi için Hadoop'un InputFormat sınıfından türetilmiş görüntü dosyalarını tanımlayan ve okumasını sağlayan bir sınıfın yazılması gerekmektedir. Hangi formatta görüntü verileri okunacaksa o formata uygun bir RecordReader sınıfının yeniden yapılandırılması gerekmektedir. Görüntü işleme işlemi map işlemi sırasında yapılır. Kullanıcı tarafından tanımlanan map fonksiyonuna anahtar, değer çifti olarak okunan görüntüleri gönderilir. Map aşamasında bu görüntülere görüntü işleme teknikleri uygulanır. Map işleminin sonucu bir görüntü, görüntüden çıkarılan bir özellik veya görüntüden elde edilen özel bir bilgi olabilir. Yapılmak istenen işleme göre map fonksiyonu sonucu olarak üretilen çıktı değişebilir. Bu aşamada yapılacak görüntü işleme işlemleri için uygun bir görüntü işleme kütüphanesinin kullanılmasında fayda vardır. Map fonksiyonu tarafından kullanılacak ve map fonksiyonu tarafından üretilen anahtar, değer veri çifti olarak tanımlanmaktadır. Bu çiftleri tanımlamak için Hadoop tarafından sağlanan veri tipleri kullanılabilir. Hadoop'un sağladığı veri tiplerinin ihtiyaca cevap vermemesi durumda veriye özel yeni bir veri tipi tanımlaması gerekecektir. Map fonksiyonu sonucunda elde edilen ara (anahtar, değer) çiftleri sıralama işleminden geçirilir. Daha sonra bu ara (anahtar, değer) çiftleri kullanıcı tarafından tanımlanan reduce fonksiyonuna verilir. Reduce fonksiyonu sonucunda elde edilen sonuç OutputFormat sınıfı ile RecordWriter sınıfı kullanılarak HDFS ortamına yazılır. İhtiyaca göre programcı yeni bir OutputFormat sınıfı ve RecordWriter sınıfı tanımlaması gerekebilir.

Bu tez çalışmasında video ve görüntü dosyaları için yukarıda bahsedilen sisteme benzer bir sistem geliştirilmiştir. Bunun için yukarıda bahsedilen InputFormat, RecordReader, OutputFormat, RecordWriter ve yeni anahtar, değer veri yapıları gerçekleştirilen sistemin ihtiyaçlarına göre yeniden yazılmıştır. Tez kapsamında gerçekleştirilen sistem sonraki bölümde ayrıntılı olarak anlatılacaktır.



Şekil 3.9. Hadoop MapReduce ile dağıtık görüntü işleme

4. BÜYÜK VERİ TEKNOLOJİLERİ İLE DAĞITIK GÖRÜNTÜ İŞLEME SİSTEMİ TASARIMI

Bu tez çalışmasında Hadoop ile dağıtık görüntü ve video işleme ve için gerekli olan Hadoop sınıfları yazılmış. OpenIMAJ çoklu medya işleme ve analiz kütüphanesinin Hadoop ile kullanılabilmesi için gerekli Java sınıfları geliştirilmiştir. Daha sonra yazılan bu sınıflar ile dağıtık olarak görüntü ve video işleme için iki uygulama geliştirilmiştir. Yazılan bu uygulamalar TÜBİTAK BİLGEM Bilişim Teknolojileri Enstitüsü (BTE) tarafından, T.C. Kalkınma Bakanlığı Yatırım Programı desteğiyle yürütülen "Bulut Bilişim ve Büyük Veri Araştırma Laboratuvarı" (B3LAB) [93] projesi çerçevesinde bilimsel araştırmaların yapılması için sunulan OpenStack kurulu bulut sistemi üzerinde oluşturulan Hadoop kümesi üzerinde koşturularak performans analizi yapılmıştır.

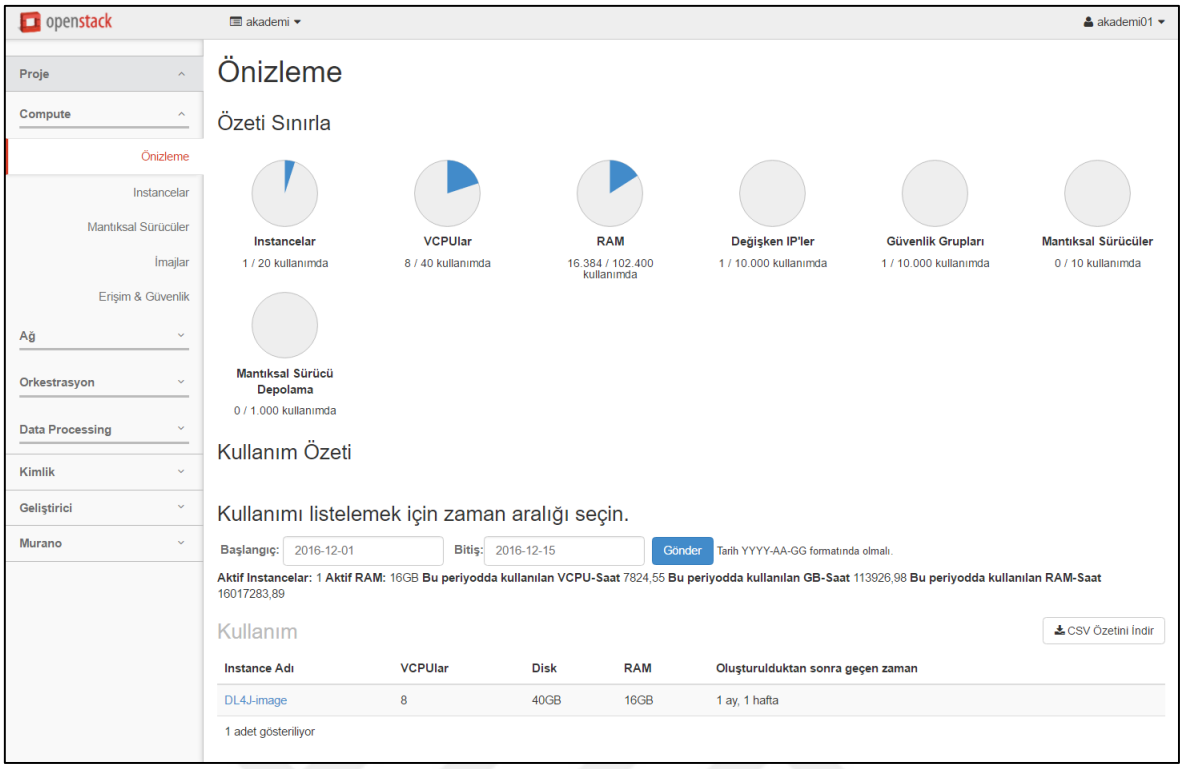
4.1. OpenStack ile Sanal Makine ve Hadoop Kümesi Oluşturma

OpenStack [33] ağ, depolama, hesaplama kaynaklarından oluşan büyük hacimli kaynak havuzlarının kontrolünü ve bir web ara yüzü ile de yöneticilere bu sistemin yönetimini sağlayan bir bulut işletim sistemidir [94]. OpenStack, altyapıyı hizmet olarak (IaaS) sunan açık kaynak kodlu bir yazılımdır. OpenStack'ın dünya çapında birçok kullanıcısı bulunmakta ve IBM, Intel, Redhat, Cisco gibi birçok firma tarafından desteklenmektedir [95]. OpenStack sayesinde kuruluşlar kendi bulut bilişim altyapılarını kolay bir şekilde oluşturabilmektedir.

Bu bölümde TÜBİTAK BİLGEM bulut sisteminin OpenStack web ara yüzü kullanılarak sanal makine ve Hadoop kümelerinin oluşturulması anlatılacaktır. Şekil 4.1'deki gibi bir giriş ekranına sahip olan B3LAB OpenStack çoklu kullanıcılı bir sitemdir.

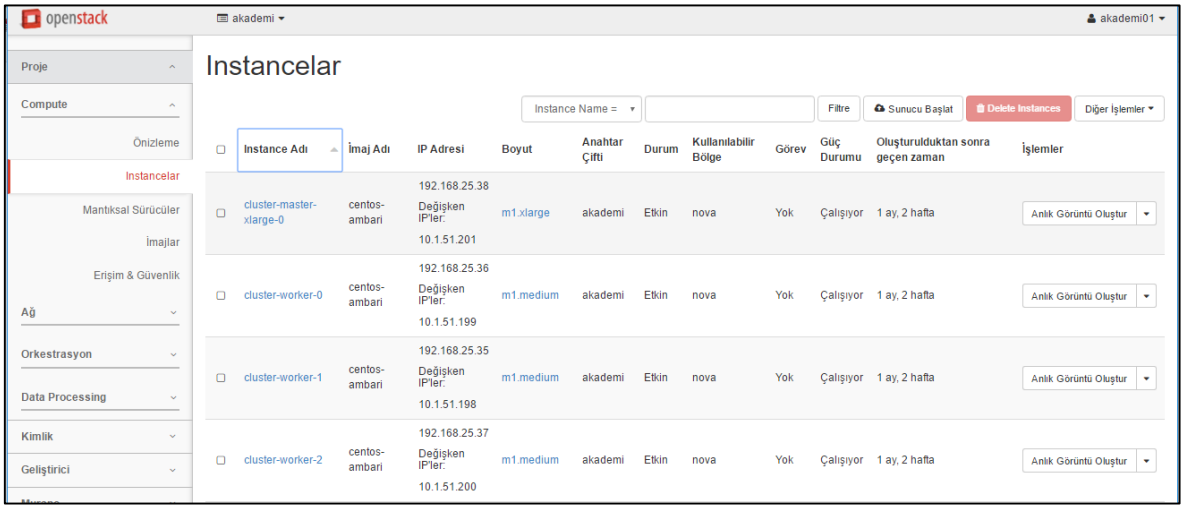
Şekil 4.1. OpenStack giriş ekranı

OpenStack sisteminde kullanıcılar farklı yetki seviyelerinde olabilir. Yönetici yetkisine sahip bir kullanıcı proje oluşturma, silme; sanal makine oluşturma, değiştirme, silme; kullanıcı ekleme, güncelleme, silme gibi birçok işlemi gerçekleştirebilir. Sisteme yönetici yetkisine sahip kullanıcı adı ile giriş yapılarak bir proje oluşturulması gerekmektedir. Proje oluşturulurken kullanıcı ve kaynak kotaları belirtilir. Şekil 4.2'de oluşturulan proje ön izleme ekranı gösterilmiştir. Bu ekranda projenin kaynakları ile ilgili bilgiler yer almaktadır.



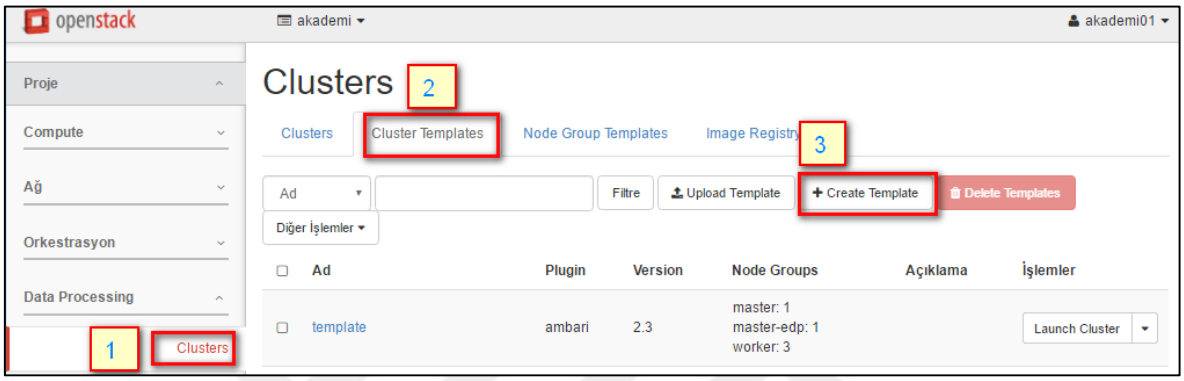
Şekil 4.2. Proje ön izlemesi

OpenStack ara yüzü ile tez çalışması için oluşturulan sanal makine örnekleri Şekil 4.3'de gösterilmiştir. Başlangıçta 1 adet usta 2 adet işçi sanal makine oluşturulmuştur. Usta sanal bilgisayar 8 vCPU, 8GB RAM, 160GB disk kapasitesine sahiptir. İşçi sanal bilgisayarları kaynakları 4 vCPU , 8GB RAM, 80GB disk kapasitesi olacak şekilde ayarlanmıştır. Daha sonra işçi sanal bilgisayar sayısı artırılarak küme yatay olarak ölçeklendirilmiştir.



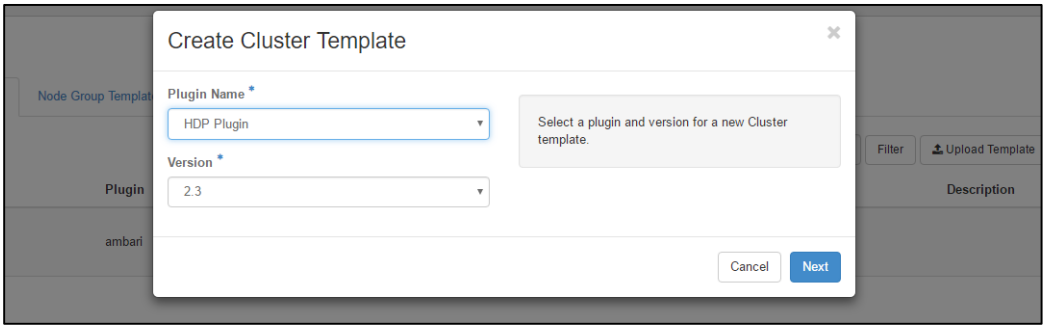
Şekil 4.3. OpenStack ile oluşturulan sanal makine örnekleri

Aşağıda OpenStack ara yüzü aracılığı ile Hadoop kümesini oluşturulması anlatılmıştır. Bunun için ilk önce bir küme şablonu oluşturulur. Şekil 4.4' de gösterildiği şekilde Data Processing menüsünün altındaki Cluster alt menüsü tıklanır. Daha sonra Cluster Templates sekmesine tıklanır. Bu ekranda daha önce oluşturulan şablonlar yer almaktadır. Yeni bir şablon oluşturmak için Create Template butonu tıklanır.



Şekil 4.4. Küme şablonu oluşturma 1. adım

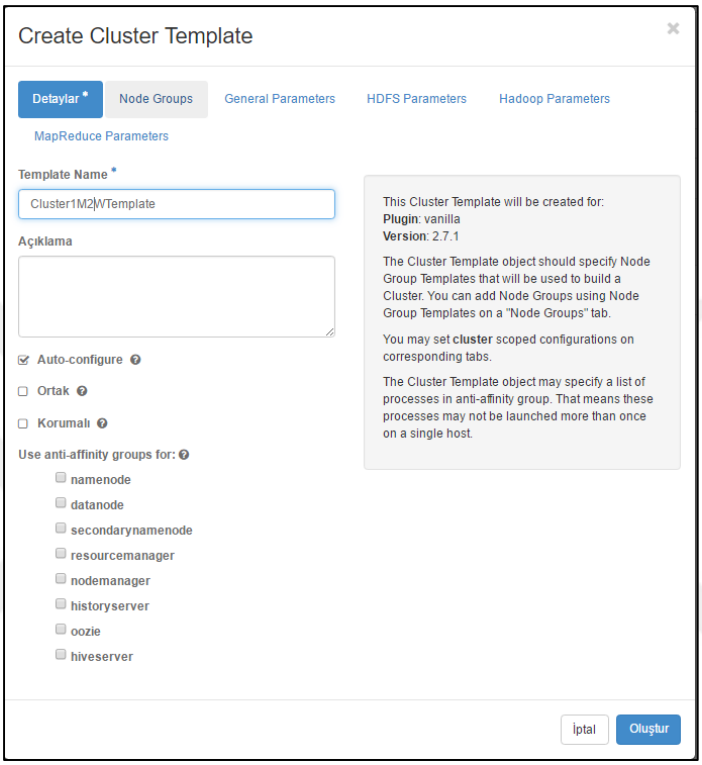
Şekil 4.5' de gösterildiği gibi açılan ekrandan sistemde kurulu olan eklentilerden biri olan HDP plugin ve versiyonu seçilerek Next butonu tıklanır.



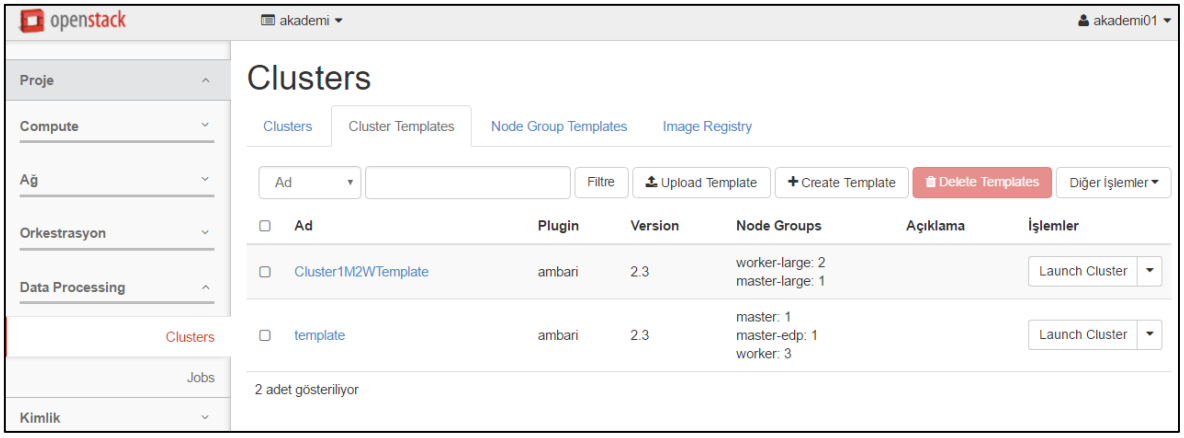
Şekil 4.5. Küme şablonu oluşturma 2. adım

Şekil 4.6'da gösterilen ekran aracılığı ile küme üzerinde çalışacak veri işleme platformlarının parametre ayarları yapılır. Şekil 4.6'da gösterildiği ekrandan usta ve işçi (worker - slave) olarak çalışacak makine özellikleri ve sayısı belirlenir. Bu tez çalışmasında kullanılmak üzere başlangıçta 1 usta makine, 2 işçi makineden oluşan bir küme için bir şablon oluşturulmuştur. Usta (Master) makine 8 vCPU, 16 GB RAM, 160 GB depolama kaynağına sahip olacak şekilde yapılandırıldı. İşçi makineler 4 vCPU, 8 GB RAM, 80 GB

depolama kaynağına sahip olacak şekilde ayarlanmıştır. Oluşturulan Cluster1M2WTemplate adında küme şablonu Şekil 4.7' de gösterilmektedir. Bu ekranda her şablonun yanında bulunan Launch Cluster butonu ile çok hızlı bir şekilde daha önce şablonu oluşturulan küme oluşturulup başlatılabilir.



Şekil 4.6. Küme şablonu oluşturma 3. adım



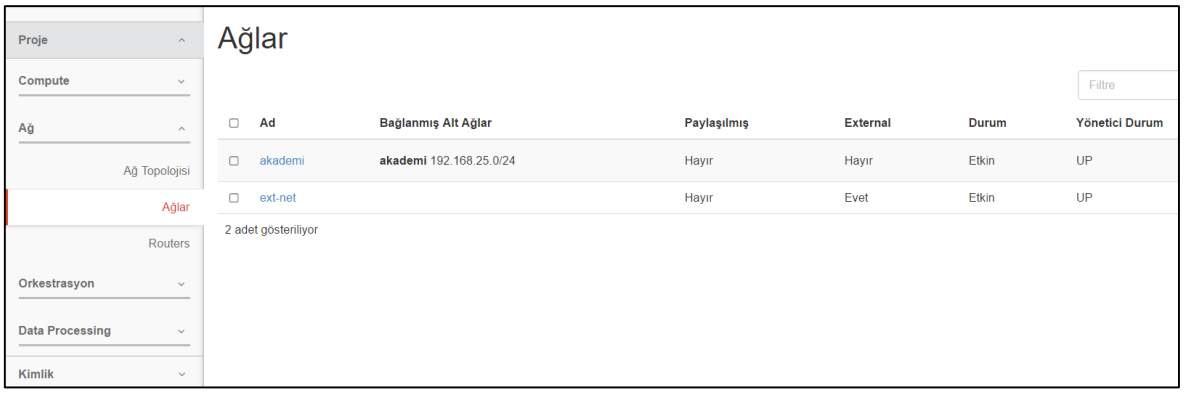
Şekil 4.7. Oluşturulan küme şablonu

Küme şablonundan küme oluştururken gerekli olan ve oluşturulan makinelere erişmek için kullanılacak olan bir anahtar çiftinin daha önceden oluşturulmuş olması gerekmektedir.

Oluşturulan anahtar çiftlerini listelemek ve yeni anahtar çifti oluşturmak için "Compute" menüsünün altındaki "Erişim & Güvenlik" alt menüsü ile erişilen Şekil 4.8'de gösterilen grafik ara yüzü kullanılabilir. Oluşturulacak kümedeki makinelere ssh bağlantısı için gerekli olan akademi adında bir anahtar oluşturulmuştur. Bu anahtar daha sonra oluşturulan küme şablonundan Hadoop kümesi oluşturmak için kullanılacaktır.

Şekil 4.8. Anahtar çiftlerini listeleme ve yeni anahtar çifti oluşturma

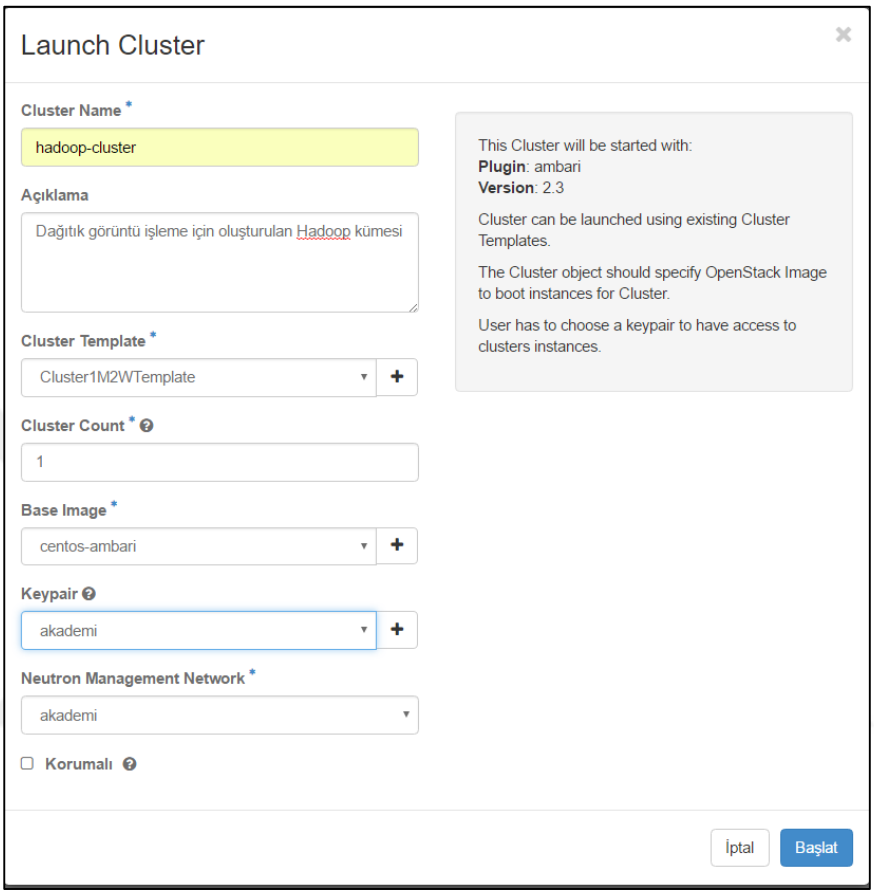
Küme oluşturulurken gerekli olan, ağ yapısının da önceden ayarlanması gerekmektedir. Şekil 4.9'da gösterilen ara yüz aracılığı ile gerekli ayarlamalar ve ağ yapısı oluşturulur. Oluşturulacak küme bilgisayarlarının dahil edileceği akademi adında bir ağ topolojisi oluşturulmuştur.



Şekil 4.9. Ağ topolojisi oluşturma

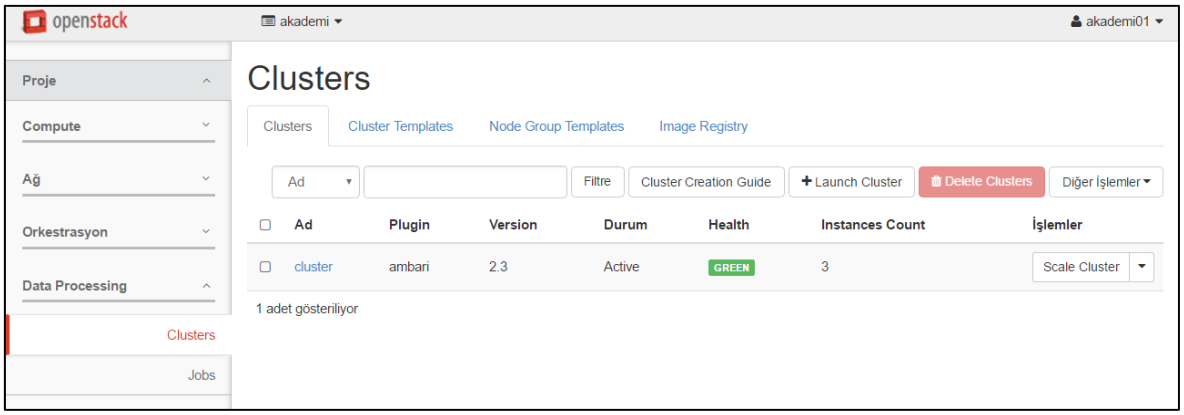
Anahtar çifti, ağ topolojisi ve küme şablonu oluşturulduktan sonra Hadoop kümesini oluşturmak için Şekil 4.7'de gösterilen ekrandan Cluster1M2WTemplate şablonun yanında bulunan Launch Cluster butonuna tıklanır. Şekil 4.10' da gösterilen ekrandan Hadoop kümesine verilmek istenen ad girilir. Daha önce oluşturulan Cluster1M2WTemplate

şablonu seçilerek, oluşturulacak küme sayısı, kullanılacak işletim sistemi imajı, daha önce oluşturulan akademi adında anahtar çifti ve yine daha önce oluşturulan akademi ağı seçilerek başlat butonuna tıklanarak küme başlatılır.



Şekil 4.10. Küme şablonundan küme oluşturma

Yukarıdaki işlem bittiğinde Cluster1M2WTemplate şablonundan oluşan küme Şekil 4.11' de gösterildiği gibi kullanıma hazır duruma gelir.

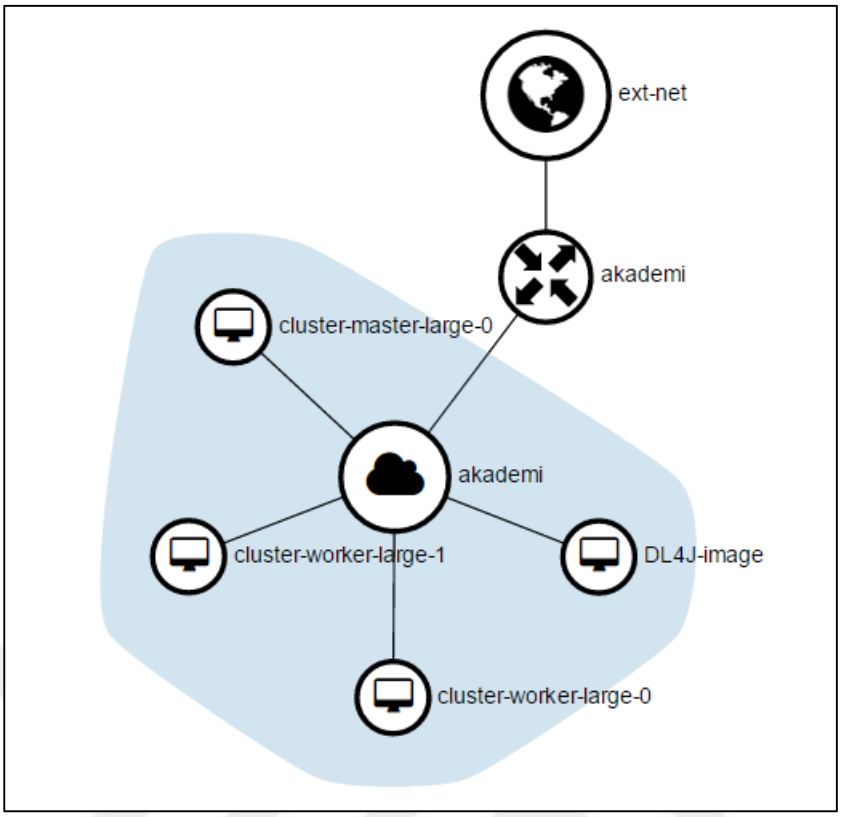


Şekil 4.11. Cluster1M2WTemplate şablonundan oluşturulan Hadoop kümesi

Tablo 4.1'de oluşturulan Hadoop kümesindeki makinelerin özellikleri gösterilmiştir. Tabloda gösterilen vCPU, RAM, disk boyutu özellikleri küme şablonu oluşturulurken belirlenen değerlerdir. Küme hangi şablondan oluşturulursa o şablonda tanımlanan özelliklerle başlar. Şekil 4.12' de ise oluşturulan Hadoop kümesinin ağ topolojisi gösterilmektedir. Küme başlatılırken akademi ağı seçildiği için oluşturulan makineler akademi ağının birer üyesi olarak sisteme dahil edilmiştir.

Tablo 4.1. Hadoop kümesindeki makine özellikleri

Adı	**VCPUlar**	**Disk**	**RAM**
cluster-master-large-0	8	160GB	8GB
cluster-worker-large-0	4	80GB	8GB
cluster-worker-large-1	4	80GB	8GB



Şekil 4.12. Hadoop kümesinin ağ topolojisi

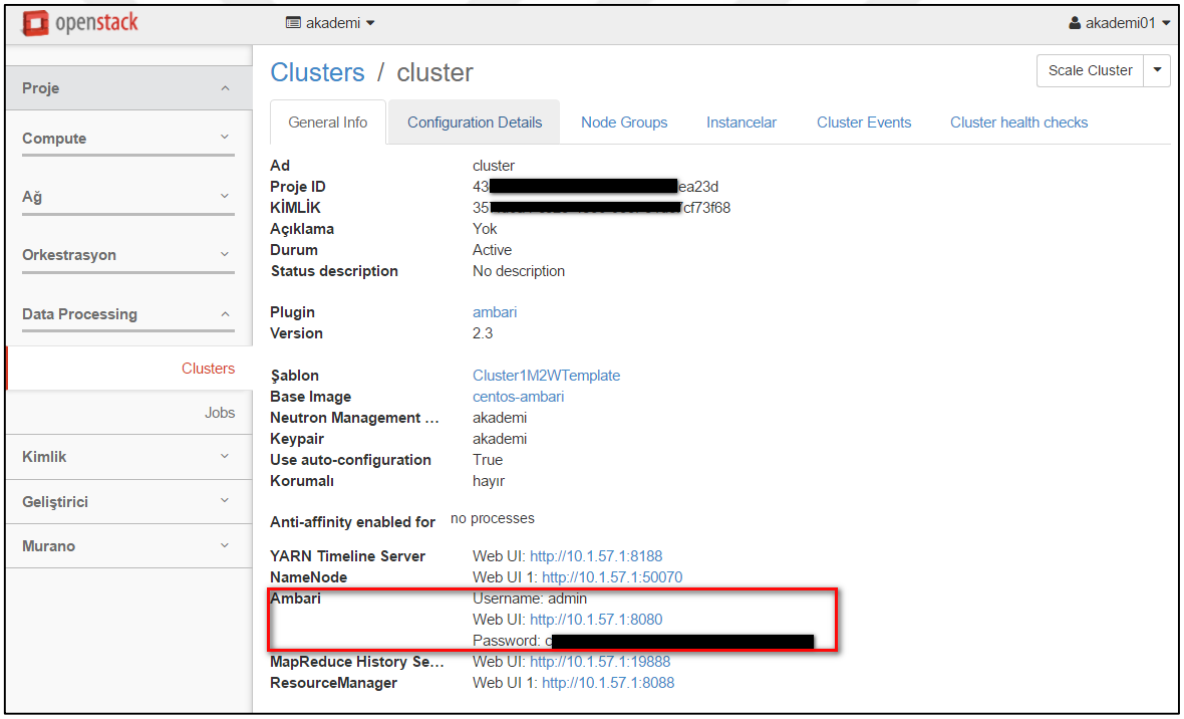
B3LAB OpenStack ortamına entegre edilmiş Apache Ambari [96] projesi sayesinde oluşturulan kümeler kolay bir şekilde yönetilebilmekte ve izlenebilmektedir. Apache Ambari, Hadoop kümelerinin kolay bir şekilde oluşturulması, yönetilmesini ve izlenmesini sağlayan açık kaynak kodlu bir projedir [96]. Ambari'nin sağladığı web ara yüzleri ile Hadoop kümelerinin oluşturulması, yönetilmesi ve izlenmesi oldukça basite indirgenmiştir.

Ambari ile gerçekleştirilebilen bazı işlemler aşağıda maddeler halinde verilmiştir [96]:

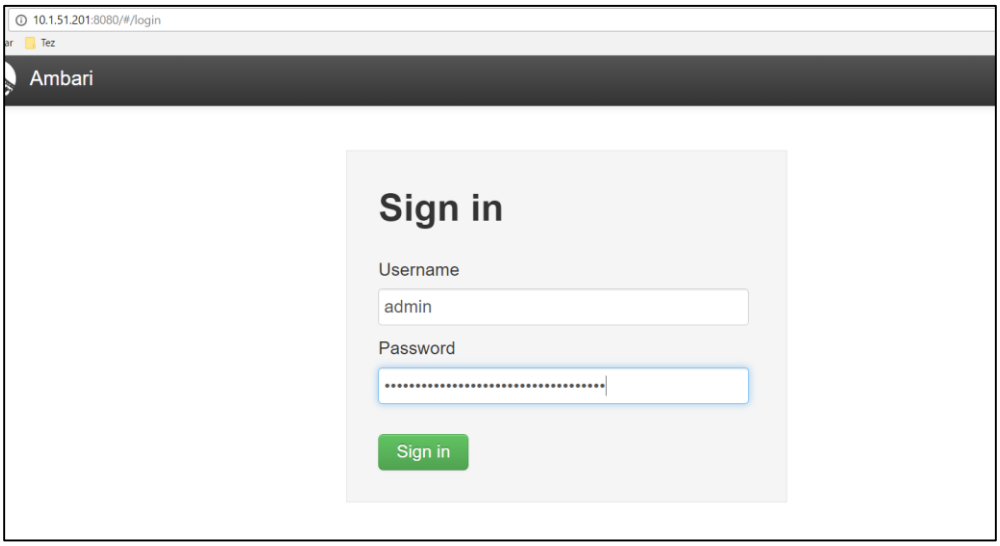
- Hadoop kümeleri hızlı bir şekilde oluşturulabilir. Ambari bunun için adım adım kurulumu sağlayan bir sihirbaz sunar.
- Ambari ile çok kolay bir şekilde Hadoop servislerinin ayarları yapılabilir ve ihtiyaç halinde web ara yüzü ile çok kolay bir şekilde bu ayarlar değiştirilebilir. Yapılan değişikliklerin geçmişi versiyon olarak tutulmaktadır. İstendiğinde eski ayarlara dönülebilir.
- Hadoop servislerinin yapılandırılması, başlatılması, durdurulması, yeniden başlatılması gibi işlemlerin gerçekleştirilmesi için merkezi bir yönetim sağlar.
- Ambari, sistem metriklerini ve Hadoop servislerinin metriklerini toplar ve bu metrikleri grafikler ile kullanıcılara sunar.

- Ambari Alert Framework sayesinde herhangi bir servis çalışmadığında veya yolunda gitmeyen bir durum olduğunda, kaynaklar tükenmek üzere olduğunda uyarı üretilerek kullanıcılar uyarılır.
- Ambari Restfull Api sayesinde uygulama geliştiricileri yukarda sayılan özellikleri kolaylıkla sistemlerine entegre edebilirler

Oluşturulan Hadoop kümesinin genel bilgileri Şekil 4.13'de gösterilmiştir. Bu bilgiler arasında B3LAB OpenStack ortamına entegre edilmiş Apache Ambari sistemine giriş bilgileri de yer almaktadır. Bu bilgiler ile Ambari web sistemine giriş yapılarak oluşturulan küme kolay bir şekilde yönetilip, izlenebilir. Şekil 4.14'de Ambari giriş ekranı gösterilmiştir.

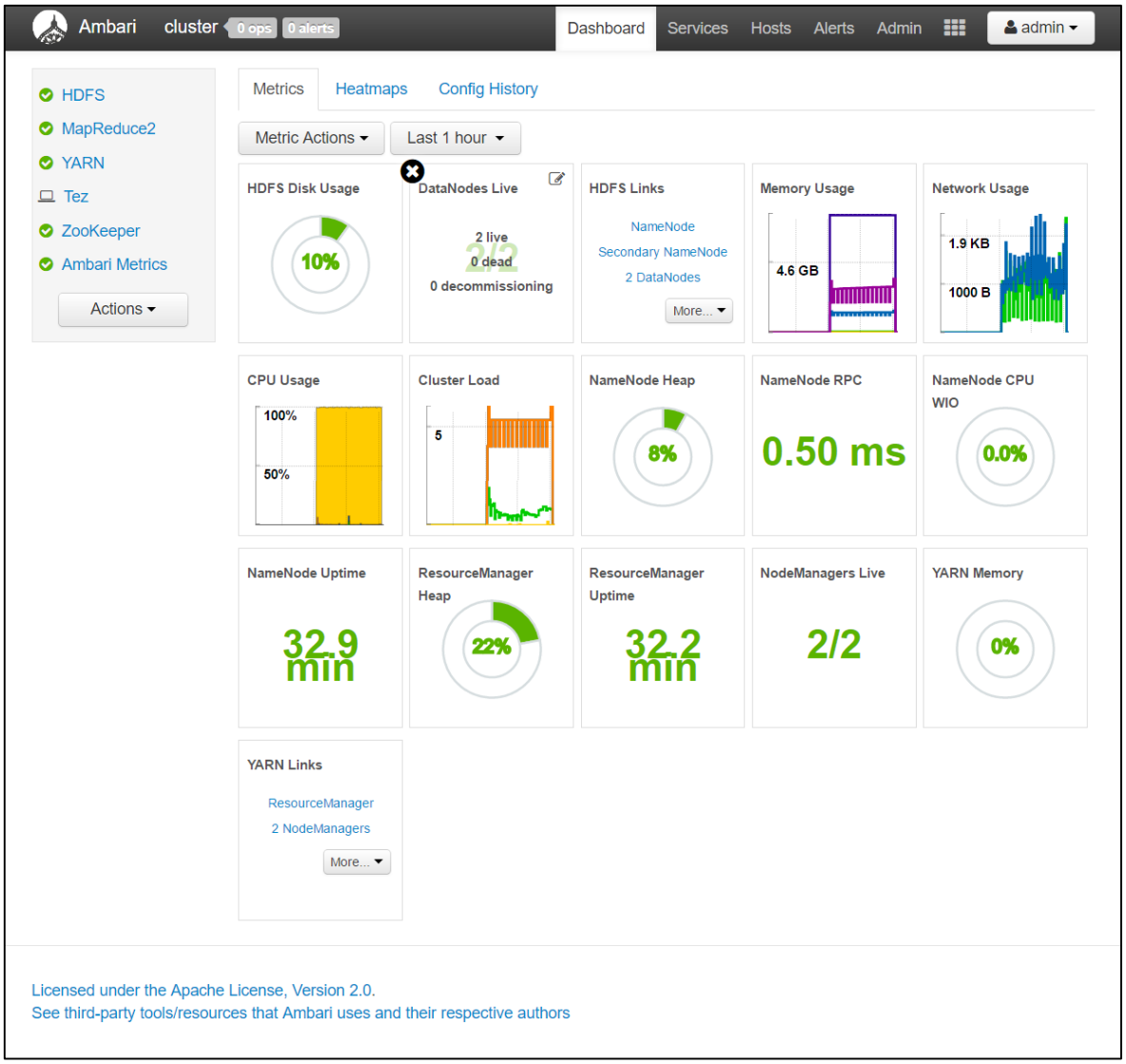


Şekil 4.13. Oluşturulan Hadoop kümesine ait genel bilgiler

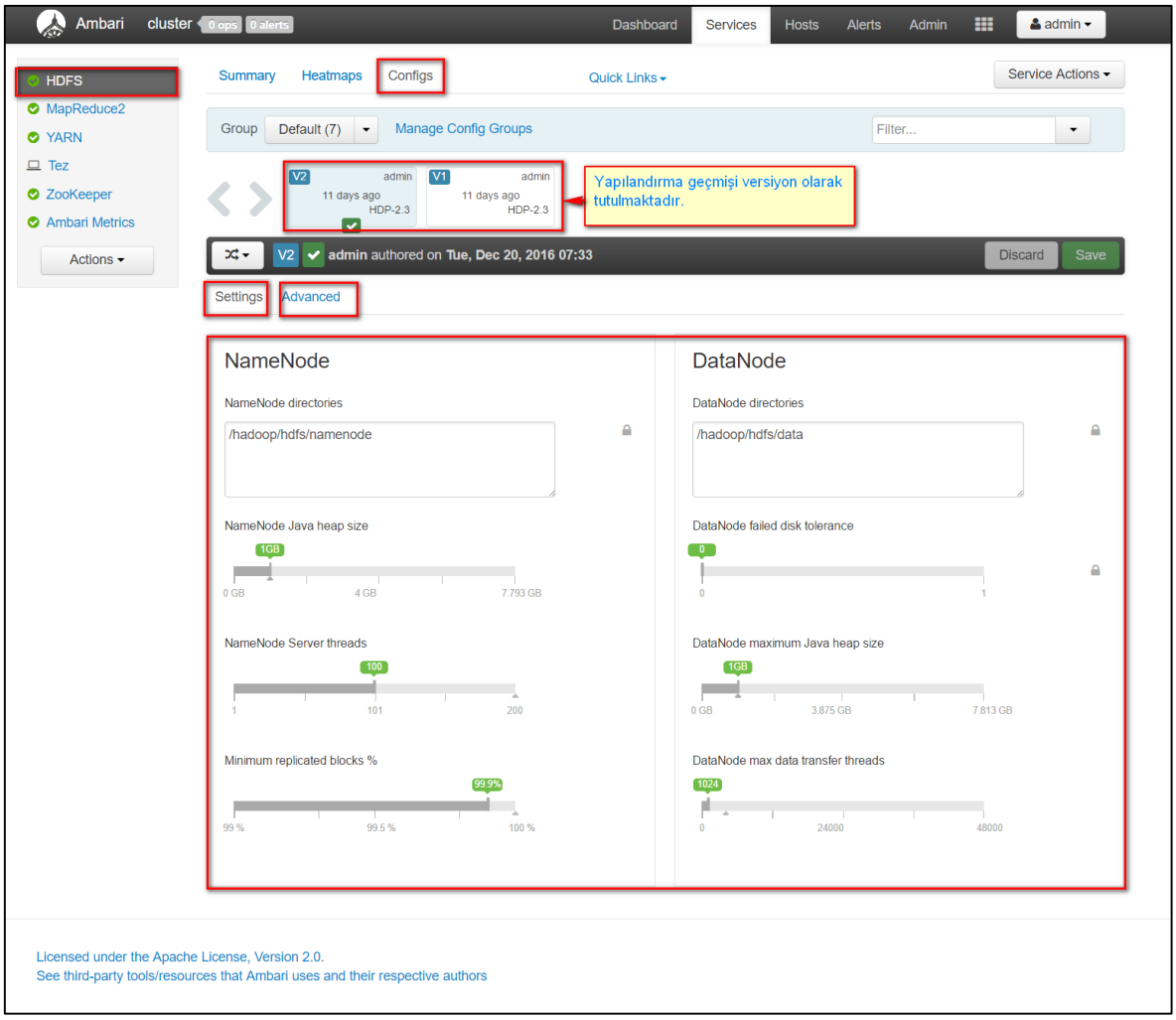


Şekil 4.14. Ambari giriş ekranı

Bu giriş ekranına Şekil 4.13'de yer alan Ambari kullanıcı adı ve şifre bilgileri ile giriş yapılır. Giriş yapıldıktan sonra Şekil 4.15'daki gibi kümeye ait genel metrikler izlenebilir. Şekilde Hadoop kümesi üzerine kurulu olan Hadoop servislerinin listesi ve durumları gözükmektedir.

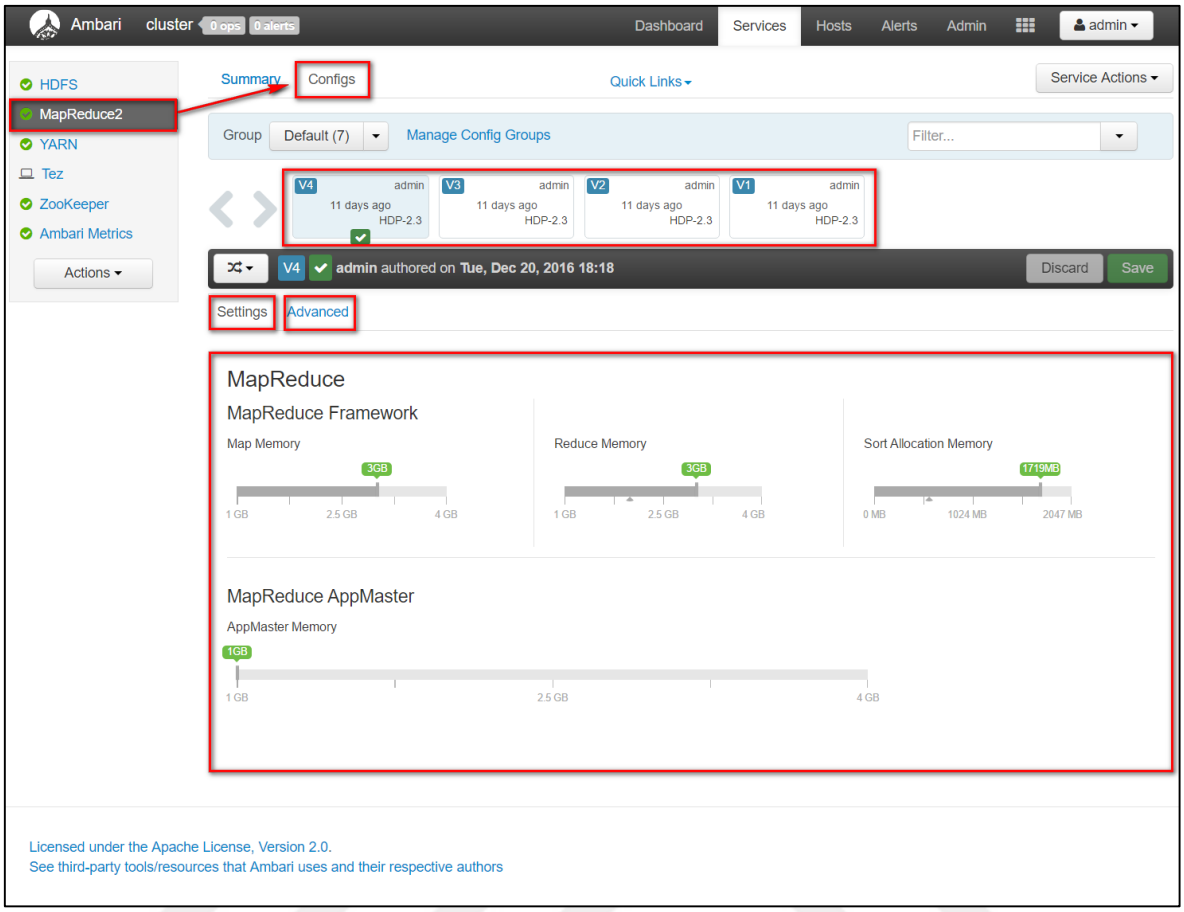


Şekil 4.15. Ambari web arayüzü



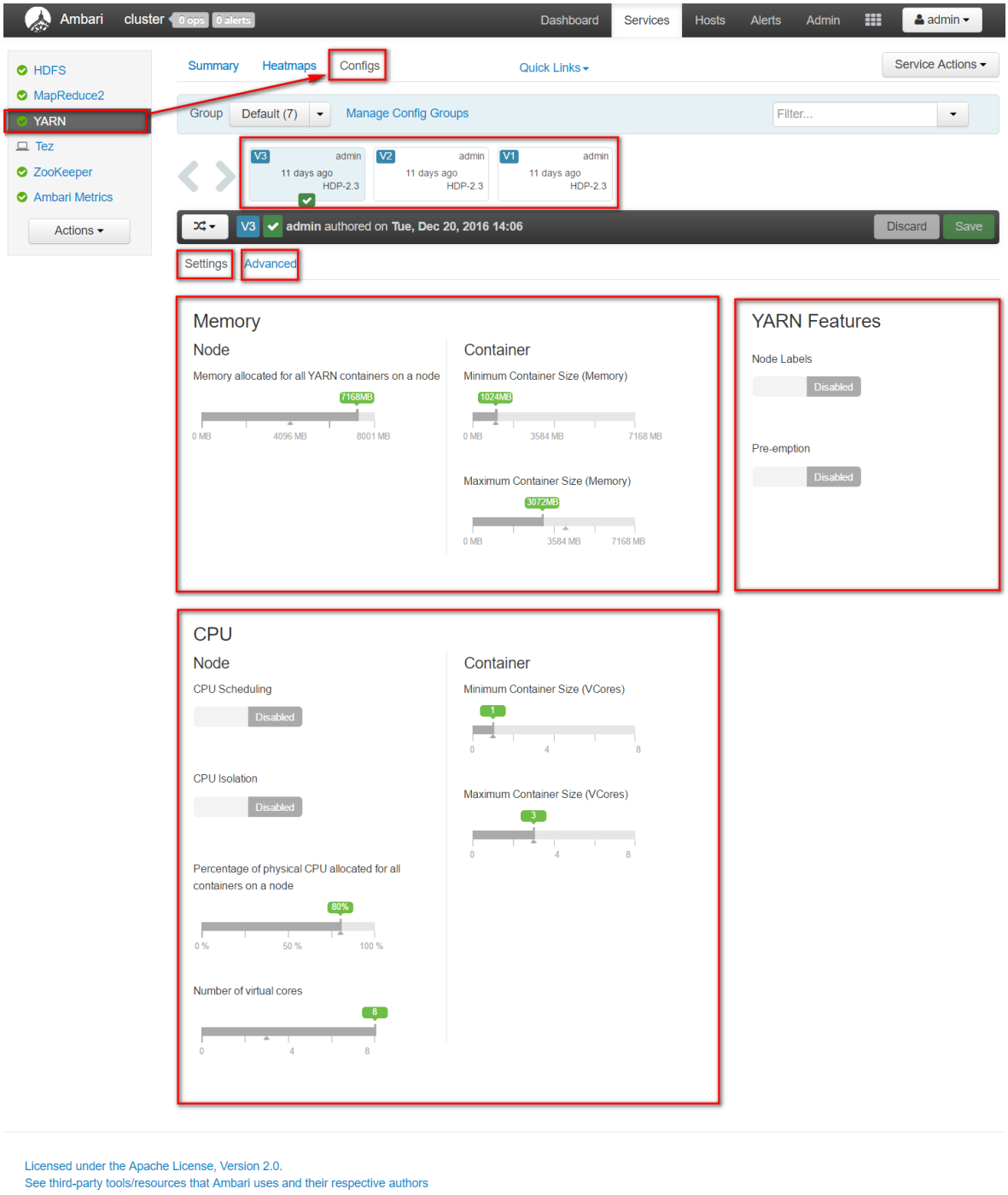
Şekil 4.16. Ambari ara yüzü ile HDFS' in yapılandırma ayarlarının yapılması

Şekil 4.16'da gösterildiği şekilde Ambari ara yüzü ile çok kolay bir şekilde HDFS yapılandırma ayarları yapılabilir. Yapılan ayarların versiyon geçmişi tutulduğundan istendiği zaman istenilen ayarlara kolay bir şekilde geçilebilmektedir. HDFS sisteminde bulunan NameNode ve DataNode bileşenlerinin temel yapılandırma ayarları Şekil 4.16'daki gibi yapılmıştır. Detaylı ayarlar için "Advanced" sekmesi kullanılabilir.



Şekil 4.17. Ambari ara yüzü ile MapReduce2 yapılandırma ayarlarının yapılması

Şekil 4.17'de Ambari ara yüzü ile yapılan MapReduce2 yapılandırma ayarları gösterilmektedir. Bu yapılandırma ayarları dağıtık video işleme uygulaması için yapılan ayarlardır. Dağıtık video işleme uygulamasında her map işlemi için 3 GB hafıza tahsis edilmiştir. Map hafızası olarak 1 GB hafıza verildiğinde yetersiz hafıza alanı uyarısı alındığından bu değer 3 GB olarak ayarlanmıştır. Dağıtık görüntü işleme uygulamasında map işlemi daha az hafıza gerektirdiği için "Map Hafıza- Map Memory" değeri 1 GB olarak verilmiştir. Detaylı ayarlar için "Advanced" sekmesi kullanılmıştır. Yapılandırma ayarlarında yapılan güncellemelerin aktif olması için bu ayarları kullanan servislerin yeniden başlatılması gerekmektedir. Bu işlem de Ambari ara yüzü ile kolay bir şekilde gerçekleştirilebilir. Geliştirilen uygulamalar yürütülürken kullanılan temel yapılandırma ayarları sonraki bölümlerde tablo olarak verilmiştir. "Application Master" uygulamanın küme üzerinde yönetimini sağlayan servistir. Bu servis, 1 GB hafıza kullanabilecek şekilde ayarlanmıştır.



Şekil 4.18. Ambari ara yüzü ile YARN yapılandırma ayarlarının yapılması

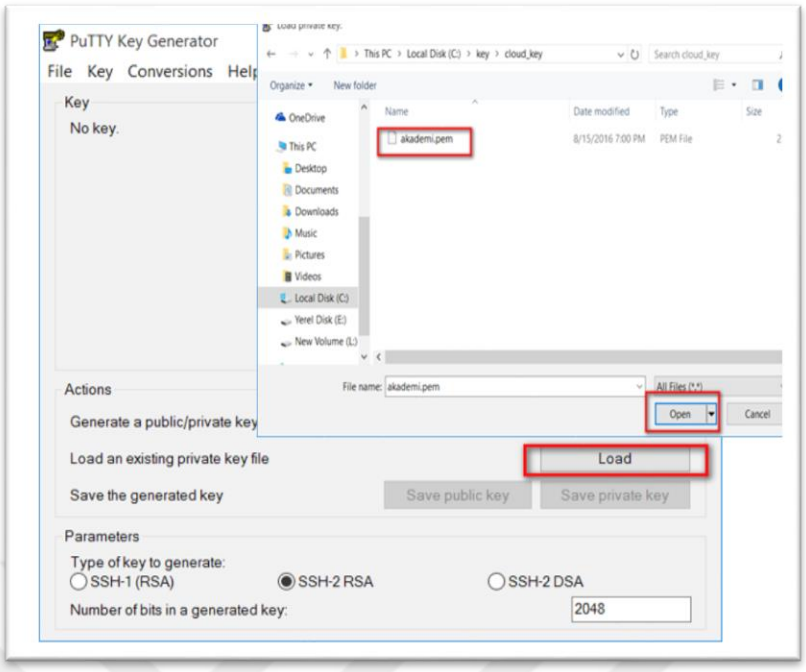
Şekil 4.18'de Ambari ara yüzü ile YARN yapılandırma ayarlarının yapılması gösterilmektedir. YARN kaynak yönetimini sağlamaktadır. YARN konteynır adı verilen bileşenler içinde map, reduce işlemleri ve Application Master çalışmaktadır. Konteynır kendisi için tanımlanan kaynakları kendi içinde çalışan uygulamalara kullandırır. Bu sayede konteynırlara kaynak atama işlemi ile uygulamalara özel kaynak ataması mümkün

olmaktadır. YARN yapılandırma ayarları ile bir düğüm de YARN konteynırlarının kullanabileceği toplam hafıza ve CPU kullanımı ve her bir konteynırın kullanabileceği en az ve en fazla hafıza ve CPU değerleri ayarlanabilmektedir. Bir düğümdeki YARN konteynırları tarafından kullanılabilecek en fazla hafıza değeri ataması ile YARN konteynırlarının düğümdeki bütün hafıza alanını kullanarak sistemi çalışamaz hale getirmesi engellenmektedir. Şekil 4.18'deki yapılandırma ayarlarında düğümün sahip olduğu 8 GB hafıza alanın 7 GB'lık alanın YARN konteynırları tarafından kullanılmasına izin verilmiştir. Geriye kalan 1 GB hafıza alan düğüm üzerinde çalışan diğer servislerin kullanımı için ayrılmıştır. Ayrıca YARN konteynırlarının düğümün sahip olduğu CPU kaynaklarının % 80'nini kullanabilmesine izin verilmiştir. Konteynır başına en az hafıza alanı 1 GB, en fazla hafıza alanı 3 GB olarak belirlenmiştir. Konteynır başına en az 1 vCPU en fazla 3 vCPU kullanılacak şekilde ayarlanmıştır. Bu ayarlamalar yapılırken MapReduce2 yapılandırma ayarları göz önünde bulundurulmuştur. MapReduce2 yapılandırma ayarlarında en fazla 1 GB hafıza kullanabilecek olan Application Master 1 GB hafıza alanına sahip konteynırlar üzerinde çalışacaktır. 3 GB olarak ayarlanan map fonksiyonları ise 3 GB hafıza alanına sahip konteynır üzerinde çalışacaktır. Konteynır hafıza alanı, üzerinde çalıştırılacak uygulamanın hafıza ihtiyacına göre yapılandırılmalıdır. Bu ayarın yürütülecek uygulamanın gereksinimlerine göre yapılması gerekir.

4.2. Oluşturulan Hadoop Kümesindeki Sanal Makinelere Erişim

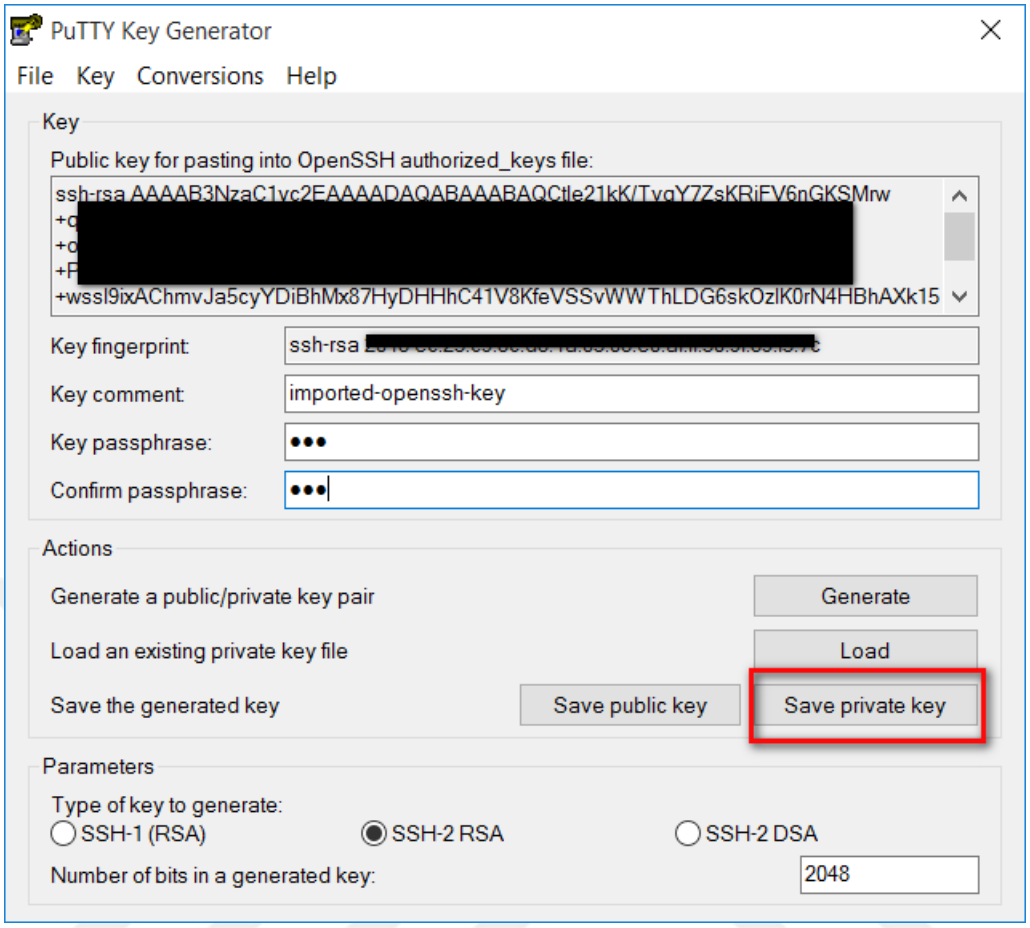
Oluşturulan Hadoop kümesindeki sanal bilgisayarlara erişmek için daha önce üretilen ve Hadoop kümesi oluşturulurken kullanılan "akademi" adında anahtar çiftinin akademi.pem adında sertifika dosyası kullanılmaktadır. Linux/Unix işletim sistemine sahip bir bilgisayarın komut satırına "`ssh -i akademi.pem cloud-user@10.1.57.1`" komutu yazılarak 10.1.57.1 ip adresine sahip usta (master) sanal makinesine ssh bağlantısı kurulabilir. Aynı şekilde işçi sanal makinelerine de IP adresleri yazılarak erişilebilir.

Windows işletim sistemine sahip bir bilgisayardan Hadoop kümesindeki sanal bilgisayarlara erişmek için PuTTY [97] terminal yazılımı kullanılabilir. PuTTY ile sanal makinelere erişmek için ilk olarak Şekil 4.19'da gösterildiği şekilde akademi.pem sertifika dosyası PuTTY Key Generator yazılımı ile açılır.



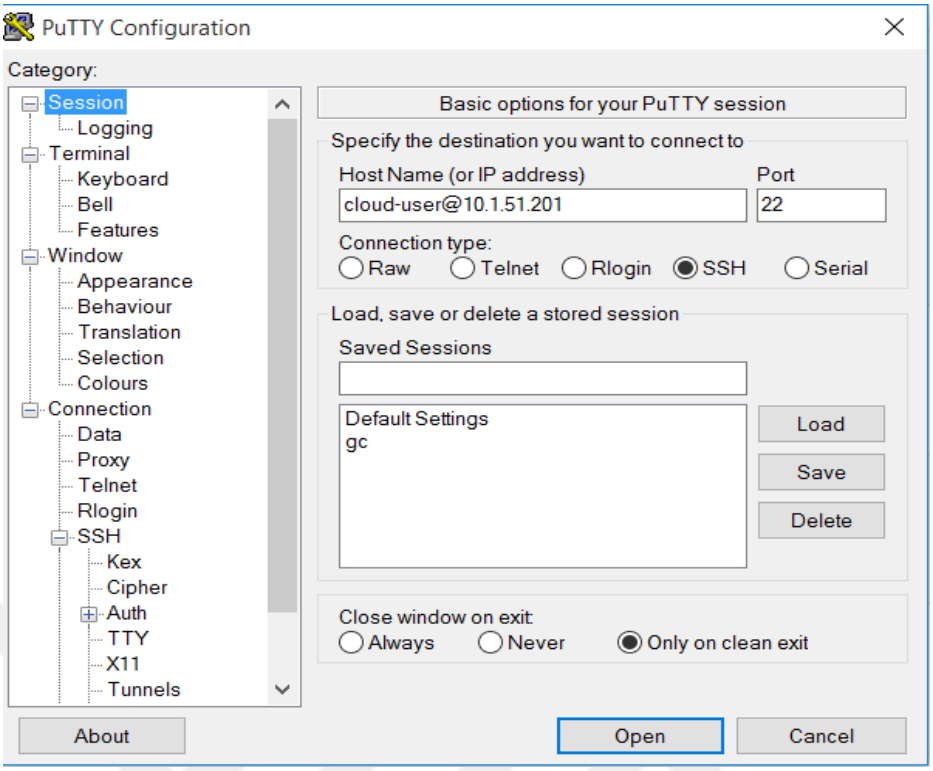
Şekil 4.19. Putty Key Generator ile akademi.pem dosyasının açılması

Şekil 4.20'de gösterildiği şekilde PuTTY Key Generator yazılımıyla açılan akademi.pem sertifika dosyasından private.ppk adında private anahtar oluşturulup kaydedilir.

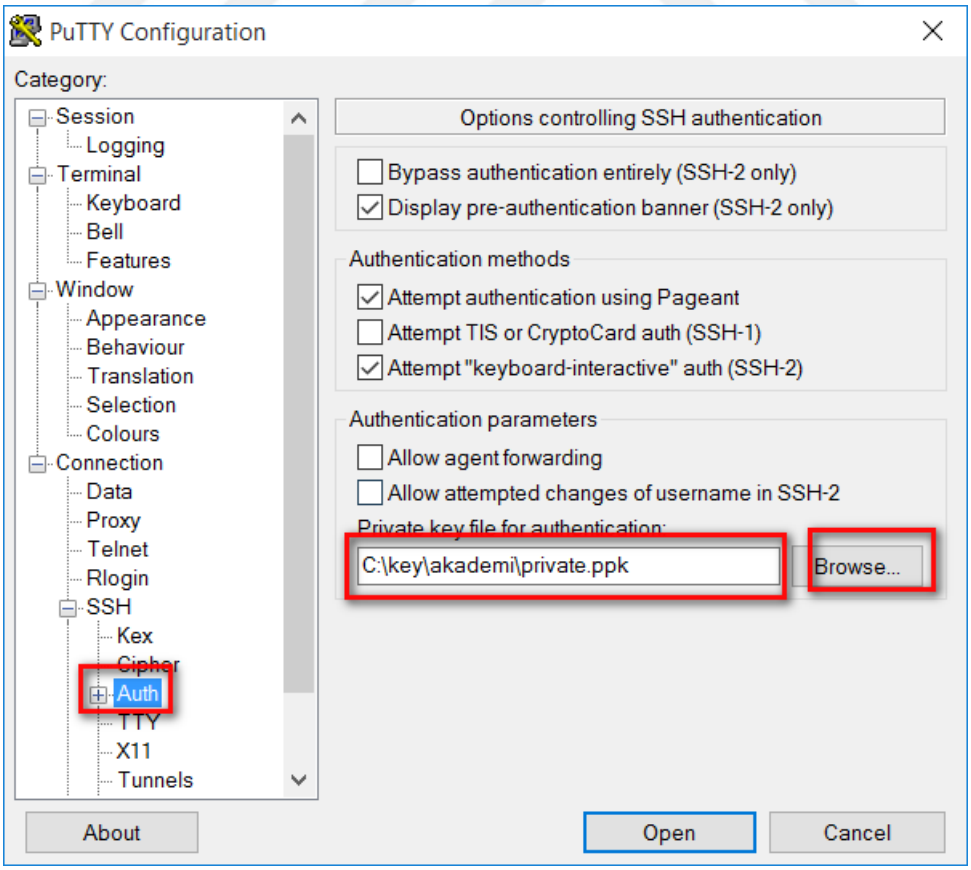


Şekil 4.20. PuTTY Key Generator ile private anahtar oluşturma

Oluşturulan private.ppk özel anahtar kullanılarak ilk olarak şekil 4.21'de gösterildiği gibi "Category" alanından "session" alanı tıklanarak hostname ve kullanıcı adı yazılır. Daha sonra Şekil 4.22'de gösterildiği gibi "Category" alanında ssh altında yer alan "Auth" seçeneği tıklanır. Şekilde gösterildiği gibi daha önce oluşturulan akademi.ppk özel anahtar seçilerek "Open" butonu tıklanarak ssh bağlantısı açılır.



Şekil 4.21. PuTTY ile sanal makiye erişme (1)



Şekil 4.22. PuTTY ile sanal makiye erişme (2)

5. UYGULAMALAR VE TESTLER

Bu tez çalışmasında görüntü ve video dosyalarının dağıtık olarak işlenmesi için dağıtık bir mimari kurulmuştur. Önceki bölümde kurulumu anlatılan Hadoop Kümesi kullanılarak, bu küme üzerinde MapReduce programlama modeli ile görüntü ve video dosyalarının işlenmesi için iki farklı uygulama geliştirilmiştir. Bu uygulamalar Apache Maven [98] projesi olarak Java programlama dili ile geliştirilmiştir.

Apache Maven, yazılım projeleri için kullanılan, yazılım projelerinin geliştirilmesini, derlenmesini, paketlenmesini, yönetimi kolaylaştıran açık kaynak kodlu ücretsiz bir proje yönetim aracıdır. Apache Maven projelerinde "Project Object Model (POM)" adında XML dosyası ile proje yönetimi gerçekleştirilir. Maven, IDE'ler arası geçişi ve kütüphane yönetimi gibi birçok kolaylık sağladığı için tercih edilmiştir. Bu sayede yazılan uygulamaların kaynak kodlarının, Maven desteği olan bütün IDE'lerde sorunsuz olarak açılabilir ve geliştirilebilir olması sağlanmıştır. Maven, projelere kütüphane ekleme ve kütüphaneleri yönetmede yazılım geliştiricilerine kolaylıklar sağlamaktadır. Yazılım geliştiricileri tarafından POM dosyasına, formatına uygun bir şekilde kütüphaneler ve kütüphanelerin indirileceği depolar tanımlar. Kullanılmak istenen kütüphaneler ve kütüphanelerin indirileceği depolar tanımlandıktan sonra, maven otomatik olarak depolardan kütüphaneleri indirir ve projeye ekler. Bu sayede proje kopyalanırken veya taşınırken ayrıca kullanılan kütüphanelerin taşınmasına ve kopyalanmasına gerek kalmaz. POM dosyasındaki tanımlara göre kütüphaneler otomatik olarak indirilir ve projeye entegre edilir.

Java Maven projesi olarak geliştirilen bu uygulamalardan ilki görüntülerin dağıtık olarak Hadoop kümesi üzerinde MapReduce programlama modeline göre işlenmesini sağlamak için geliştirilmiştir. Bu uygulamanın geliştirilmesinde görüntü işleme kütüphanesi olarak önceki bölümlerde anlatılan OpenIMAJ kütüphanesi kullanılmıştır. Bu projede Hadoop MapReduce kullanılarak görüntü dosyaları dağıtık olarak işlenmiş dosyalarındaki yüzler bulunarak kaydedilmiştir. Geliştirilen bu uygulama, 12 GB boyutunda olan "People in Photo Album (PIPA) dataset'inde [99] yer alan görüntüler ile test edilmiştir. Hadoop kümesi yatay olarak ölçeklendirilerek performans testleri yapılmıştır. Bu uygulamanın geliştirilmesi ve aşamaları, elde edilen sonuçlar aşağıda ayrıntılı olarak anlatılacaktır.

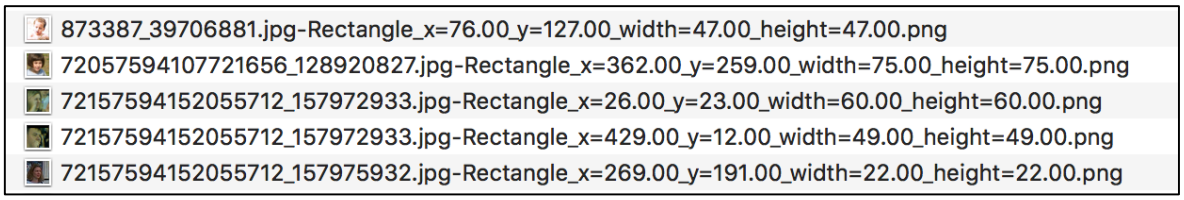
Java Maven projesi olarak geliştirilen ikinci uygulama ise video dosyalarının Hadoop kümesi üzerinde MapReduce programlama modeline göre işlenmesi işlemini gerçekleştiren

uygulamadır. Bu uygulamada video dosyaları çerçeve çerçeve okunarak çerçeveler arası değişimler hesaplanmakta ve sahne değişimleri bulunmaktadır. Sahne değişimlerindeki son anahtar çerçeve alınıp png uzantılı görüntü dosyası olarak kaydedilmektedir. Her video dosyası için video dosyasındaki her sahneye ait bir çerçeve tutulmaktadır. Bu çerçeveler videonun bir özetini oluşturmaktadır. Video dosyasının okunması işlenmesi Hadoop kütüphanesinin sağladığı standart sınıflar ile yapılamadığı için videoların okunması ve işlenmesi için gerekli olan Hadoop sınıfları yazılmıştır. Bu uygulama da oluşturulan Hadoop kümesi üzerinde test edilmiş, küme yatay olarak ölçeklendirilerek performans analizleri yapılmıştır. Bu uygulamanın geliştirilmesi ve aşamaları, elde edilen sonuçlar aşağıda ayrıntılı olarak anlatılacaktır.

5.1. Dağıtık Görüntü İşleme Uygulaması (Görüntü Dosyalarından Dağıtık Görüntü İşleme ile Yüz Tespiti)

Hadoop ile dağıtık görüntü işleme işlemini gerçekleştirmek için dağıtık görüntü işlemenin yapılacağı Hadoop kümesine ve Hadoop kümesi üzerinde çalıştırılacak uygulamaya ihtiyaç vardır. Hadoop Kümesi kurulumu yukarda anlatılmış olup, bu bölümde Hadoop kümesi üzerinde çalıştırılacak MapReduce programlama modeline göre geliştirilmiş Dağıtık Görüntü İşleme uygulaması anlatılacaktık.

Geliştirilen uygulama, görüntü dosyalarını dağıtık olarak işleyerek görüntü dosyalarındaki kişilerin yüzlerini tespit etmektedir. Tespit edilen yüzleri, yeni bir görüntü dosyası olarak HDFS ortamına kaydetmektedir. Oluşturulan bu görüntü dosyaları, Şekil 5.1' de gösterildiği gibi yüzün tespit edildiği orijinal görüntü bilgisi ve yüzün görüntü dosyasında bulunduğu konum bilgisi kullanılarak adlandırılmaktadır.



Şekil 5.1. Yüz görüntü dosyalarının adlandırılması

Uygulama 12 GB boyutunda olan "People in Photo Album (PIPA) [99, 100]" data setinde yer alan görüntüler ile test edilmiştir. PIPA data setinde 37107 adet fotoğraf bulunmaktadır. Bu data set yaklaşık 2000 kişinin herkese açık olan Flickr fotoğraf albümlerinden

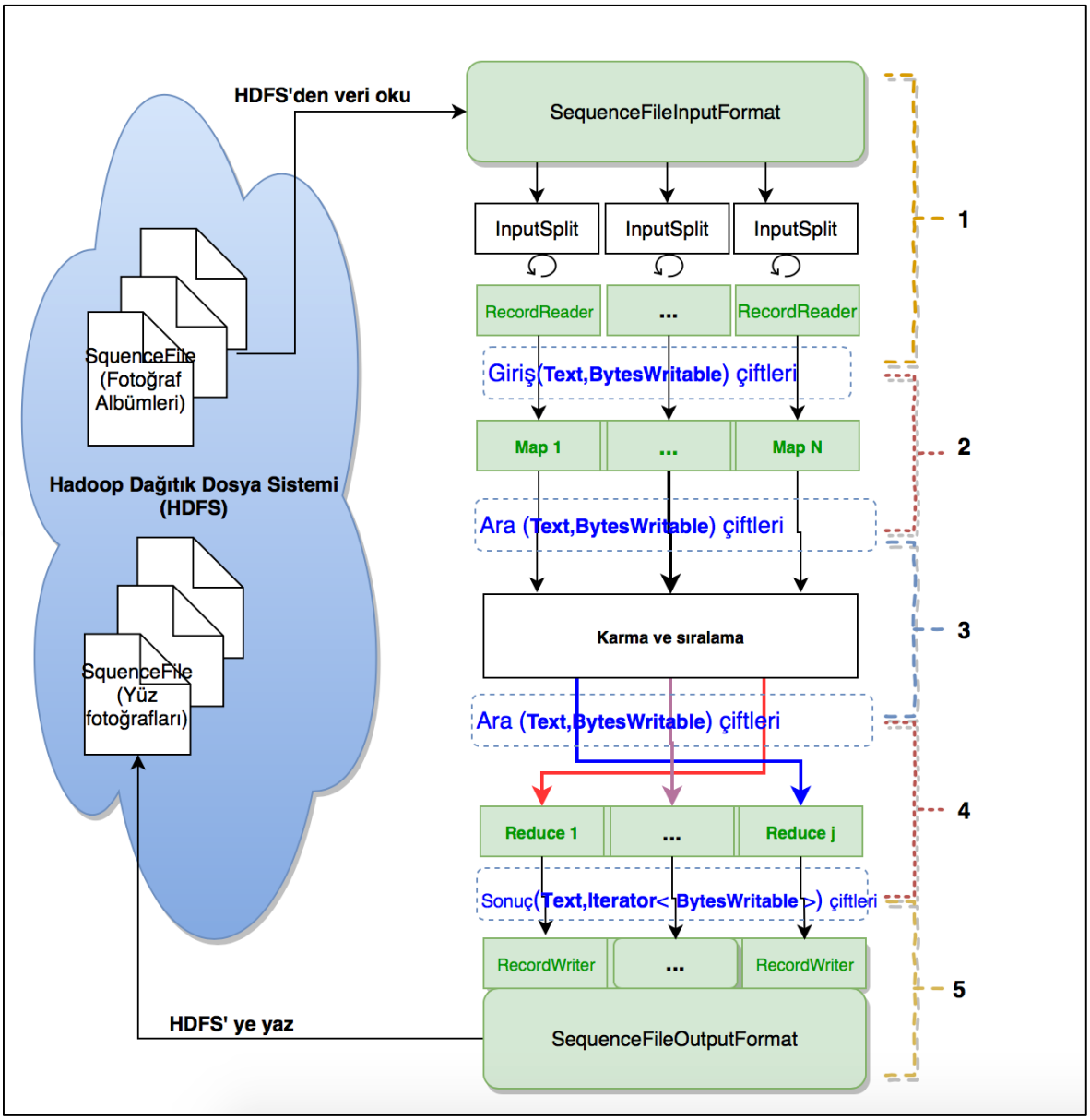
oluşmaktadır [99, 100]. Geliştirilen Hadoop MapReduce uygulaması ile bu görüntü dosyalarından yüz tespiti yapılmıştır. Yüzleri bulmak için OpenIMAJ kütüphanesinin HaarCascade sınıfı kullanılmıştır.

Bu uygulama ile amaçlanan OpenIMAJ kütüphanesinin HaarCascade algoritmasının doğruluğunu test edilmesi değildir. Dağıtık görüntü işleme alanında kompleks işlerden biri olan yüz bulma algoritması özelde kullanılarak, oluşturulan Hadoop kümesi üzerinde görüntü işleme işinin gerçekleştirilmesi ve ölçeklendirme yaparak kurulan Hadoop MapReduce mimarisinin test edilmesidir. Dağıtık görüntü işleme mimarisi kurulduktan sonra özelde gerçekleştirilen yüz bulma algoritması yerine OpenIMAJ kütüphanesinin sağladığı diğer algoritma ve yöntemler beraber veya ayrı ayrı kullanılarak problem özelinde farklı çözümlerin geliştirilmesi mümkün olacaktır. Burada asıl amaç daha önce bahsedildiği gibi dağıtık görüntü işleme için bir sistem kurulması ve bu sisteme uygun geliştirilen yazılım ile sistemin test edilmesidir.

Bu tez çalışmasında, kurulan Hadoop kümesinin üzerinde çalıştırılacak MapReduce programlama modeli kullanılarak dağıtık görüntü işleme işlemi gerçekleştirilmiştir. Bunun için PIPA data seti HDFS ortamına kaydedilmiştir. HDFS mimarisi büyük boyutlu dosyalarla çalışmak için geliştirildiği için çok sayıda küçük dosyalardan oluşan dosyalarının HDFS ortamında tutulup işlenmesinde performans kayıplarının olduğu yapılan çalışmalarda gösterilmiştir [60, 67, 68]. Çok sayıda küçük boyutlu (dosya boyutu< 64 MB ve HDFS blok boyutu >=64 MB)görüntü dosyasının daha performanslı bir şekilde işlenebilmesi için HIPI [92] kütüphanesi geliştirilmiştir. Bu kütüphane daha sonra OpenCV kütüphanesi ile entegre edilmiştir. Diğer bir çözüm ise görüntü dosyalarını birleştirerek Hadoop'un sağladığı SequenceFile formatına dönüştürmektir. Her iki yöntemde de asıl amaç küçük boyutlardaki çok sayıda görüntüyü birleştirip büyük boyutlu dosyalar elde etmektir. Bu tez çalışmasında bu yöntemlerden SequenceFile yöntemi tercih edilmiştir. SequenceFile dosya formatının tercih edilmesinde OpenIMAJ kütüphanesinin tercih edilmiş olması etkilidir. OpenIMAJ kütüphanesi, Java ile yazılmış kütüphane olması, kurulum gerektirmemesi gibi daha önce bahsedilen nedenlerden dolayı seçilmiştir.

Şekil 5.2'de gerçekleştirilen dağıtık görüntü işleme uygulamasının adımları gösterilmektedir. İlk önce işlenmek istenen görüntü dosyaları SequenceFile formatına dönüştürülür. Daha sonra oluşturulan bu SequenceFile formatındaki dosya HDFS ortamına yazılır. Bu işlem bittikten sonra 1 numaralı aşamada HDFS ortamında bulunan SequenceFile dosyaları SequenceFileInputFormat sınıfı ile okunur ve Text key, BytesWritable value

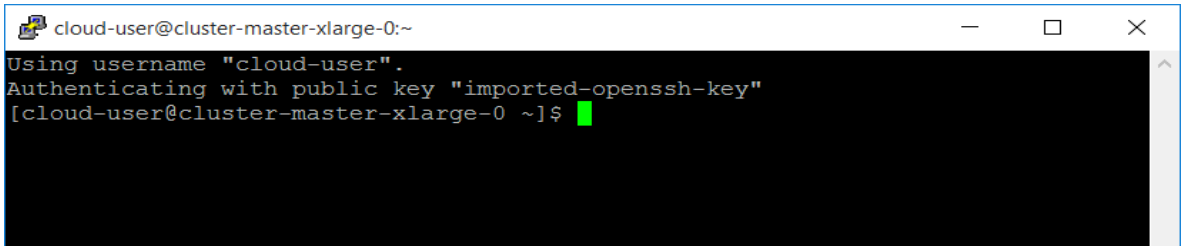
anahtar değer çiftleri oluşturularak map metotlarına dağıtılır. 2 numaralı aşamada Map metodunda BytesWritable değerleri OpenIMAJ görüntü formatı olan MBFImage görüntü formatına dönüştürülerek bu görüntüler için OpenIMAJ HaarCascadeDetector sınıfı uygulanarak yüzler tespit edilmiştir. Sistem 3 ve 4 numaralı aşamaları gerçekleştirebilecek şekilde kurulmuştur fakat yüz bulma işlemi map metodunda tamamlandığı için bu uygulamada 3 ve 4 numaralı aşamalar kullanılmamıştır. 3 numaralı aşamada anahtar verisine göre görüntü verileri gruplanarak 4 numaralı reduce metoduna verilmektedir. Bu aşamada bulunan yüzlerin filtrelenmesi gibi işlemler uygulanabilir. Bu uygulamada bu aşamalara ihtiyaç olmadığı için kullanılmamıştır. 5 numaralı aşamada bulunan yüzler ayrı görüntü dosyaları olarak HDFS'e yazılmaktadır. 3 ve 4 numaralı aşamalar kullanılmadığından map metodu tarafından bulunan yüzler 5 numaralı aşamada SquenceFileOutputFormat sınıfı ile HDFS ortamına yazılmıştır. Bu uygulamada Hadoop kütüphanesinin sağladığı SequenceFileInput, SquenceFileOutputFormat, Text, BytesWritable gibi standart sınıflar kullanılmıştır. Sonraki uygulamada bu sınıflar yerine bu tez çalışması için yazılan sınıflar kullanılmıştır.



Şekil 5.2. Dağıtık görüntü işleme adımları

Yukarıda anlatılan uygulama aşağıdaki adımlar uygulanarak Hadoop kümesi üzerinde çalıştırılmıştır.

1- Daha önce anlatıldığı şekilde Hadoop kümesi oluşturulduktan sonra ve anlatılan erişim yöntemlerinden biri ile usta sanal makinesine erişilir. PuTTY yazılımı ile erişildiğinde Şekil 5.3'de gösterilen bir çıktı elde edilir.



Şekil 5.3. PuTTY ile usta sanal makinesi erişilmesi

2- "hdfs" kullanıcısı ile giriş yapılır ve hdfs kullanıcısının ana dizinine gidilir.

```
[cloud-user@cluster-master-large-0 ~]$ sudo su hdfs
[hdfs@cluster-master-large-0 cloud-user]$ cd
```

3- PIPA data seti wget komutu ile aşağıdaki şekilde indirilmektedir.

```
[hdfs@cluster-master-large-0 ~]$ wget
https://people.eecs.berkeley.edu/~nzhang/datasets/pipa_test.tar
[hdfs@cluster-master-large-0 ~]$ wget
https://people.eecs.berkeley.edu/~nzhang/datasets/pipa_train.tar
[hdfs@cluster-master-large-0 ~]$ wget
https://people.eecs.berkeley.edu/~nzhang/datasets/pipa_val.tar
[hdfs@cluster-master-large-0 ~]$ wget
https://people.eecs.berkeley.edu/~nzhang/datasets/pipa_leftover.tar
```

Tar dosyası olarak indirilen PIPA data setinin bilgileri ls –lh komutu ile listelenmiştir.

```
[hdfs@cluster-master-large-0 ~]$ ls -lh
total 12G
-rw-r--r--. 1 hdfs hadoop 2.1G Feb  6  2015 pipa_leftover.tar
-rw-r--r--. 1 hdfs hadoop 2.8G Feb  5  2015 pipa_test.tar
-rw-r--r--. 1 hdfs hadoop 5.3G Feb  5  2015 pipa_train.tar
-rw-r--r--. 1 hdfs hadoop 1.8G Feb  5  2015 pipa_val.tar
```

4- Tar dosya formatında olan PIPA data seti "pipa_exracted_files" dizini oluşturularak bu dizine aşağıdaki şekilde çıkarılmaktadır.

```
[hdfs@cluster-master-large-0 ~]$ mkdir -p pipa/pipa_tar_files
[hdfs@cluster-master-large-0 ~]$ mkdir -p pipa/pipa_exracted_files
[hdfs@cluster-master-large-0 ~]$ mv pipa_* pipa/pipa_tar_files
[hdfs@cluster-master-large-0 ~]$ cd pipa/pipa_tar_files
[hdfs@cluster-master-large-0 pipa_tar_files]$ for i in *.tar; do echo working on $i; tar -xvf
$i -C ../pipa_exracted_files ; done
```

12 GB boyutundaki tar dosyalarında çıkarıldığında 13 GB boyutuna ulaşmaktadır. Aşağıdaki komutlar ile bu bilgiler elde edilmiştir.

```
[hdfs@cluster-master-large-0 pipa_tar_files]$ cd ..
[hdfs@cluster-master-large-0 pipa]$ du -sh *
13G     pipa_exracted_files
12G     pipa_tar_files
```

5- Açılan tar dosyalarında nokta ile başlayan dosyalarla beraber 74214 dosya bulunmaktadır. Nokta ile başlayan dosyalar silindiğinde 37107 görüntü dosyası kalmaktadır. Bu işlemler sırasıyla aşağıdaki şekilde yapılmaktadır.

```
[hdfs@cluster-master-large-0 ~]$ find pipa/pipa_exracted_files/ -type f | wc -l
74214
[hdfs@cluster-master-large-0 ~]$  find pipa/pipa_exracted_files/. -iname '._*' -exec rm -rf
{} \;
[hdfs@cluster-master-large-0 ~]$ find pipa/pipa_exracted_files/ -type f | wc -l
37107
```

6- 37107 dosyadan oluşan PIPA data setindeki görüntü dosyaları OpenIMAJ kütüphanesi ile Hadoop SequenceFile için geliştirilen SequenceFileTool [101] aracı ile SequenceFile formatına dönüştürülmüştür. SequenceFileTool açık kaynak kodlu bir projedir. Bu projenin kaynak kodları indirilerek tekrar derlenerek jar dosyası olarak paketlenmiştir. Bu jar dosyasının bulunduğu bilgisayara ssh ile bağlanılmaktadır. Bu işlemin gerçekleştirilebilmesi için 10.1.43.3 IP'li bilgisayarın ssh ile uzaktan erişime açık olması gerekmektedir. SequenceFileTool.jar dosyası aynı ağda bulunan 10.1.43.3 IP'li bilgisayardan Hadoop kümesinin usta bilgisayarına aşağıdaki şekilde

kopyalanmaktadır. Bağlantı için istenen şifre bilgisi olarak murattezgider kullanıcısının şifresi girilmiştir.

```
[hdfs@cluster-master-large-0 ~]$ scp
murattezgider@10.1.43.3:/Users/murattezgider/NetBeansProjects/tez/SequenceFileTool/tar
get/SequenceFileTool.jar SequenceFileTool.jar
```

7- SequenceFileTool.jar aracı aşağıdaki şekilde kullanılarak PIPA data setindeki görüntü dosyaları "pipa_all.seq" adında SequenceFile formatına dönüştürülmektedir.

```
[hdfs@cluster-master-large-0 ~]$ java -jar SequenceFileTool.jar -m CREATE
pipa/pipa_exracted_files/ -R -kns RELATIVEPATH -o pipa_all.seq
```

8- Daha sonra oluşturulan "pipa_all.seq" dosyası HDFS ortamında mapred/app/dgi/input/pipa dizini oluşturularak –copyFromLocal komutu ile "pipa_all.seq" dosyası HDFS mapred/app/dgi/input/pipa dizinine aşağıdaki şekilde kopyalanmaktadır

```
[hdfs@cluster-master-large-0 ~]$ hadoop fs -mkdir -p /mapred/app/dgi/input/pipa
[hdfs@cluster-master-large-0 ~]$ hadoop fs –copyFromLocal pipa_all.seq
/mapred/app/dgi/input/pipa
```

9- Geliştirilen dağıtık görüntü işleme uygulaması usta sanal bilgisayarına aşağıdaki şekilde kopyalanır. Geliştirilen uygulama önce daha 10.1.43.3 IP'li bilgisayarda dgi.jar olarak paketlenmiştir.

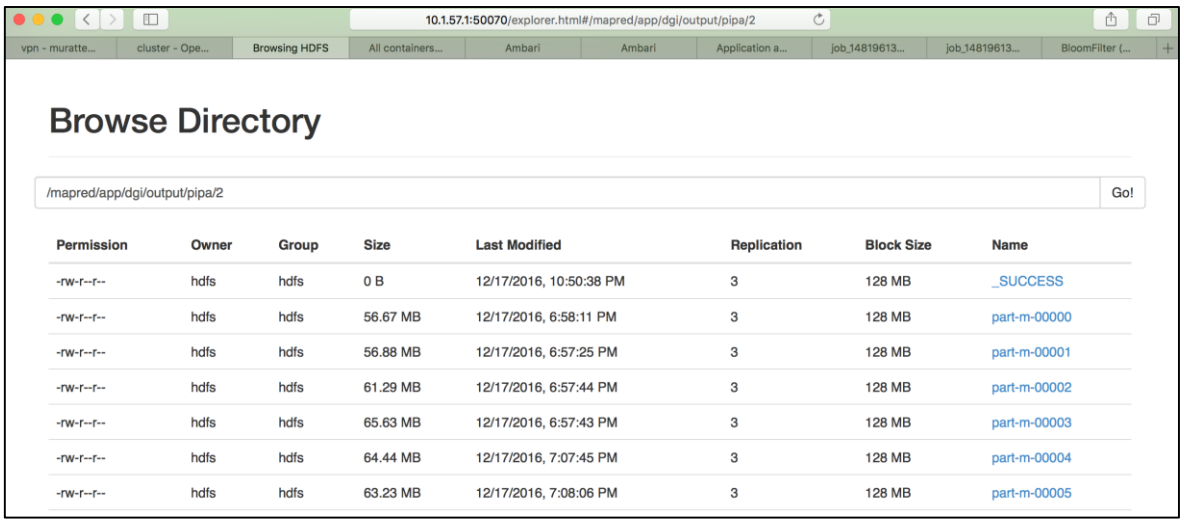
```
[hdfs@cluster-master-large-0 ~]$ scp
murattezgider@10.1.43.3:/Users/murattezgider/NetBeansProjects/tez/hadoop/target/dgi.jar
dgi.jar
```

10- Usta makineye kopyalanan uygulama aşağıdaki şekilde giriş ve çıkış dizinleri verilerek çalıştırılır.

```
[hdfs@cluster-master-large-0 ~]$ hadoop jar dgi.jar -i
hdfs://10.1.57.1/mapred/app/dgi/input/pipa/pipa_all.seq -o
hdfs://10.1.57.1/mapred/app/dgi/output/pipa/2
```

Şekil 5.4'de PIPA data setinde bulunan fotoğraf albümünden iki örnek gösterilmiştir. Dağıtık görüntü işleme uygulamasının yürütülmesi tamamlandığında Şekil 5.5'de gösterilen SequenceFile formatındaki yüz fotoğraflarını içeren dosyalar oluşmaktadır. Bu dosyalardan "part-m-00000" dosyasında bulunan yüz görüntüleri Şekil 5.6'de gösterilmektedir.

Şekil 5.4. PIPA data setinde bulunan görüntü örnekleri



Şekil 5.5. Dağıtık görüntü işleme uygulaması tarafından oluşturulan yüz dosyalarının yer aldığı SequenceFile dosyaları

Şekil 5.6. Dağıtık görüntü işleme algoritması ile tespit edilen yüzler

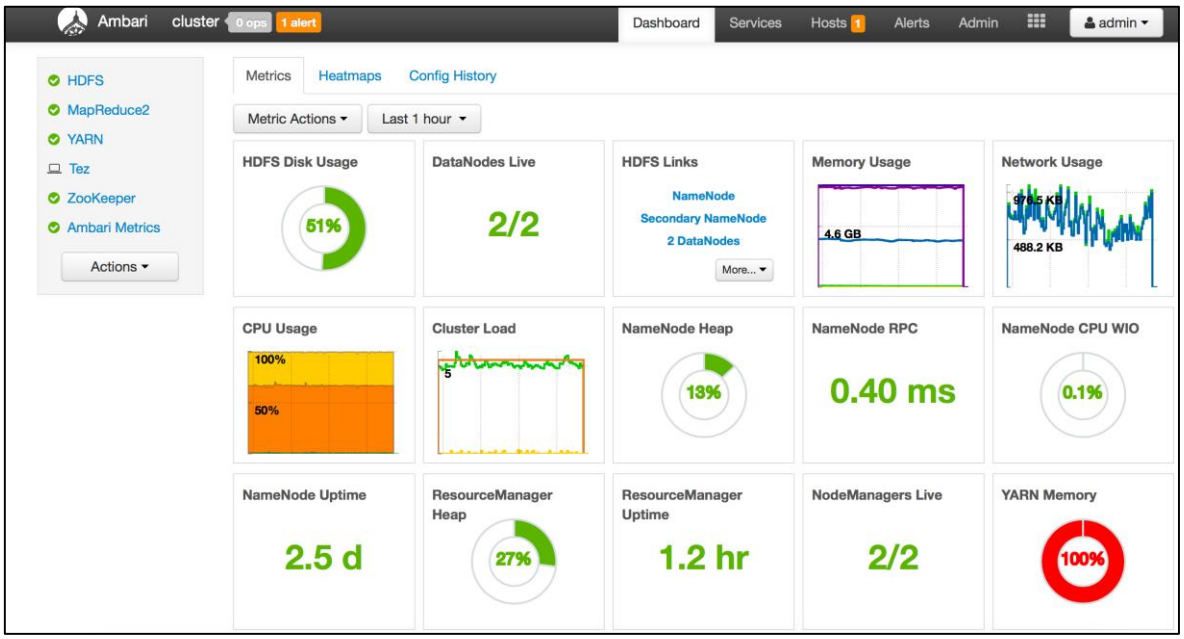
5.2. Dağıtık Görüntü İşleme Sistemi Değerlendirme Sonuçları

Dağıtık görüntü işleme uygulamasının çalıştırıldığı Hadoop kümesinin yapılandırma bilgileri Tablo 5.1'de gösterilmiştir. MapReduce uygulamalarının çalıştırılacağı Hadoop kümesinin doğru yapılandırılması, kaynakların uygulama tarafından verimli kullanılması açısından oldukça önemlidir. Bir uygulamaya ihtiyaç duyulandan daha az veya daha fazla kaynak atanmamalıdır. Az kaynak atandığında uygulamanın yürütülmesinde problemler oluşur veya problem oluşmasa bile ortalama map işlem süresi artar. Fazla kaynak atandığında başka bir konteyner tarafından kullanılabilecek kaynağın boş yere rezerve edilmesine ve kullanılmamasına sebep olunur.

Tablo 5.1. Görüntü işleme uygulaması için yapılan yapılandırma

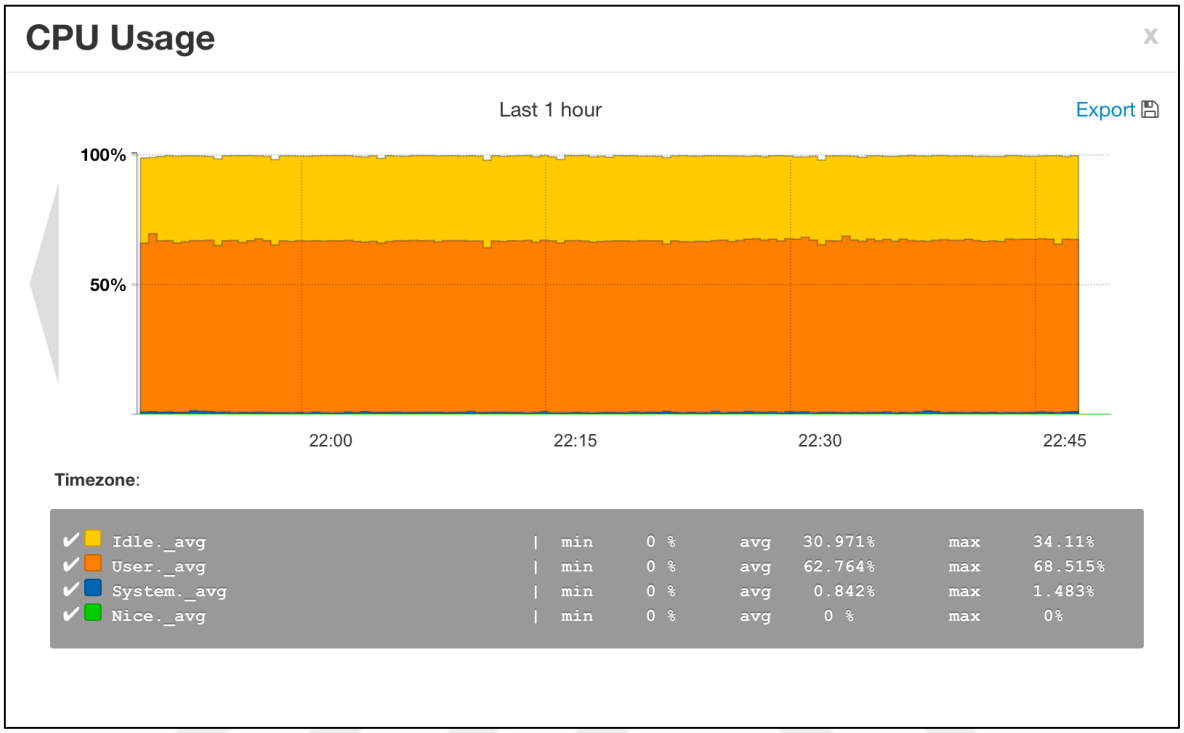
YARN HAFIZA YAPILANDIRMASI			
Sanal Makine RAM	**Bütün YARN Konteynırlarının Kullanabileceği RAM**	**YARN Konteynır Minimum RAM**	**YARN Konteynır Maksimum RAM**
8 GB	7 GB	1 GB	4GB
YARN vCPU YAPILANDIRMASI			
Sanal Makine vCPU	**Bütün YARN Konteynırlarının Kullanabileceği vCPU**	**YARN Konteynır Minimum vCPU**	**YARN Konteynır Maksimum vCPU**
8 core	%80	1 core	3 core
MAPREDUCE YAPILANDIRMASI			
Map	**Reduce**	**Sort Allocation**	**Application Master**
1GB	3GB	1.5 GB	1 GB

Dağıtık görüntü işleme uygulaması yukardaki yapılandırma ile 1 usta 2 işçi sanal bilgisayar üzerinde çalışırken Ambari ara yüzünde gösterilen metrik değerleri Şekil 5.7'de gösterilmiştir.



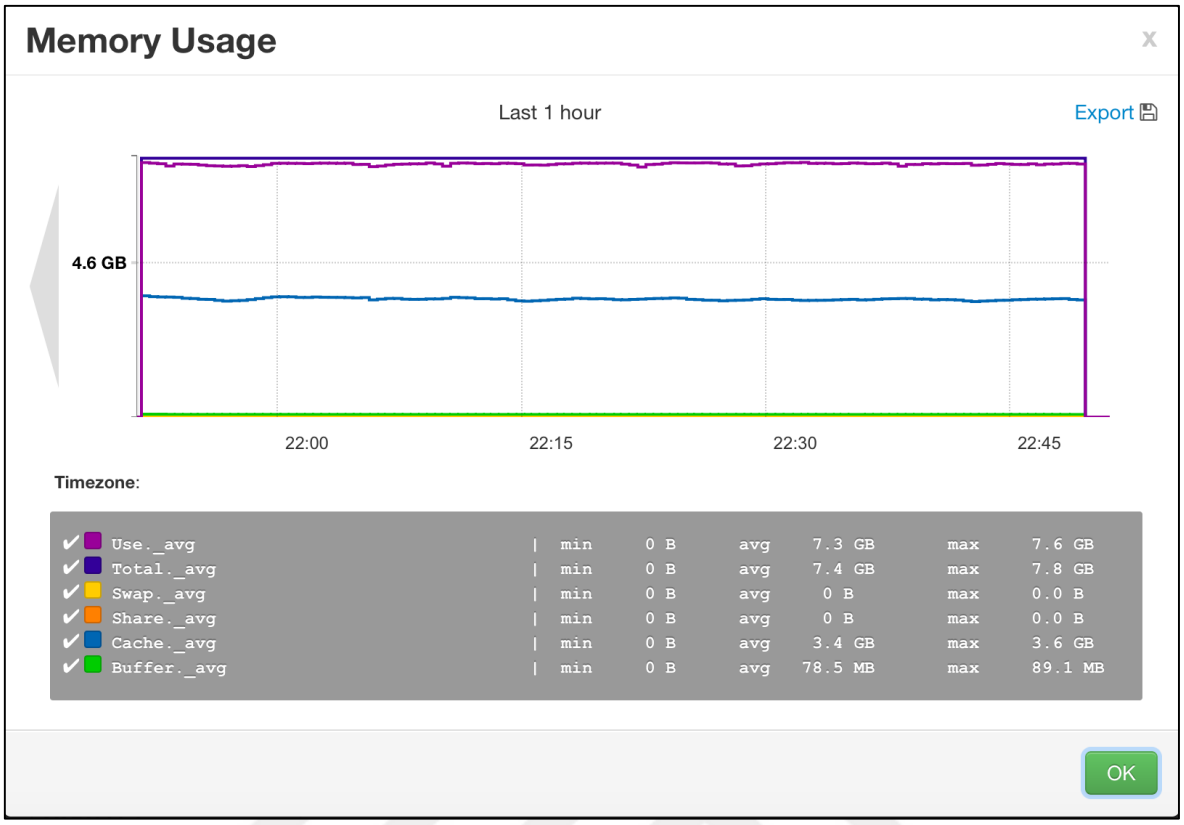
Şekil 5.7. Görüntü işleme uygulaması çalışırken Ambari metrikleri

Şekil 5.8’ de gösterildiği üzere görüntü işleme uygulaması 1 usta 2 işçi bilgisayardan oluşan Hadoop kümesi üzerinde çalışırken %62 oranında CPU kullanmaktadır.



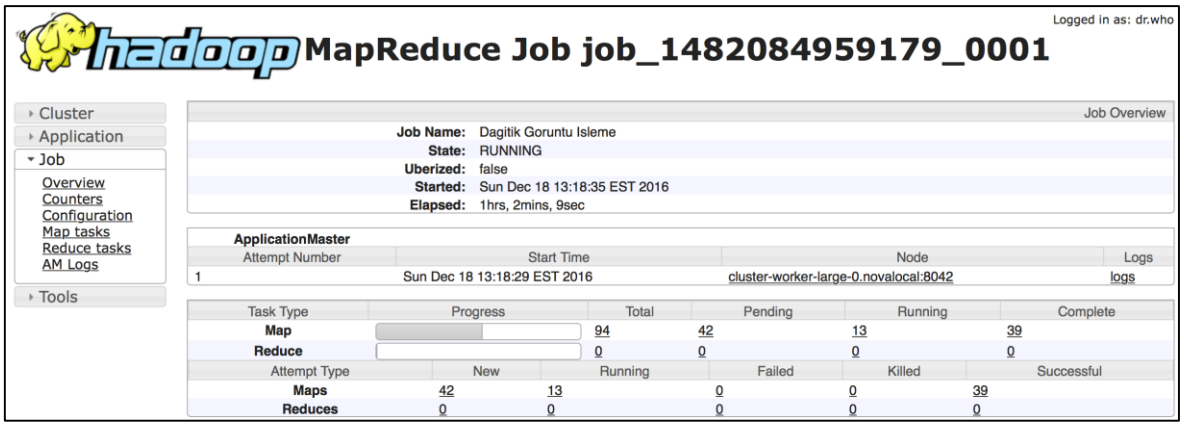
Şekil 5.8. Görüntü işleme uygulaması çalışırken CPU kullanım grafiği

Şekil 5.9’ da gösterildiği üzere görüntü işleme uygulaması 1 usta 2 işçi bilgisayardan oluşan Hadoop kümesi üzerinde çalışırken ortalama 7.3 GB RAM kullanmaktadır.



Şekil 5.9. Görüntü işleme uygulaması çalışırken hafıza kullanım grafiği

Hadoop, MapReduce uygulaması başlatıldığında uygulamanın takip edilmesini sağlamak için Şekil 5.10'daki gibi bir ara yüz sağlar. Bu ara yüz sayesinde uygulamanın hangi aşamada olduğu yürütülen ve kalan işlerin neler olduğu izlenebilir.

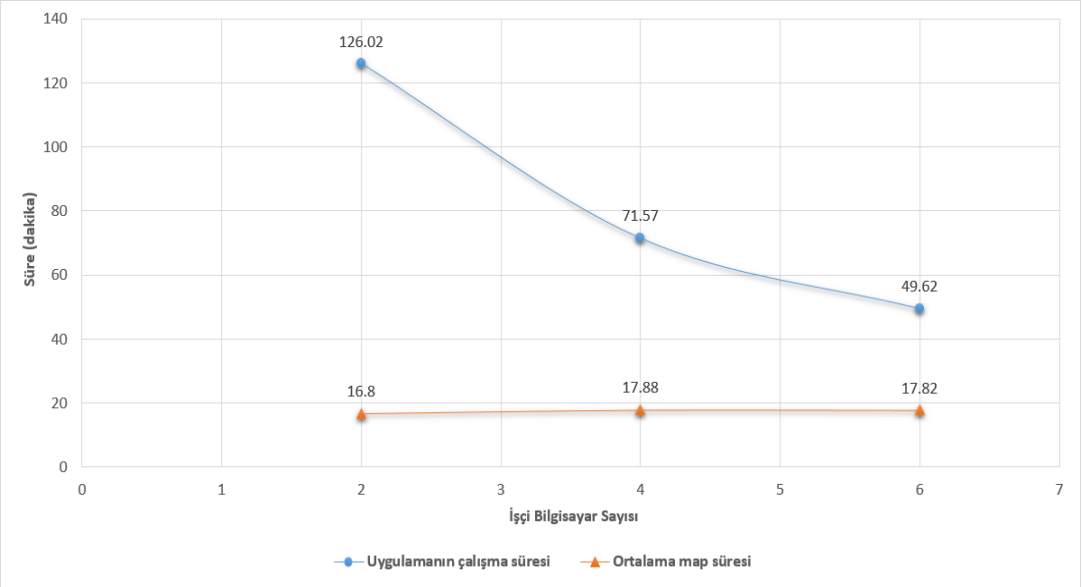


Şekil 5.10. Dağıtık görüntü işleme uygulaması çalışırken takip edilmesi

Dağıtık görüntü işleme uygulamasında 13 GB boyutunda olan PIPA veri kümesindeki fotoğraf albümleri 12 GB boyutunda SequenceFile formatına dönüştürülmüştür. Bu dosya blok boyutu 128 MB olacak şekilde HDFS ortamına kaydedilmektedir. Bu 12 GB SequenceFile formatındaki dosya, dağıtık görüntü işleme algoritması tarafından 128 MB parçalara bölünerek 94 parça elde edilmiştir. 94 adet bölümden oluşan her bir dosya bir map fonksiyonu tarafından işlenmiştir. Map fonksiyonu çalışırken SequenceFile formatındaki dosya içindeki fotoğraf albümleri alınarak bu albümlerden yüz fotoğrafları tespit edilmekte ve bu tespit edilen yüz görüntü dosyaları tekrar SequenceFile dosya formatında HDFS ortamına yazılmaktadır. Map işlemleri işçi bilgisayarlar üzerinde bulunan konteynırlarda çalışmaktadır. Map fonksiyonlarının üzerinde çalıştırılacağı konteynır sayısı artırılarak aynı anda çalıştırılabilecek map sayısı da artırılabilir. Aynı anda çalıştırılabilir map sayısının artırılması işlenecek verinin daha hızlı işlenmesi anlamına gelmektedir. Aynı anda çalışan map sayısını artırmak yani daha fazla YARN konteynırı oluşturabilmek için iki yöntem izlenebilir. Bu yöntemler veri işlemenin yapıldığı işçi bilgisayar kaynakları artırılması yani dikey ölçeklendirme veya mevcut işçi bilgisayar sayısının artırılması yani yatay ölçeklendirme sonucunda konteynır sayısının artırılmasıdır. Yapılan çalışmada Hadoop kümesindeki işçi bilgisayar sayısı artırılarak yani yatay ölçeklendirme yapılmıştır. Geliştirilen dağıtık görüntü işleme uygulamasının yürütme süresi Tablo 5.2'de ve Şekil 5.11' de gösterildiği şekilde azaltılmıştır.

Tablo 5.2. Dağıtık görüntü işleme uygulamasının farklı ölçeklerdeki Hadoop kümesi üzerinde yürütülmesi

Usta Bilgisayar Sayısı	İşçi Bilgisayar Sayısı	Ortalama Map Süresi	Toplam Map İşlemi Sayısı	Toplam Reduce İşlemi Sayısı	Toplam Süre
1	2	16 dk 48 sn	94	0	2 saat 6 dk 1 sn
1	4	17 dk 53 sn	94	0	1 saat 11 dk 34 sn
1	6	17 dk 49 sn	94	0	49 dk, 37 sn



Şekil 5.11. Dağıtık görüntü işleme uygulaması işçi sanal bilgisayar sayısına göre yürütme süresi

5.3. Dağıtık Video İşleme Uygulaması (Video Dosyalarında Sahne Değişim Tespiti)

Java Maven projesi olarak geliştirilen bu uygulamada da video dosyalarının işlenmesi için Hadoop MapReduce programlama modeli kullanılmıştır. Bu uygulamada video dosyaları çerçeve çerçeve okunmaktadır. Okunan çerçeveler arası değişimler OpenIMAJ kütüphanesinde bulunan LocalHistogramVideoShotDetector kullanılarak hesaplanmakta ve sahne değişimleri bulunmaktadır. Her farklı sahne için bir çerçeve seçilerek kaydedilmektedir.

Video dosyasında saniyede gösterilen çerçeve sayısı (fps), video standardına ve video çeken aygıta göre değişmektir. Bunun yanında 24 fps, 25 fps ve 30 fps olan videolar yaygın olarak kullanılmaktadır. Bir videoda belli süre değişmeyen sahneler, videonun o sahneye ait çok sayıda çerçeveyi içermesi anlamına gelmektedir. Videodan ekran görüntüleri alınarak video özeti çıkarılmak istendiğinde aynı sahneden birden fazla görüntü almak yerine farklı sahnelerden oluşan görüntüleri almak daha makul bir çözüm sağlar. Bu uygulamada video dosyalarından farklı sahneleri bulmak için bir yöntem geliştirilmiştir. Bu uygulamada da amaç farklı sahneleri bulmak için kullanılan algoritmanın doğruluğunu veya performansını test etmek değil, gerçekleştirilen dağıtık video işleme sistemini test etmektir. Tasarlanan sistemde Map veya Reduce aşamasında farklı algoritmalar kullanılarak farklı çözümler üretilebilir. Bu uygulamadaki amaç video dosyalarını Hadoop MapReduce programlama modeli kullanarak dağıtık işlemeyi sağlamak için bir sistem tasarlamak ve gerekli olan

yazılımı yazmak daha sonra da bu sistemi test etmektir. Bunun için Şekil 5.12'de gösterilen sistem gerçekleştirilmiştir.

Bu sistemde video dosyaları HDFS ortamında tutulmaktadır. Şekil 5.12'de gösterilen 1 numaralı aşamada HDFS ortamından video dosyaları okunmakta ve videolardaki çerçevelerden giriş anahtar, değer çiftleri oluşturulmaktadır.

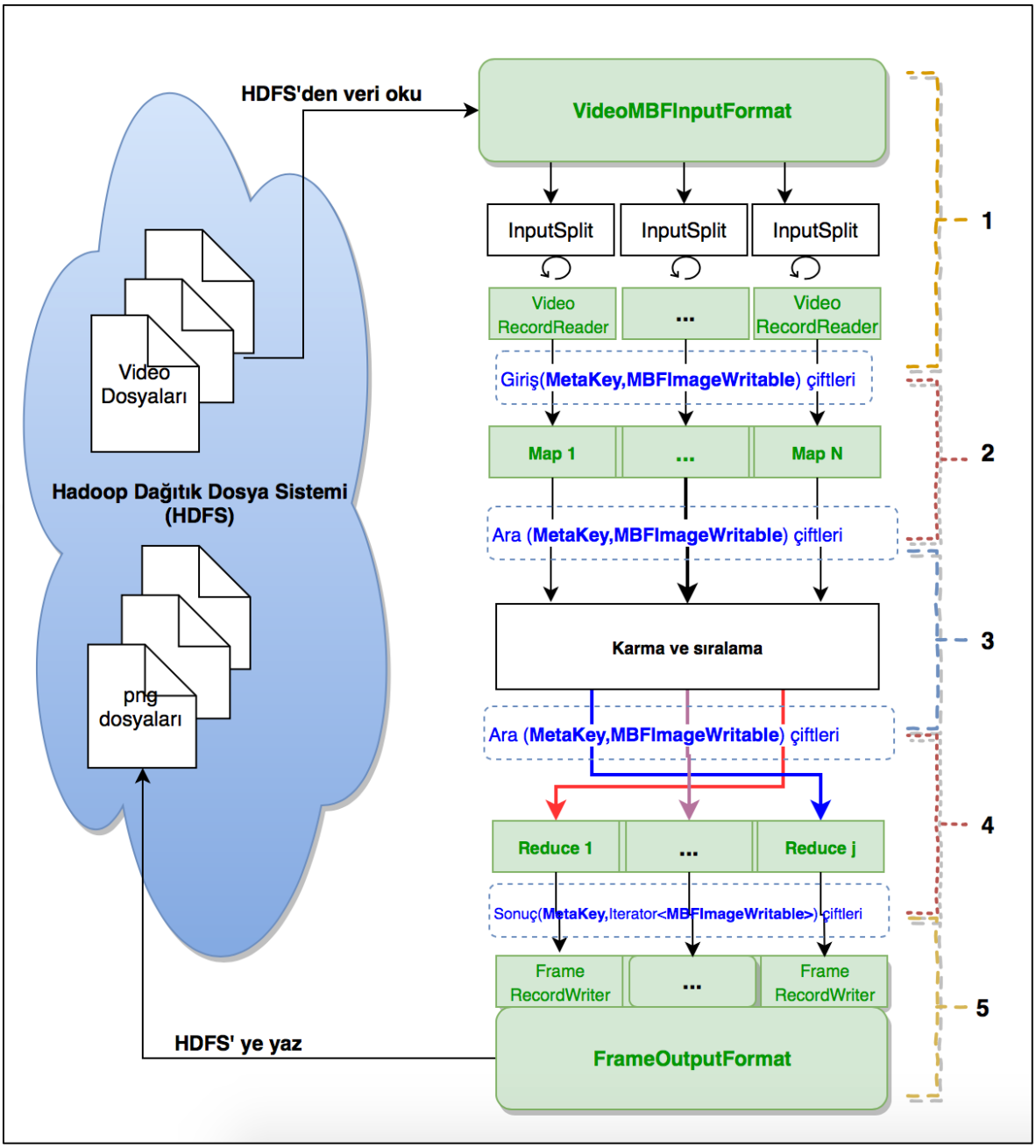
Hadoop, MapReduce ile işlenecek verileri okumak için InputFormat sınıfından türetilen sınıfları kullanmaktadır. Hadoop kütüphanesi InputFormat sınıfından türetilen SequenceFileInputFormat, FileInputFormat, TextInputFormat gibi farklı sınıfları sağlamaktadır. Video dosyaları gibi özel formattaki dosyaların okunabilmesi için InputFormat sınıfından türetilen o işleme özel yeni bir InputFormat sınıfının yazılması gerekmektedir.

Bu uygulamada video dosyalarının okunması için FileInputFormat sınıfından türetilen VideoMBFInputFormat adında yeni bir sınıf geliştirilmiştir. Ayrıca VideoRecordReader adında OpenIMAJ kütüphanesi kullanılarak geliştirilen yeni bir RecordReader sınıfı yazılmıştır. VideoRecordReader sınıfı ile HDFS ortamındaki videolar okunarak Map metodu tarafından kullanılan anahtar, değer çiftleri üretilmektedir. VideoMBFInputFormat sınıfı VideoRecordReader sınıfını RecordReader sınıfı olarak kullanmaktadır.

HDFS ortamındaki dosya ve dizin URL'leri "hdfs://" ön eki ile başlamaktadır. OpenIMAJ kütüphanesinde video okuma kütüphanesi hdfs protokolünü tanımadığından hdfs ortamından videoların okunması için gerekli protokol sınıfı yazılmıştır.

MapReduce programlama modelinde Map ve Reduce metotları verileri anahtar değer çifti olarak alır işler ve anahtar-değer çiftleri olarak sonuç üretir. Hadoop WritableComparable interface sınıfından türetilen Text, BooleanWritable, BytesWritable gibi bir çok anahtar veri türünü sağlamaktadır. Writable interface sınıfından türetilen ArrayWritable, BooleanWritable, BytesWritable gibi değerleri ifade etmek için kullanılan veri türü sınıfları da mevcuttur. Bu standart anahtar ve veri türleri dışında veriye özel anahtar ve veri türleri oluşturulabilir. Bu uygulama için videodan okunan çerçeve verisini tutan MBFImageWritable adında Writable interface sınıfından türetilen yeni bir değer (value) veri türü sınıfı oluşturulmuştur. Çerçevenin sahip olduğu video bilgisi, çerçeve numarası, çerçevenin zaman damgası, çerçeve grup numarası gibi bilgileri tutan MetaKey adında WritableComparable interface sınıfından türetilen yeni bir anahtar(key) veri türü geliştirilmiştir. Geliştirilen MetaKey, MBFImageWritable veri türleri bu uygulamada kullanılmıştır.

Şekil 5.12'de gösterilen 2 numaralı aşamada VideoRecordReader tarafından üretilen MetaKey, MBFImageWritable veri çiftleri Map metodunun içerisinde işlenmektedir. MBFImageWritable veri tipinde OpenIMAJ kütüphanesinin görüntü dosyalarını temsil etmek için kullandığı MBFImage nesnesi bulunmaktadır. Bu da OpenIMAJ kütüphanesinin sağladığı MBFImage için uygulanabilen tüm işlemlerin kolayca uygulanabilmesini sağlamaktadır. MBFImageWritable içinde tutulan MBFImage nesnesi videolardan elde edilen çerçeve bilgilerini tutmaktadır. Bu uygulamada Map metodunun içinde LocalHistogramVideoShotDetector ile videodan farklı sahnelere ait çerçeve görüntüleri elde edilmiş, elde edilen bu görüntülere bir numara verilmiştir. Bu numara bilgisi MetaKey anahtar verisinin içinde tutulmuştur. MetaKey anahtar verisi görüntüler HDFS ortamında yazılırken dosya isimlendirmesinde kullanılmıştır.



Şekil 5.12. Dağıtık video işleme uygulamasının mimarisi

Şekil 5.12'de gösterilen 3 numaralı aşamada MetaKey veri yapısı oluşturulurken yazılan compareTo metodu kullanılarak üretilen ara MetaKey, MBFImageWritable çiftleri sıralanır. Daha sonra aynı anahtar değerine göre MetaKey, MBFImageWritable çiftleri gruplanarak Reduce metoduna verilecek hale getirilir. Bu aşamada farklı Map metotları tarafından üretilen compareTo metoduna göre aynı anahtar olan çiftler bir araya getirilir.

Şekil 5.12'de gösterilen 4 numaralı aşamada 3 numaralı aşamada sıralanan ve gruplanan MetaKey, Iterator <MBFImageWritable> değer çiftleri Reduce metodu ile alındıktan sonra herhangi bir analiz, filtreleme veri birleştirme işlemi gerçekleştirilebilir. Bu uygulamada

yapılmak istenen işlem Map metodunda gerçekleştirilmiştir. Geliştirilen ilk uygulamada Reduce metodunu tanımlı olmasına rağmen herhangi bir işleme işlemi gerçekleştirilmemiştir. Hiçbir veri işleme işlemi gerçekleştirilmeden Reduce aşamasının olduğu ve Reduce aşamasının olmadığı uygulamalar yürütülüp Tablo 5.3'de gösterilen sonuçlar elde edilmiştir. Yapılan testte Reduce işleminde sadece FrameRecordWriter sınıfına yazma işlemi yapılsa da Reduce işleminin tanımlı olması 3 numaralı ve 4 numaralı aşamaların yürütüldüğü ve bu aşamalarda önemli derecede zaman harcandığı gözlemlenmiştir. Bundan dolayı Reduce metodu "job.setNumReduceTasks(0)" ayarlaması ile Map metodunun sonucunda üretilen veriler 3 numaralı ve 4 numaralı aşamalar yürütülmeden 5 numaralı aşamaya geçmesi sağlanmıştır. Bu işlem sonucunda elde edilen kazanç Tablo 5.3'de gösterilmiştir. Reduce metodu kullanılması gereken problemler için Reduce metodu tanımlanarak kullanılabilir. Bu uygulamada Reduce metodunun kullanılmasına gerek olmadığı için kullanılmamıştır.

Tablo 5.3. Dağıtık video işleme uygulama aşamalarında harcanan süreler

Usta Makine Sayısı	**İşçi Makine Sayısı**	**Ortalama Map Süresi**	**Ortalama Shuffle Süresi**	**Ortalama Merge Süresi**	**Ortalama Reduce Süresi**	**Map Sayısı**	**Reduce Sayısı**	**Toplam Süre**
1	2	74 sn	25 dk 25 sn	8 sn	9 dk 19 sn	84	1	43 dk 7 sn
1	2	70 sn	0	0	0	84	0	25 dk 53 sn

Şekil 5.12'de gösterilen 5 numaralı aşamada elde edilen sonuçlar HDFS ortamına yazılmaktadır. Hadoop elde edilen sonuçları yazmak için OutputFormat sınıflarını kullanılır. Standart olarak FileOutputFormat, SequenceFileOutputFormat gibi sınıfları sağlamaktadır. Fakat "png" gibi özel bir formatta veri yazmak için özel OutputFormat sınıfının yazılması gerekmektedir. Bu uygulamada farklı sahneler için elde edilen görüntü dosyalarının png formatında HDFS ortamına yazılması için FrameOutputFormat adında yeni bir sınıf geliştirilmiştir.

Geliştirilen dağıtık video işleme uygulaması Hadoop kümesi üzerinde aşağıdaki şekilde çalıştırılmıştır.

1- Önceki bölümde anlatılan erişim yöntemlerinden biri ile usta sanal bilgisayarına erişilir.
2- "hdfs" kullanıcısı ile giriş yapılır ve hdfs kullanıcısının ana dizinine gidilir.

```
[cloud-user@cluster-master-large-0 ~]$ sudo su hdfs
[hdfs@cluster-master-large-0 cloud-user]$ cd
```

3- Hollywood2-scenes [102] veri kümesi wget komutu ile aşağıdaki şekilde indirilmektedir. İndirilen veri kümesinden 40 MB ile 64 MB boyut aralığındaki video dosyaları dağıtık video işleme uygulamasını test etmek için kullanılacaktır. Bunun için bahsedilen video dosyaları "filteredVideoFiles" adında bir klasör oluşturularak o klasöre kopyalanmıştır. "filteredVideoFiles" klasöründe 86 adet video dosyası bulunmaktadır. Bu video dosyalarının toplam boyutu 4.1 GB' dır.

```
[hdfs@cluster-master-large-0 ~]$ wget ftp://ftp.irisa.fr/local/vistas/actions/Hollywood2-
scenes.tar.gz
[hdfs@cluster-master-large-0 ~]$ tar -zxvf Hollywood2-scenes.tar.gz
[hdfs@cluster-master-large-0 ~]$ mkdir filteredVideoFiles
[hdfs@cluster-master-large-0 ~]$ cd Hollywood2/AVIClipsScenes/
[hdfs@cluster-master-large-0 AVIClipsScenes]$  find  . -size +40M -a -size -64M -exec cp
-t /home/hdfs/filteredVideoFiles/ {} \+
[hdfs@cluster-master-large-0 AVIClipsScenes]$ cd /home/hdfs/filteredVideoFiles
[hdfs@cluster-master-large-0 filteredVideoFiles]$ ls -lh
total 4.1G

[hdfs@cluster-master-large-0 filteredVideoFiles]$ find /home/hdfs/filteredVideoFiles/ -
type f| wc -l
86
```

4- "filteredVideoFiles" dizinindeki videolar HDFS ortamına aşağıdaki şekilde "/mapred/app/dgi/input/video/Hollywood2/filteredVideoFiles" adında bir dizin oluşturularak o dizinin içerisine kopyalanmaktadır. HDFS dosya işlemleri için "hadoop fs" komutu kullanılmaktadır.

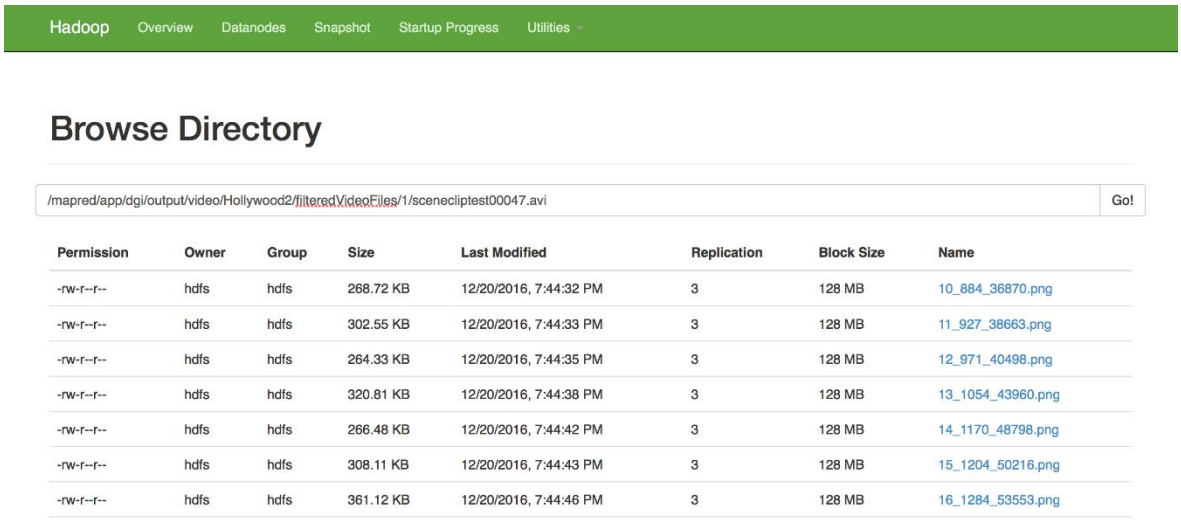
```
[hdfs@cluster-master-large-0 filteredVideoFiles]$ hadoop fs -mkdir -p
/mapred/app/dgi/input/video/Hollywood2/filteredVideoFiles

[hdfs@cluster-master-large-0 filteredVideoFiles]$ hadoop fs -copyFromLocal *
/mapred/app/dgi/input/video/Hollywood2/filteredVideoFiles
```

5- Geliştirilen dağıtık video işleme uygulaması dvi.jar dosyası olarak usta sanal bilgisayara aşağıdaki şekilde kopyalanır. Bunun için "scp" komutu kullanılmaktadır. 10.1.43.3 IP'li bilgisayar usta bilgisayardan ssh ile erişilebilir bir bilgisayardır.

```
[hdfs@cluster-master-large-0 ~]$ scp
murattezgider@10.1.43.3:/Users/murattezgider/NetBeansProjects/tez/hadoop/target/dvi.jar
dvi.jar
```

6- Dağıtık video işleme uygulaması aşağıdaki şekilde Hadoop kümesi üzerinde çalıştırılmaktadır. HDFS adresi olarak usta sanal bilgisayarın IP adresi verilmektedir. Sonuçlar yine HDFS ortamına yazılacaktır.

```
hadoop jar dvi.jar
hdfs://10.1.57.1/mapred/app/dgi/input/video/Hollywood2/filteredVideoFiles
hdfs://10.1.57.1/mapred/app/dgi/output/video/Hollywood2/filteredVideoFiles/1
```



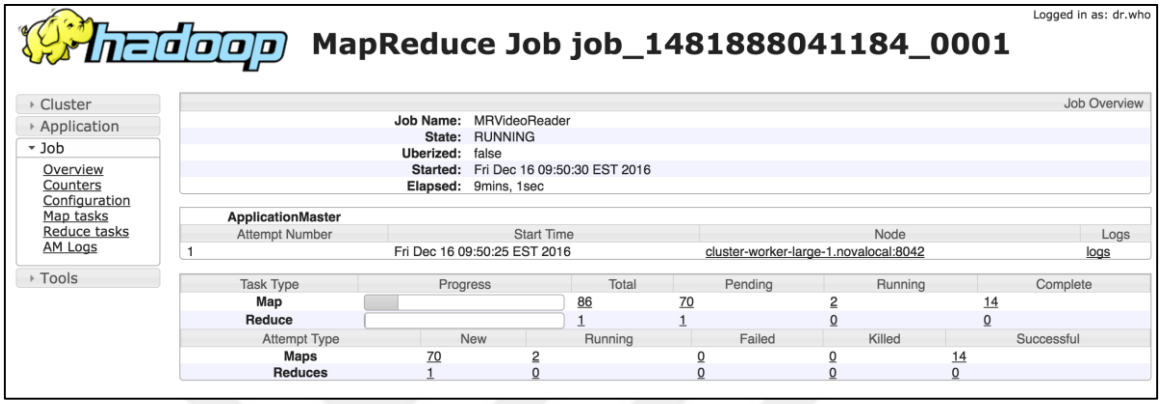
Şekil 5.13. Dağıtık video işleme uygulaması ile elde edilen sahne görüntülerinin HDFS ortamında listelenmesi

Şekil 5.14. Dağıtık video işleme uygulaması ile elde edilen sahneler

Geliştirilen dağıtık video işleme uygulaması B3LAB bulut sistemi kullanılarak test edilmiştir. Bunun için önceki bölümde oluşturulma aşamaları anlatılan 1 usta 2 işçi sanal bilgisayardan oluşan Hadoop kümesi üzerinde uygulama yukarıda anlatılan adımlar takip edilerek çalıştırılmıştır. Uygulama sonuçlar için her video dosyası için video ismiyle dizinler oluşturmaktadır. Bu dizinlere o videodan elde edilen sahneleri kaydetmektedir. Şekil 5.13'de uygulama tarafından oluşturulan bir dizin ve altında tutulan görüntü dosyaları gösterilmektedir. Şekil 5.14' de ise o Hollywood2 veri kümesinde bulunan scenecliptest0047.avi video dosyasından elde edilen ilk 15 sahne gösterilmektedir. Elde

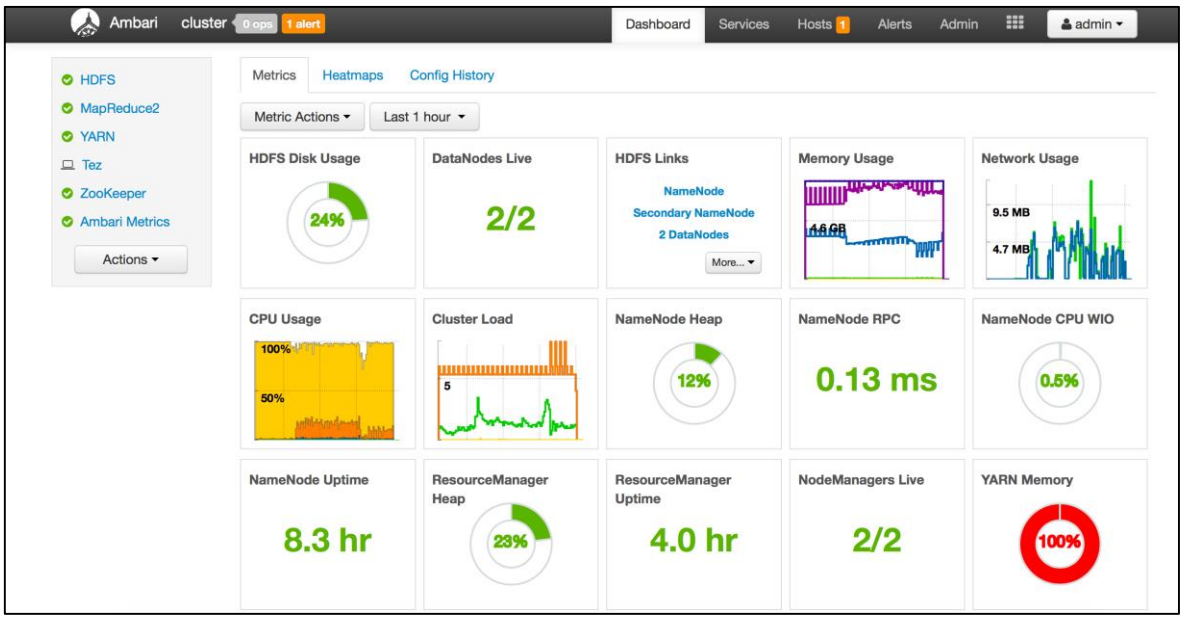
edilen sahneler; sahne numarası, çerçeve numarası ve çerçevenin video içinde geçtiği zaman bilgisi ile adlandırılmıştır.

Dağıtık video işleme uygulaması Hadoop kümesi üzerinde çalışırken Şekil 5.15'deki ekran ile takip edilebilmektedir. Bu ekrana uygulama çalıştırılırken oluşturulan izleme link kullanılarak erişilebilir.



Şekil 5.15. Dağıtık video işleme uygulamasının çalışırken izlenmesi

Dağıtık video işleme uygulaması çalıştırılırken Ambari metrikleri Şekil 5.16'de gösterilmiştir. Bu metriklerde YARN konteynırları için ayrılan hafızanın tamamının kullanıldığı görülmektedir.



Şekil 5.16. Dağıtık video işleme uygulaması çalışırken Ambari metrikleri

Hadoop kümesi üzerinde uygulamalar çalıştırılırken elde edilen sonuçlarda kullanılan sanal bilgisayarların kaynaklarının yanında, YARN ve MapReduce ayarlarının da doğru bir

şekilde yapılandırılması oldukça önemlidir. Bunun için 1 usta 2 işçi bilgisayardan oluşan Tablo 5.4 ve Tablo 5.5'deki yapılandırma ayarlarına sahip Hadoop kümesi üzerinde uygulamalar denenmiş en iyi sonucun alındığı Tablo 5.5'deki yapılandırma ayarları dağıtık video işleme uygulaması için kullanılmıştır.

Tablo 5.4. Video işleme uygulaması için yapılan 1. yapılandırma

YARN HAFIZA YAPILANDIRMASI			
Sanal Makine RAM	Bütün YARN Konteynırların Kullanabileceği RAM	YARN Konteynır Minimum RAM	YARN Konteynır Maksimum RAM
8 GB	7 GB	2 GB	7GB
YARN vCPU YAPILANDIRMASI			
Sanal Makine vCPU	Bütün YARN Konteynırların kullanabileceği vCPU	YARN Konteynır Minimum vCPU	YARN Konteynır Maksimum vCPU
8 core	8 core	2 core	8 core
MAPREDUCE YAPILANDIRMASI			
Map	Reduce	Sort Allocation	Application Master
3GB	3GB	1719 MB	2 GB

Tablo 5.5'de gösterilen gibi her bir düğümde bulunan 8 GB hafıza alanından 7 GB hafıza alanı YARN konteynırları tarafından kullanılabilmesine izin verilmiştir. Her konteynır en az 1 GB, en fazla 3 GB hafıza alanı kullanabilmektedir. Her konteynır en az 1 CPU, en fazla 3 CPU kullanabilmektedir. Her bir map fonksiyonun 3 GB hafıza kullanmasına izin verilmiştir. Application Master için ise 1 GB hafıza alanı ayrılmıştır. Bu yapılandırma ayarlarına göre 1 GB hafıza alanını sahip konteynır üzerinde 1 adet Application Master ve 3 GB hafıza alanına sahip konteynırlar üzerinde map işlemi çalışmaktadır.

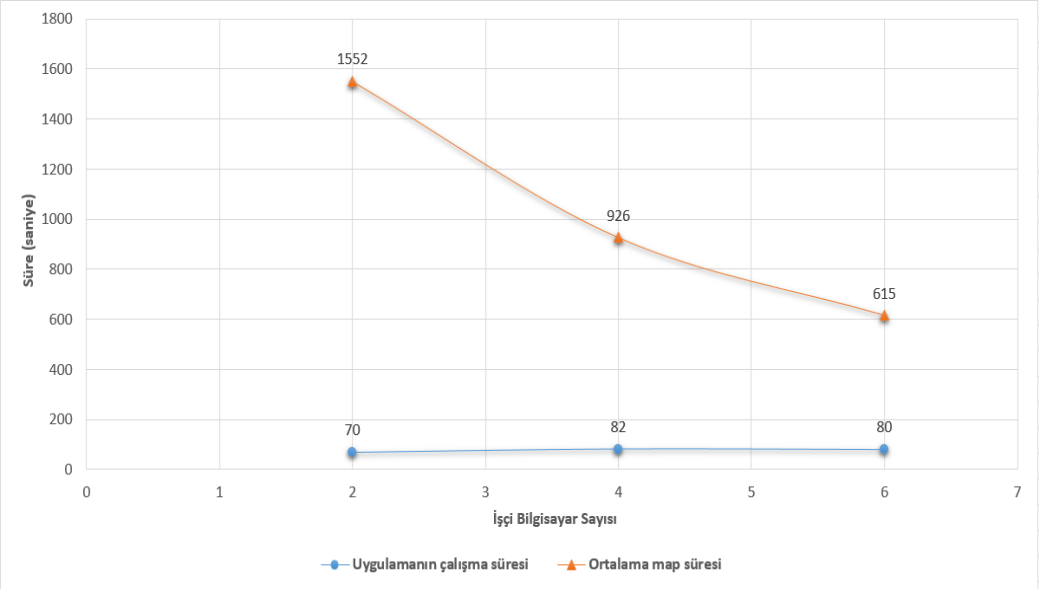
Tablo 5.5. Video işleme uygulaması için yapılan 2. yapılandırma

YARN HAFIZA YAPILANDIRMASI			
Sanal Makine RAM	Bütün YARN Konteynırların kullanabileceği RAM	YARN Konteynır Minimum RAM	YARN Konteynır Maksimum RAM
8 GB	7 GB	1 GB	3GB
YARN vCPU YAPILANDIRMASI			
Sanal Makine vCPU	Bütün YARN Konteynırlarının kullanabileceği vCPU	YARN Konteynır Minimum vCPU	YARN Konteynır Maksimum vCPU
8 core	%80	1 core	3 core
MAPREDUCE YAPILANDIRMASI			
Map	Reduce	Sort Allocation	Application Master
3GB	3GB	1.5 GB	1 GB

Tablo 5.5'deki yapılandırma ayarlarına sahip Hadoop kümesindeki işçi bilgisayar sayısı artırılarak yani yatay ölçeklendirme yapılarak dağıtık video işleme uygulaması küme üzerinde çalıştırılmıştır. Farklı işçi bilgisayar sayısına sahip Hadoop kümesi üzerinde çalıştırılan dağıtık video işleme uygulaması için Tablo 5.6'da ve Şekil 5.17' de gösterilen sonuçlar elde edilmiştir. Elde edilen sonuçlara göre işçi bilgisayar sayısının artırılması ile dağıtık görüntü işleme uygulamasının yürütülme süresi azalmıştır.

Tablo 5.6. Dağıtık video işleme uygulamasının farklı ölçeklerdeki Hadoop kümesini üzerinde yürütülmesi

Usta Bilgisayar Sayısı	**İşçi Bilgisayar Sayısı**	**Ortalama Map Süresi**	**Toplam Map İşlemi Sayısı**	**Toplam Reduce İşlemi Sayısı**	**Toplam Süre**
1	2	1 dk, 10sn	86	0	25 dk, 52 sn
1	4	1 dk, 22sn	86	0	15 dk 26 sn
1	6	1 dk, 20sn	86	0	10 dk 15 sn



Şekil 5.17. Dağıtık video işleme uygulaması işçi sanal bilgisayar sayısına göre yürütme süresi

6. SONUÇLAR

Bu tez çalışmasında büyük veri dünyasında önemli bir yer tutan görüntü ve video dosyalarının dağıtık bir şekilde işlenmesi konu alınmıştır. Bu dosyaların dağıtık olarak işlenmesi için TÜBİTAK BİLGEM B3LAB OpenStack IaaS bulut bilişim altyapı sistemi kullanılarak oluşturulan sanal makineler üzerine farklı ölçeklerde Hadoop kümeleri oluşturulmuştur. OpenStack ile Hadoop kümesi oluşturma işlemi ilgili bölümde ayrıntılı olarak anlatılmıştır. Daha sonra Hadoop Mapreduce programlama modeli kullanılarak dağıtık görüntü işleme ve dağıtık video işleme uygulamaları geliştirilmiştir. Hadoop kütüphanesi ile video dosyalarının okunması ve işlenmesi için Hadoop kütüphanesi tarafından sağlanmayan gerekli sınıflar yazılmıştır. Geliştirilen bu uygulamalar 1 usta 2 işçi, 1 usta 4 işçi ve 1 usta 6 işçi bilgisayardan oluşan farklı ölçekteki Hadoop kümeleri üzerinde çalıştırılmıştır.

Yapılan bu çalışmada elde edilen sonuçlara göre video ve görüntü dosyalarının işlenmesi için Hadoop kümelerinin ve MapReduce programlama modelinin kullanılmasının uygun bir çözüm sağladığı anlaşılmaktadır. YARN konteynır sayesinde CPU ve RAM gibi hesaplama kaynaklarının verimli bir şekilde yönetilebilmesi ayrı bir avantaj sağlamaktadır. Dağıtık görüntü ve video işleme için geliştirilen yazılımda video ve görüntü okuma ve işleme işlemlerinde OpenIMAJ kütüphanesi kullanılmıştır. Geliştirilen sistem ile OpenIMAJ kütüphanesinin sağladığı algoritmaların görüntü ve video işleme alanında dağıtık olarak uygulanması sağlanmıştır. Geliştirilen bu sistem sayesinde hızlı bir şeklide artan görüntü ve video dosyalarının işlenmesi ve analizi için dağıtık sistemler kullanılarak önemli bir çözüm sağlamıştır. Bu mimari sosyal ağlar gibi yoğun bir şekilde video ve görüntü paylaşımın olduğu alanlarda kullanılabilir.

Geliştirilen sistem ile yoğun hesaplama kaynağı gerektiren büyük miktardaki video ve görüntü dosyalarının işlenmesi ve analizi için ölçeklenebilir, güvenilir bir çözüm sunulmaktadır. Ayrıca görüntü ve video dosyalarının depolanması problemi için dağıtık dosya sistemleri kullanılabilir. HDFS ortamında tutulan veriler Hadoop ile işlenerek tekrar HDFS ortamına yazılabilir.

KAYNAKLAR

[1] **A. S. Tanenbaum and M. v. Steen,** *Distributed Systems: Principles and Paradigms*: Prentice-Hall, Inc., 2007.

[2] **J. Venner, S. Wadkar, and M. Siddalingaiah,** *Pro Apache Hadoop*: Apress, 2014.

[3] **J. Hurwitz, R. Bloor, M. Kaufman, and F. Halper,** *Cloud computing for dummies*: John Wiley & Sons, 2010.

[4] **S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema,** "A performance analysis of EC2 cloud computing services for scientific computing," in *Cloud computing*, ed: Springer, 2009, pp. 115-131.

[5] **S. El-Kazzaz and A. El-Mahdy,** "A Hadoop-Based Framework for Large-Scale Landmine Detection Using Ubiquitous Big Satellite Imaging Data," in *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, 2015, pp. 274-278.

[6] **D. Mera, M. Batko, and P. Zezula,** "Towards fast multimedia feature extraction: Hadoop or storm," in *Multimedia (ISM), 2014 IEEE International Symposium on*, 2014, pp. 106-109.

[7] **A. K. Sabarad, M. H. Kankudti, S. Meena, and M. Husain,** "Color and Texture Feature Extraction Using Apache Hadoop Framework," in *Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on*, 2015, pp. 585-588.

[8] **S. M. Banaei and H. K. Moghaddam,** "Hadoop and its role in modern image processing," *Open Journal of Marine Science*, vol. 4, p. 239, 2014.

[9] **L. Kennedy, M. Slaney, and K. Weinberger,** "Reliable tags using image similarity: mining specificity and expertise from large-scale multimedia databases," in *Proceedings of the 1st workshop on Web-scale multimedia corpus*, 2009, pp. 17-24.

[10] **H. Kocakulak and T. T. Temizel,** "A Hadoop solution for ballistic image analysis and recognition," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, 2011, pp. 836-842.

[11] **Y. Li, D. J. Crandall, and D. P. Huttenlocher,** "Landmark classification in large-scale image collections," in *ICCV*, 2009, pp. 1957-1964.

[12] **K. Michael and K. W. Miller,** "Big data: New opportunities and new challenges [guest editors' introduction]," *Computer*, vol. 46, pp. 22-24, 2013.

[13] **Shelly and N. Raghava,** "Iris recognition on hadoop: A biometrics system implementation on cloud computing," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 2011, pp. 482-485.

[14] **L. Shi, B. Wu, B. Wang, and X. Yan,** "Map/reduce in CBIR application," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, 2011, pp. 2465-2468.

[15] **B. White, T. Yeh, J. Lin, and L. Davis,** "Web-scale computer vision using mapreduce for multimedia data mining," in *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, 2010, p. 9.

[16] **R. Yan, M.-O. Fleury, M. Merler, A. Natsev, and J. R. Smith,** "Large-scale multimedia semantic concept modeling using robust subspace bagging and MapReduce," in *Proceedings of the First ACM workshop on Large-scale multimedia retrieval and mining*, 2009, pp. 35-42.

[17] **C.-T. Yang, L.-T. Chen, W.-L. Chou, and K.-C. Wang,** "Implementation of a medical image file accessing system on cloud computing," in *Computational Science*

and Engineering (CSE), 2010 IEEE 13th International Conference on, 2010, pp. 321-326.

[18] **J. Zhao, Q. Li, and H. Zhou,** "A cloud-based system for spatial analysis service," in *Remote Sensing, Environment and Transportation Engineering (RSETE), 2011 International Conference on*, 2011, pp. 1-4.

[19] **G. E. Moore,** "Cramming More Components Onto Integrated Circuits," *Proceedings of the IEEE,* vol. 86, pp. 82-85, 1998.

[20] **S. H. Fuller and L. I. Millett,** "Computing Performance: Game Over or Next Level?," *Computer,* vol. 44, pp. 31-38, 2011.

[21] **R. Duncan,** "A survey of parallel computer architectures," *Computer,* vol. 23, pp. 5-16, 1990.

[22] **M. J. Flynn,** "Some computer organizations and their effectiveness," *Computers, IEEE Transactions on,* vol. 100, pp. 948-960, 1972.

[23] **P. Pacheco,** *An introduction to parallel programming*: Elsevier, 2011.

[24] **A. SAYAR and U. ERGÜN,** "Fonksiyonel Programlama Dilleri ile Paralel Programlama," *Niğde Üniversitesi Mühendislik Bilimleri Dergisi,* vol. 3, pp. 1-17, 2014.

[25] **P. M. Mell and T. Grance,** "SP 800-145. The NIST Definition of Cloud Computing," National Institute of Standards \& Technology2011.

[26] **G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, *et al.*,** "Scientific workflow applications on Amazon EC2," in *2009 5th IEEE International Conference on E-Science Workshops*, 2009, pp. 59-66.

[27] **E. Network and I. S. Agency,** *Cloud Computing: Benefits, risks and recommendations for information security*: ENISA, 2009.

[28] **O. Şanlı,** "Bulut Bilişim," *Akademik Bilişim,* pp. 2-4, 2011.

[29] **K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa,** "Science clouds: Early experiences in cloud computing for scientific applications," *Cloud computing and applications,* vol. 2008, pp. 825-830, 2008.

[30] **Wikipedia.** (20/06/2016). *Cloud Computing*. Available: https://en.wikipedia.org/wiki/Cloud_computing

[31] **S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema,** "A performance analysis of EC2 cloud computing services for scientific computing," in *International Conference on Cloud Computing*, 2009, pp. 115-131.

[32] **T. Fifield, D. Fleming, A. Gentle, L. Hochstein, J. Proulx, E. Toews, *et al.*,** *OpenStack Operations Guide*: " O'Reilly Media, Inc.", 2014.

[33] (20/06/2016). *OpenStack.* Available: http://www.openstack.org/

[34] *OpenNebula*. Available: http://opennebula.org/

[35] (20/06/2016). *Nimbus Project.* Available: http://www.nimbusproject.org/

[36] **Wikipedia.** (20/07/2016). *Bulut Bilişim*. Available: https://tr.wikipedia.org/wiki/Bulut_bili%C5%9Fim

[37] **A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton,** "Big data," *The management revolution. Harvard Bus Rev,* vol. 90, pp. 61-67, 2012.

[38] **A. Zaslavsky, C. Perera, and D. Georgakopoulos,** "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159,* 2013.

[39] **M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales, and P. Tufano,** "Analytics: The real-world use of big data," *IBM Global Business Services,* pp. 1-20, 2012.

[40] **Y. Demchenko, C. De Laat, and P. Membrey,** "Defining architecture components of the Big Data Ecosystem," in *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, 2014, pp. 104-112.
[41] **A. Baaziz and L. Quoniam,** "How to use Big Data technologies to optimize operations in Upstream Petroleum Industry," *Baaziz, A., & Quoniam, L.(2013). How to use Big Data technologies to optimize operations in Upstream Petroleum Industry. International Journal of Innovation-IJI*, vol. 1, pp. 19-25, 2013.
[42] **X. Jin, B. W. Wah, X. Cheng, and Y. Wang,** "Significance and Challenges of Big Data Research," *Big Data Research*, vol. 2, pp. 59-64, 6// 2015.
[43] **A. Jacobs,** "The pathologies of big data," *Commun. ACM*, vol. 52, pp. 36-44, 2009.
[44] **J. Dean and S. Ghemawat,** "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, 2008.
[45] **S. Ghemawat, H. Gobioff, and S.-T. Leung,** "The Google file system," in *ACM SIGOPS operating systems review*, 2003, pp. 29-43.
[46] (08/04/2016). *Apache Hadoop*. Available: http://hadoop.apache.org/
[47] **J. Dean and S. Ghemawat,** "MapReduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, pp. 72-77, 2010.
[48] **İ. Demir and A. Sayar,** "Hadoop plugin for distributed and parallel image processing," in *2012 20th Signal Processing and Communications Applications Conference (SIU)*, 2012, pp. 1-4.
[49] **G. F. Coulouris, J. Dollimore, and T. Kindberg,** *Distributed systems: concepts and design*: pearson education, 2005.
[50] **R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon,** "Design and implementation of the Sun network filesystem," in *Proceedings of the Summer USENIX conference*, 1985, pp. 119-130.
[51] **J. H. Howard,** *An overview of the andrew file system*: Carnegie Mellon University, Information Technology Center, 1988.
[52] *HDFS Users Guide*. Available: https://hadoop.apache.org/docs/r2.7.0/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html
[53] **U. Ergün, S. Eken, and A. Sayar,** "Güncel Dağıtık Dosya Sistemlerinin Karşılaştırmalı Analizi."
[54] **S. R. Mantena,** "Transparency in Distributed Systems," *CSE*, vol. 6306, p. 13.
[55] **S. Ghemawat, H. Gobioff, and S.-T. Leung.** *The Google File System*. Available: https://www.cs.umd.edu/class/spring2011/cmsc818k/Lectures/gfs-hdfs.pdf
[56] **T. White,** *Hadoop: The definitive guide*: " O'Reilly Media, Inc.", 2012.
[57] *Java Programlama Dili*. Available: https://www.oracle.com/java/index.html
[58] *Hadoop Kullanan Kuruluşlar*. Available: https://wiki.apache.org/hadoop/PoweredBy
[59] (08/12/2016). *Hadoop Architecture - HDFS and Map Reduce*. Available: http://a4academics.com/tutorials/83-hadoop/835-hadoop-architecture
[60] **M. A. Khan, Z. A. Memon, and S. Khan,** "Highly Available Hadoop NameNode Architecture," in *Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on*, 2012, pp. 167-172.
[61] **A. B. Patel, M. Birla, and U. Nair,** "Addressing big data problem using Hadoop and Map Reduce," in *2012 Nirma University International Conference on Engineering (NUiCONE)*, 2012, pp. 1-5.
[62] (2016). *http://hortonworks.com/blog/apache-hadoop-yarn-background-and-an-overview/*. Available: http://hortonworks.com/blog/apache-hadoop-yarn-background-and-an-overview/

[63] **K. Shvachko, H. Kuang, S. Radia, and R. Chansler,** "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, 2010, pp. 1-10.

[64] **S. Mohanty, G. Das, H. Suman, P. Maharana, and R. Ratnakar,** "A Survey on Working Principle and Application of Hadoop," 2015.

[65] *HDFS Architecture*. Available: https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[66] **P. Zikopoulos and C. Eaton,** *Understanding big data: Analytics for enterprise class hadoop and streaming data*: McGraw-Hill Osborne Media, 2011.

[67] **X. Liu, J. Han, Y. Zhong, C. Han, and X. He,** "Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS," in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1-8.

[68] **G. Mackey, S. Sehrish, and J. Wang,** "Improving metadata management for small files in HDFS," in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1-4.

[69] **V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, *et al.*,** "Apache Hadoop YARN: yet another resource negotiator," presented at the Proceedings of the 4th annual Symposium on Cloud Computing, Santa Clara, California, 2013.

[70] **M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly,** "Dryad: distributed data-parallel programs from sequential building blocks," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 59-72, 2007.

[71] **M. Weimer, Y. Chen, B.-G. Chun, T. Condie, C. Curino, C. Douglas, *et al.*,** "REEF: Retainable Evaluator Execution Framework," presented at the Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, 2015.

[72] **M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica,** "Spark: cluster computing with working sets," *HotCloud*, vol. 10, pp. 10-10, 2010.

[73] *Storm*. Available: http://storm-project.net/

[74] *Apache Tez.* Available: http://incubator.apache.org/projects/tez.html

[75] (2016). *Apache Hadoop 2.7.2*. Available: http://hadoop.apache.org/docs/r2.7.2/index.html

[76] **R. C. Gonzalez and R. E. Woods,** *Digital image processing*. Upper Saddle River, NJ [etc.]: Pearson/Prentice Hall, 2008.

[77] **O. Öztaş.** (07/10/2016). *Görüntü İşleme Teknikleri Ders Notları*. Available: http://www.oguzhanoztas.com/gi/ders1.pdf

[78] **B. Bayram.** (20/07/2016). *Sayısal Görüntü İşleme*. Available: http://www.yildiz.edu.tr/~bayram/sgi/saygi.htm

[79] **Wikipedia.** (20/07/2016). *Elektromanyetik tayf*. Available: https://tr.wikipedia.org/wiki/Elektromanyetik_tayf

[80] **S. Karasu,** "Vinç Sistemlerinde Görüntü İşleme Tekniği İle Salınım İncelenmesi Ve Giriş Şekillendirici Denetim," Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı, Bülent Ecevit Üniversitesi, 2013.

[81] **G. M. Perihanoğlu,** "Dijital Görüntü İşleme Teknikleri Kullanılarak Görüntülerden Detay Çıkarımı," Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü, Geomatik Mühendisliği Anabilim Dalı, Geomatik Mühendisliği Programı, İstanbul Teknik Üniversitesi, 2015.

[82] **F. ABA,** "Görüntü İşleme ve Yapay Sinir Ağları Kullanarak Mineral Tanıma," Yüksek Lisans Tezi, Jeoloji Mühendisliği Anabilim Dalı, Fırat Üniversitesi, Elazığ, 2014.
[83] **S. Kazdal,** "Beyin Tümörlerinin İleri Görüntü İşleme Ve Örüntü Tanıma Teknikleri Kullanılarak Bilgisayar Destekli Tespiti," Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü Marmara Üniversitesi, İstanbul, 2013.
[84] **E. Özçelik,** "Ardışık Görüntüler İle Piksel Altı Bilgi Çıkarımı Ve Çözünürlük İyileştirme," Fen Bilimleri Enstitüsü, 2015.
[85] **M. Yavuz Çelikdemir,** "Yüksek Sıcaklığa Maruz Kalmış Betonlarda Meydana Gelen Çatlakların Görüntü İşleme Tekniği İle Tespit Edilmesi," Yüksek Lisans Tezi, Fen Bilimleri Enstitüsi, Elektrik-Elektronik Mühendisliği Anabilim Dalı, Fırat Üniversitesi, Elazığ, 2015.
[86] (20/07/2016). *OpenImaj*. Available: http://openimaj.org/
[87] *OpenCV*. Available: http://opencv.org/
[88] (20/07/2016). *ACM Multimedia 2011*. Available: http://www.acmmm11.org/content-awards-recognitions.html
[89] (20/07/2016). *OpenImaj Modülleri*. Available: http://openimaj.org/tutorial/modules.html
[90] **J. S. Hare, S. Samangooei, and D. P. Dupplaw,** "OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images," presented at the Proceedings of the 19th ACM international conference on Multimedia, Scottsdale, Arizona, USA, 2011.
[91] (20/07/2016). *Java Native Interface(JNI)*. Available: https://en.wikipedia.org/wiki/Java_Native_Interface
[92] **C. Sweeney, L. Liu, S. Arietta, and J. Lawrence,** "HIPI: a Hadoop image processing interface for image-based mapreduce tasks," *Chris. University of Virginia*, 2011.
[93] *Bulut Bilişim ve Büyük Veri Araştırma Laboratuvarı*. Available: http://b3lab.org/
[94] **OpenStack.** (20/07/2016). *OpenStack Documentation*. Available: http://docs.openstack.org/
[95] (20/07/2016). *OpenStack'ı Destekleyen Kuruluşlar*. Available: http://www.openstack.org/foundation/companies/
[96] *Apache Ambari*. Available: https://ambari.apache.org/
[97] *PuTTY*. Available: http://www.putty.org/
[98] *Apache Maven*. Available: https://maven.apache.org/
[99] *People in Photo Album (PIPA) dataset*. Available: https://people.eecs.berkeley.edu/~nzhang/piper.html
[100] **N. Zhang, M. Paluri, Y. Taigman, R. Fergus, and L. Bourdev,** "Beyond frontal faces: Improving person recognition using multiple cues," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4804-4813.
[101] *SequenceFileTool*. Available: http://openimaj.org/openimaj-hadoop/openimaj-hadoop-tools/SequenceFileTool/project-info.html
[102] *Hollywood2 Human Actions and Scenes Dataset*. Available: http://www.di.ens.fr/~laptev/actions/hollywood2/

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Murat TEZGİDER
E-mail : murattezgider@gmail.com

İŞ TECRÜBESİ

2014 Kasım – Devam
Hacettepe Üniversitesi Öğrenci İşleri Dairesi Bilgi Sistemleri Müdürlüğü
– Akademik Uzman, Yazılım Geliştirici

- Öğrenci Ders Kayıt Yazımı (JSF+PRIMEFACES+EJB+ECLIPSELINK(JPA)+ORACLE DATABASE)
- Öğrenci Bilgi Sistemi YÖKSİS Entegrasyonu(Java + Web Servis)

2012 Haziran –2014 Ekim
Yonca CBS Ltd.Şti., Fırat Teknokent, Elazığ

- Web Tabanlı Coğrafi Bilgi Sistemleri Yazılımları Geliştirilmesi
- Konum Tabanlı, Akıllı Reklam-Tanıtım Platformu ve Kent Ulaşım Bilgi Sistemi - Tübitak Teydeb -1507 Kobi Ar-ge Başlangıç Destek Programı Projesi
- Naviskop Araç Takip Sistemi (http://www.naviskop.com/)
- Servis İzle Akıllı Servis Yazılımı (http://servisizle.com/)
- İmar Bilgi Sistemi
- Lihkabpro Lisanslı Harita Kadastro Büro Otomasyonu (http://www.lihkabpro.com/)

2012 - Fırat Üniversite Bilgisayar Mühendisliği

- Bilgisayar ile kemik yaşı tayini yazılımı (Bitirme Tezi)

2011 - Yonca CBS Bilişim - Fırat Teknokent/ ELAZIĞ (Staj)

2011 - Turgut Özal Araştırma Hastanesi Bilgi İşlem Dairesi MALATYA- (Staj)

2010 - İnönü Üniversitesi Bilgi İşlem Daire Başkanlığı MALATYA (Staj)

EĞİTİM BİLGİLERİ

2012 – 2016 Yüksek Lisans Fırat Üni. Bilgisayar Müh.,

Tez Konusu: **Dağıtık Görüntü İşleme**

2008 – 2012 Lisans Fırat Üniversitesi Bilgisayar Mühendisliği **(Bölüm Birinciliği)**

2004 – 2008 Lise Malatya Orgeneral Eşref Bitlis Lisesi Y.D.A **(Okul Birinciliği)**

YABANCI DİL

İngilizce

YAZILIM VE TEKNOLOJİ BİLGİSİ

- Hadoop, MapReduce, OpenIMAJ, Görüntü İşleme
- Java, JSP/Servlet/JSF, JPA, EJB
- PrimeFaces, Web Servisler, JQuery, JavaScript
- AJAX, JSON, XML, PHP
- C#. DevExpress, XtraReport, CrystalReport
- PostgreSQL, PostGIS, MySQL, Sqlite, Oracle
- CBS, OpenLayers, GeoServer,
- Windows, Linux
- Commetd