T.C.
MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DÜŞÜK MALİYET İLE MİKRO SÜPER BİLGİSAYAR OLUŞTURMA VE APACHE HADOOP ENTEGRASYONU

RECEP ALİ YILMAZ

Yüksek Lisans Tezi

Tez Danışmanı
Doç. Dr. Turgay Tugay Bilgin

İSTANBUL – 2015

T.C.

MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DÜŞÜK MALİYET İLE MİKRO SÜPER BİLGİSAYAR OLUŞTURMA VE APACHE HADOOP ENTEGRASYONU

YÜKSEK LİSANS TEZİ

Recep Ali Yılmaz

Tez Danışmanı

Doç. Dr. Turgay Tugay Bilgin

İSTANBUL – 2015

Bu tez çalışması, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... / ..... / ..... tarih ve ..... / ..... sayılı kararıyla oluşturulan jüri tarafından ……. ……. Yüksek Lisansı Tezi olarak kabul edilmiştir.

JÜRİ

Doç. Dr. Turgay Tugay Bilgin

Danışman

Yrd. Doç. Dr. Ali Akman

Üye

Doç. Dr. Serhat Özekes

Üye

ÖZET

Yüksek Lisans Tezi, Düşük Maliyet İle Mikro Süper Bilgisayar Oluşturma ve Apache Hadoop Entegrasyonu, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Bu tez çalışması günümüzün önemli problemlerinden olan büyük miktardaki veri dosyalarının işlenmesindeki yüksek maliyet ve büyük sistemlere olan ihtiyacın daha düşük maliyetli, az yer kaplayan ve yönetimi kolay ARM mimarili mikro bilgisayarlar ile nasıl yapılabileceğini içermektedir.

Günümüzde Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System (HDFS)) büyük boyutlu veriyi saklama ve işlemede yaygın olarak kullanılmaya başlanmıştır. HDFS paralel işleme için MapReduce programlama modelini kullanmaktadır. Bu sayede büyük verileri sınırsız sayıda genişleyebilen yeni nesil dağıtık mimari sistemler üzerinde işlenebilmektedir. Geleneksel sistemlerde küme (cluster) yapıları kurabilmek için mimarilerin ve sistemlerin aynı olması gerekirken bu tez çalışmasında açıklanan sistemin kurulumu için mimarilerin, sunucuların ve küme içerisine dâhil edilecek düğüm (node) adedinin bir sınırı bulunmamaktadır.

Büyük verilerin yukarıda özetlenen zorluklara rağmen daha kolay ve ekonomik yöntemler ile işlenebileceği açıklanmıştır. Büyük verilerin işlenmesi konusunda yakın gelecekte mikro bilgisayarın daha ön plana çıkacağı beklenmektedir. Bu tez çalışması büyük verilerin mikro bilgisayar sistemleri ile işlenmesi konusunda önemli katkı sağlayacağından araştırmacılar ve uygulama geliştiriciler için referans olma niteliği taşımaktadır.

Anahtar Kelimeler: Büyük Veri, Dağıtık Dosya Sistemleri, Eşle/İndirge, Paralel İşleme.

ABSTRACT

Master's Thesis, Creating micro supercomputers with low cost and Apache Hadoop integration, T.C Maltepe University, Institute of Sciences, Department of Computer Engineering.

This graduate thesis explains how big data proccessing problems that requires too much resources to manage with huge costs can be solved with lower cost, easy to maintain and small take up ARM engineered micro computers.

Nowadays, Hadoop Distributed File System (HDFS) is commonly used for storing and proceesing big data. HDFS uses MapReduce programming model for paralel processing. By the help of this method, big data can be processed by new generation distubuted architecture systems which has capability of getting extention unlimitedly. Although to establish even clustered infrastuctures in conventinal systems, the architectures and system modelling must be same; there is no limitation for architectures, servers or number of nodes in clusters for the systems explained in this thesis.

Despite of the difficulties explained above to process big data, the thesis shows that there are more easy and economic techniques to solve them by the help of microcomputers which will be more widely used in near future than today. Consequently this thesis can be accepted as reference by application developers and researchers.

Keywords: Big data, Distributed File Systems, MapReduce, Parallel Processing.

TEŞEKKÜR

Tez konumu seçmeme yardımcı olan, çalışmaya teşvik eden, bu süreçte yol göstericiliğini ve bilgisini benden esirgemeyen değerli danışman hocam Doç. Dr. Turgay Tugay Bilgin'e, teşekkürlerimi sunarım.

Yüksek lisans dönemi boyunca eğitimime verdikleri destek ve hassasiyetleri için arkadaşlarıma, sevgisi ve ilgisiyle attığım her adımda arkamda olan öncelikle eşim Deniz'e ve aileme saygı, sevgi ve sonsuz teşekkürlerimi sunarım.

Bugünlere gelmemi sağlayan, eğitimim ve ileri yürümemde bana devamlı destek olan, benim hep daha fazla okumamı isteyen, yüksek lisans eğitimimin başlamasına vesile olan ve bu tezi yazmaya başladığımda kaybettiğim babama teşekkür ederim.

Mayıs – 2015 Recep Ali Yılmaz

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER DİZİNİ VE KISALTMALAR

HDFS : Hadoop Distributed File System (Hadoop Dağıtık Dosya Sistemi)

DFS : Distributed File System (Dağıtık Dosya Sistemi)

RPC : Remote Procedure Call (Uzaktan Erişimle Fonksiyon Çağırma)

JAR : Java Archive (Java Arşiv Dosyası)

ARM : Acorn RISC Machine (RISC tabanlı bir işlemci mimarisi)

GPU : Graphics Processing Unit (Grafik İşleme Ünitesi)

CLUSTER : Server cluster (Sunucu Kümele)

SQL : Structured Query Language (Yapılandırılmış Sorgu Dili)

RAID : Redundant Array of Inexpensive Disks (Birden fazla sabit diski kullanarak yapılan veri depolama mimarisi)

1. GİRİŞ

1.1 Tez Çalışmasının Amacı

Günümüzde özellikle internet üzerinde yapılan her türlü işlemlerin artması sebebi ile oluşan verilerin kontrolü, sınıflandırılması, bu verilerden analiz yapılarak yeni avantajların elde edilmesi ve güvenliği çok önemli bir ihtiyaç haline gelmiştir. Artık internet üzerinde çok hızlı bir şekilde oluşan bu verilere büyük veri denilmeye başlanmıştır. Büyük veri olması sebebi ile geleneksel sistemlerin bu büyük verileri mevcut sistemler ile işleyemeyeceği de anlaşılmış durumdadır. Büyük veriler ile çok sayıda dosya da oluşmaktadır ve bu dosyalar standart olarak değil farklı içerik ve yapılarla oluşabilmektedir. Durum böyle olunca geleneksel sistemler ile bu verileri işleyebilmek pek mümkün görünmemektedir.

Apache Hadoop' un bu alanda çok tercih edilen bir dağıtık dosya sistemi (Distributed File System (DFS)) olması nedeniyle özellikle üzerinde araştırma yapılması teşvik edici olmaktadır. Google, Yahoo, Facebook gibi firmaların web sunucularında altyapı olarak HDFS sistemini kullanıyor olması da bu çalışmayı yapmada ayrıca teşvik edici olmaktadır.

Geleneksel sistemler ile bu tür verileri işlemenin zorluğunu göz önüne alarak Apache Hadoop' un desteklediği dağıtık mimari ve bu mimariyi ARM ve birbirinin aynı olması gerekmeyen sistemlerin üzerine kurulması da ayrıca ilgi çekici bir konu olmaktadır. Ülkemizde de bu konuda çalışmalar başlatılmış fakat bu sistemlerin altyapıları yine büyük ve yüksek maliyetli sistemler ile planlanmaktadır. Buna alternatif olarak büyük ve pahalı sistemlere gereksinim duymadan çok düşük maliyetler ile dağıtık bir sistem kurularak büyük sistemlerden daha az enerji ve kaynak tüketerek yine aynı işlem kapasitesine ulaşılabileceğini düşünülmektedir. Bu sebeplerden dolayı bu çalışma başlatılmış ve bu noktaya getirilmiştir.

1.2 Problemin Tanımı

Çok hızlı şekilde ve yapısal olmayan verilerin oluşması sebebi ile bu verilerin nasıl işleneceği konusu günümüzün en önemli konuları arasında yer almaktadır. Bu verilerin işlenmesi sadece alışveriş veya sosyal yaşam için kullanımı dışında güvenlik birimleri, şehirlerin yönetimi gibi her alanda bu verilerden elde edilebilecek bilginin ne kadar önemli olduğu anlaşılmaktadır. Bu verileri işleyebilmek için çok büyük altyapıların kurulmasına ihtiyaç ortaya çıkmış ve bunların da büyük maliyet oluşturduğu bilinmektedir. Bu çalışmada büyük maliyetler yerine düşük maliyetli sistemler ile yine aynı verimin alınabileceği, aynı işlemlerin yapılabileceği ve bunun nasıl yapılabileceğine dair bilgiler açıklanmaktadır.

1.3 Tez Çalışmasının Katkıları

Bu tez çalışmasında daha düşük maliyetli sistemler ile Apache Hadoop dağıtık dosya sistemin ve bu sistem üzerinde paralel işlemleri yapılabileceği, bu işlemler yapılırken çok düşük enerji tüketimi ve mini süper bilgisayar sistemin düşük maliyetler ile kurulabileceği anlatılmaktadır. Bu tür çalışmalar sayesinde önümüzdeki dönemlerde artık sistem odalarında yüksek maliyetli sistemlerin yanında tek bir sunucu kabinine sığabilen yüzlerce ARM tabanlı mikro bilgisayar sistemleri ile çok büyük verilerin işlenmeye ve bu sistemlerin ticari olarak kullanılmaya başlanacağına inanılmış ve buna katkıda bulunulmaya çalışılmıştır.

1.4 Tez Düzeni

Bu tez çalışması, beş ana bölümde sunulmaktadır; Bölüm 2'de ARM tabanlı mimariler kullanılarak süper bilgisayarların oluşturulması ve ARM tabanlı mimarilerde Apache Hadoop ile büyük verilerin işlenmesi ile ilgili yapılmış çalışmalar hakkında bilgiler sunulmuştur. Bölüm 3'de teknik altyapı, sistemler ve bu sistemlerin kurulumu anlatılmıştır. Bölüm 4'te yazılım altyapısı, Apache Hadoop, bileşenleri ve mimari yapısı anlatılmıştır. Bölüm 5'te oluşturulan bu sistemin yani ARM tabanlı mini süper bilgisayarın test edilmesi ve performans ölçümlerinin yapılması anlatılmıştır. Yapılan tez çalışmasının sonuçları ve katkıları Bölüm 6'da değerlendirilmektedir.

2. İLGİLİ ÇALIŞMALAR

Büyük verilerin küme sistemleri ile işlenmesi konusunda bazı ülkelerde çalışmalar yapılmış ve makaleler yayınlanmıştır. Büyük veri problemi güncel bir kavram olması sebebi ile buna yönelik çalışmalarda çok eski sayılmamaktadır. Açık kaynak kodlu çalışmaların büyük firmalar tarafından desteklenmesi ile çok hızlı bir şekilde bu konuda çalışmalar yapılmaya başlanmıştır. Bu tez çalışmasının gerçek amacı büyük veri işlemleri gibi çalışmalarda kullanılan yüksek maliyetli donanımlar yerine çok düşük maliyetli donanımlar ile bunun yapılabileceğinin gösterilmesidir. Bu konuya yönelik bazı çalışmalar aşağıda açıklanmaktadır. Bu konudaki çalışmalar Hadoop ve Süper bilgisayar sistemleri olarak iki başlıkta incelenmektedir.

Wenhui Lin ve Jun Liu [1] MapReduce programlamanın bulut bilişimdeki yerini ve bunun performansını şu şekilde açıklamıştır; Hadoop araştırması bulut bilişim sanayinin bir bölümü olup Hadoop performans araştırması önemli bir araştırma yönergesidir. Hadoop performans analizi temel bir çalışma olarak diğer performans araştırmaları konusunda önemli referanslar sağlayabilmektedir. Sunucu performans analizinin önceki araştırmalarına dayanan bu araştırmada Hadoop ile ilgili olarak düğüm performans ölçüm yöntemi öne sürülmektedir. Heterojen bir Hadoop kümesindeki her bir düğümün performans değerinin nasıl ölçüleceğini detaylı olarak tanımlamakta ve ölçüm sonuçlarını MapReduce programlarını çalıştırmak suretiyle değerlendirilmektedir. Bu sırada, yöntem gerçek dünya Hadoop kümesinde de uygulanmış ve değerlendirilmiştir. Deney sonuçları yöntemin her bir düğümün performans değerini doğru şekilde ölçebildiğini göstermektedir. Bu araştırmaya dayanılarak kullanıcılar kendi Hadoop kümelerine ilişkin kapsamlı ve nesnel bir anlayış sağlayabilir ve sonrasında Hadoop ile ilgili olarak optimizasyon ve geliştirme işlemleri yapılabilir.

Ali Anwar, Krish K.R. ve Ali R. Butt [2] Hadoop sisteminin mikro süper bilgisayar (ARM tabanlı mini bilgisayarlar) üzerindeki araştırması şu şekildedir; Sayısız uygulamayı desteklemede, gömülü ana işlemcileri olan ekonomik ve düşük güçlü

mikro sunucuların kullanımında artış görülmektedir. Teknoloji harikası mikro sunucular alt sınıf geleneksel sunucuların performansına çoktan erişmiştir ve veri merkezlerinin bilişimi konusunda enerji tasarruflu bir alternatif olarak ortaya çıktığı belirtilmektedir. Bu çalışmada, mikro sunucular içeren kümenin popüler Hadoop platformunu destekleyip destekleyemediğini araştırılmaktadır. 6 temsili Hadoop uygulamasıyla beş donanım konfigürasyonu üzerinde nicel bir çalışma yürütülmektedir. Farklı kümeleri karşılaştırmak için, performans enerji tüketimi ve uygulamaların edinme maliyetini de içeren ve Hadoop uygulamalarına yönelik uygun kümelerin tanımlanmasına yardım eden kapsamlı bir ölçü, PerfEC tanımlanmıştır. Test kümelerindeki deneyler, TeraSort, RandomWriter ve Grep gibi uygulamalar için mikro sunucuların geleneksel kümlere göre PerfEC açısından daha iyi etkinlik sağlayan iki boyut sırası ortaya koyduğunu gösterilmektedir. Benzer şekilde Facebook'tan elde edilen gerçek dünya güdümlü 3000 düğümlü küme simülasyonu, üzerinde çalışılan mikro sunucuların ortalama olarak standart sunucuların performansıyla eşleşebildiğini ve edinme maliyetinin sadece % 60'ında % 31'e kadar enerji tasarrufu sağlayabildiğini gösterilmektedir. PerfEC ile geçerliliğini hala koruyan Toplam Mülkiyet Maliyetini (TCO) de karşılaştırmakta ve bu çalışmada dâhil olan değiş tokuşları daha iyi yakalayabildiği sonucuna varılmaktadır.

Shaun Franks ve Johnathan Yerby [3] Düşük maliyetli Raspberry Pi cihazları üzerindeki araştırması şu şekildedir; Bu makale Raspberry Pi kullanılarak bir süper bilgisayar sisteminin oluşturulmasını açıklamaktadır. Geleneksel hesaplama tek bir seferde tek bir talimatla ve genellikle tek işlemci kullanılarak yapılmaktadır, ancak açık kaynaklı yazılım kullanılarak yaratılan kümede paralel hesaplama ve dağıtımlı depolamanın mümkün olduğu açıklanmıştır. Görevlerin tamamlandığı hız, verinin donanım içerisinde ne kadar hızlı hareket ettiğine bağlı olduğu belirtilmektedir. Paralel hesaplama, büyük görevlerin küçük parçalara bölünmesi ve verinin eş zamanlı olarak işlenmesinde koordineli bir çaba gösterilmesi suretiyle, görevlerin işletilmesi için daha hızlı bir yol olduğu belirtilmektedir. Paralel hesaplama tipik olarak, maliyetleri milyon dolarlarla milyar dolarlar arasında değişen süper bilgisayarlarca yapıldığı açıklanmaktadır. Bu çalışma düşük maliyetli süper bilgisayarlar yaratmanın tasarımı,

zorlukları ve başarısı ile sanayide ve daimi öğrenmede kullanılmak üzere yaratılma olasılığını detaylandırmaktadır.

Dumitrel Loghin, Bogdan Marius Tudor, Hao Zhang, Beng Chin Ooi ve Yong Meng Teo [4] ARM tabanlı küçük sistemler üzerinde Büyük veri uygulamalarına ait performans araştırması şu şekildedir; Büyük Veri'nin hacmindeki, çeşitliliğindeki ve hızındaki daimi artış, veri merkezi kaynak ölçeklendirmesini enerji kullanım problemine maruz bırakmaktadır. Geleneksel olarak, veri merkezleri onlarca ila yüzlerce Watt arasında değişen güç kullanımına sahip x86-64 (büyük) sunucu düğümleri kullanırlar. Ancak son zamanlarda, aslında mobil cihazlar için geliştirilmiş olan düşük güçlü (küçük) sistemler performans açısından önemli gelişmeler kaydetmiştir. Bu gelişmeler, büyük sanayi oyuncuları tarafından belirtildiği gibi bu küçük sistemlerin sunuculara adapte edilmesine imkân sağlamaktadır. Bu bağlamda, Büyük Verinin geleneksel büyük sunuculara nazaran küçük sunuculardaki performansına yönelik sistematik çalışmalar yürütülmekte ve ileriye yönelik gelişimde yararlı olabileceği düşünülmektedir. ARM big.LITTLE boards ve Intel Xeon sunucu sistemlerinin kümeleri üzerindeki Hadoop MapReduce, MySQL ve iç bellek Shark iş yükleri çalıştırılmaktadır. Bunlara ilişkin zaman ve enerji kullanımı ile selfhosted ARM ve Xeon düğümlerindeki iş yüklerinin çalıştırılmasının toplam maliyetini değerlendirilmektedir. Bu araştırma büyük veri iş yüklerinin küçük ve büyük düğümler üzerinde yürütülmesinin etkinliğine karar vermek için bütün kurallara uyan tek bir şey olmadığını göstermektedir. Ancak; küçük bellek boyutu, düşük bellek ile I/O bant genişliği ve yazılımın olgunlaşmaması, ARM sunucularının daha düşük güç avantajını geçersiz kılmaktadır. I/O yoğun MapReduce iş yüklerinin Xeon düğümlerinde çalıştırılmasının daha çok enerji tasarruflu olduğunu gösterilmektedir. Bunun aksine, veri tabanı sorgulama işlemi de her zaman ARM sunucuları üzerinde daha çok enerji tasarruflu ve daha az maliyetli olmaktadır. Küçük yazılım modifikasyonlarıyla, ana işlemci yoğun MapReduce iş yüklerinin ARM sunucuları üzerinde yürütülmesi neredeyse 4 kat daha ucuz hale geldiği açıklanmaktadır.

3. TEKNİK ALTYAPI

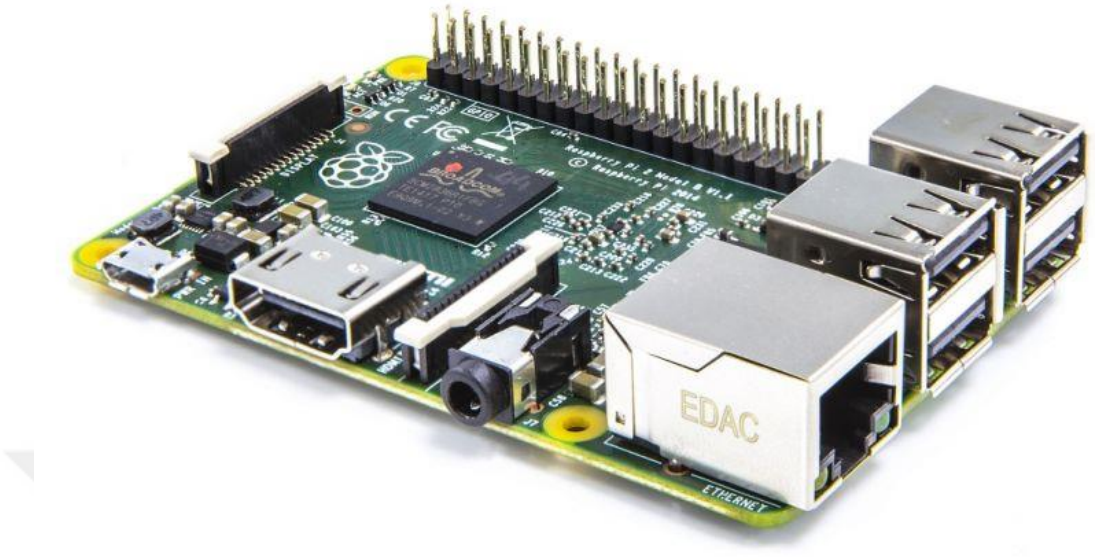
3.1 Raspberry Pi

Raspberry Pi; İngiltere' de Raspberry Pi Vakfı tarafından okullarda bilgisayar bilimini öğretmek amacıyla geliştirilmiş kredi kartı büyüklüğünde tek kartlı bir bilgisayardır [5]. Türkiye' de geliştirilen PardusARM sürümünün ilk çalıştığı modeldir [6].

Raspberry Pi cihazı uluslararası birkaç firma tarafından üretilmektedir. Bu tez çalışması kapsamında kullanılan cihazlar "RS Components" firmasının üretimidir. Raspberry Pi cihazlarının internet üzerinden de satışları yapılmaktadır ve fiyatı 35 ABD doları şeklindedir. Türkiye'deki fiyatı ise Raspberry Pi 2 Model B sürümü için 120 TL şeklindedir.

Raspberry Pi cihazı ilk sürümü ile birlikte 700 MHz işlemci (CPU) gücü ile üretilmiştir. CPU üretimi ve modeli ilk olarak Broadcom BCM2835 (700MHz, ARM1176JZF-S tabanlı) mikroçipi mimarisi üzerine kurulmuştur. İlk sürümünde 256 megabayt bellek ile üretilmiştir. Daha sonraki sürümlerinde bellek kapasitesi 512 megabayta kadar çıkarılmıştır. Üzerine entegre edilmiş bir sabit diske sahip değildir. İşletim sisteminin kurulması ve veri depolama işlemleri için Micro SD kart kullanılmaktadır. İlk modeli 29 Şubat 2012 tarihinde çıkarılmıştır. İkinci sürümü ise 4 Şubat 2013 tarihinde piyasaya çıkarılmıştır. Son güncel sürümü ve bu tez çalışmasında kullanılan Model B 2 ise Şubat 2015 tarihinde piyasaya sürülmüştür. Raspberry Pi 2 Model B sürümü 900MHz quad-core ARM Cortex-A7 CPU ve 1GB RAM ile üretilmiştir.

Cihazın üzerine kurulabilecek farklı işletim sistemleri bulunmaktadır. Bu işletim sistemlerine ait kurulum dosyaları ve imajlar web üzerinden ücretsiz olarak yayınlanmaktadır.



Şekil 1.Raspberry Pi 2 Model B Cihazının genel görünümü.

Son olarak üretilen Raspberry Pi 2 Model B cihazı önceki sürümlerine göre en büyük farkı Windows 10 işletim sistemi ile çalışabilmesi ve donanım kapasitesinin iki katına çıkarılmasıdır.

3.2 Teknik Özellikleri

Raspberry Pi 2 Model B cihazının teknik özellikleri tablo 1' de ki gibidir.

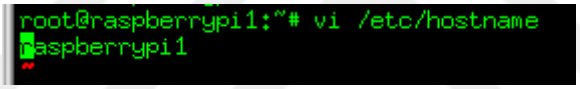
İşlemci	Dört Çekirdekli ARM Cortex-A7 CPU (900MHz)
Dahili Hafıza	1GB SDRAM – 450 MHz
USB Bağlantıları	4 x USB 2.0 soketi
Görüntü Bağlantısı	1080p HDMI 1.3 soketi
Karma Bağlantı	PAL/NTSC çıkışları
Ses	Stereo ses çıkışı
Bellek	MicroSD Kart soketi
Güç	+5V / 2A microUSB soketi
Ebatlar	86 mm x 56 mm x 20 mm
Ağ Bağlantısı	10/100 BaseT RJ45 Ethernet soketi
Giriş / Çıkış Bağlantıları	40 noktalı GPIO Giriş/Çıkış genişleme yuvaları

Tablo 1.Raspberry Pi 2 Model B cihazının Teknik Özellikleri.

3.3 Ağ Ayarları

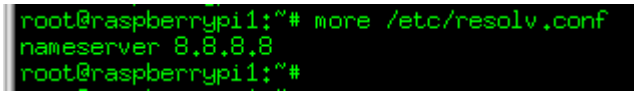
Raspberry Pi cihazları kablolu veya kablosuz olarak ağlara bağlanabilirler. Kablosuz bağlantı için USB portunu kullanarak kablosuz ağ adaptörleri kullanılmaktadır. Kablosuz bağlantılar için mini wifi adaptörleri bulunmaktadır.

Raspberry Pi ağ ayarları monitör bağlantısı var ise görsel ara yüzden de yapılabilmektedir. Bu çalışma kapsamında gerekli ayarlar terminal bağlantısı üzerinden yapılmaktadır. Cihazın tüm ağ ayarları ve buna benzer sistem ayarlarının yapılabilmesi için root user ile bağlanılması veya komutları çalıştırırken başında sudo komutunun eklenmesi gerekmektedir. Cihazın hostname bilgisinin düzenlenmesi için "vi /etc/hostname" komut satırını kullanılarak hostname dosyasında istenilen değişiklik yapılıp kaydedilmesi gerekmektedir.



Şekil 2.Hostname dosyasının içeriği.

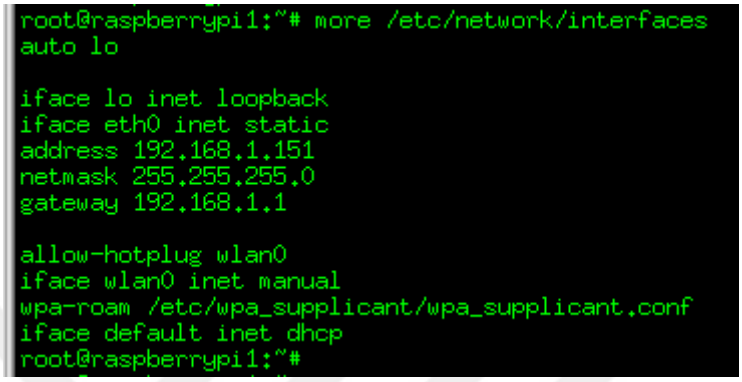
Cihazın DNS sunucu ayarlarını "/etc/resolv.conf" dosyasında tanımlanmaktadır. Bu dosya vi editörü ile açılması ve gerekli düzenlemelerin yapılması gerekmektedir. Bu çalışmada DNS sunucu olarak Google DNS sunucuları kullanılmaktadır. Projelerin ihtiyaçlarına göre Raspberry Pi cihazları üzerinde de DNS servisleri aktif edilebilir ve master cihaz DNS servis hizmeti verebilir bu durumda ilgili cihazın IP adresi DNS sunucu olarak tanımlanması gerekmektedir.



Şekil 3.DNS sunucu ayarlarının tutulduğu resolv.conf dosyasının içeriği.

Ağ ayarları "/etc/network/interfaces" dosyasında tutulmaktadır. Cihaz iki şekilde ayarlanabilir, otomatik olarak ip adresi atanabilir veya sabit bir ip adresi kullanılarak

çalıştırılabilmektedir. Sabit adres tanımlanması durumunda "/etc/network/interfaces" dosya içeriği şekil 4'deki gibi olacaktır. Buradaki bilgiler sırasıyla sabit ip adresi ve ağ maskesi, ağ geçidi bilgisidir.

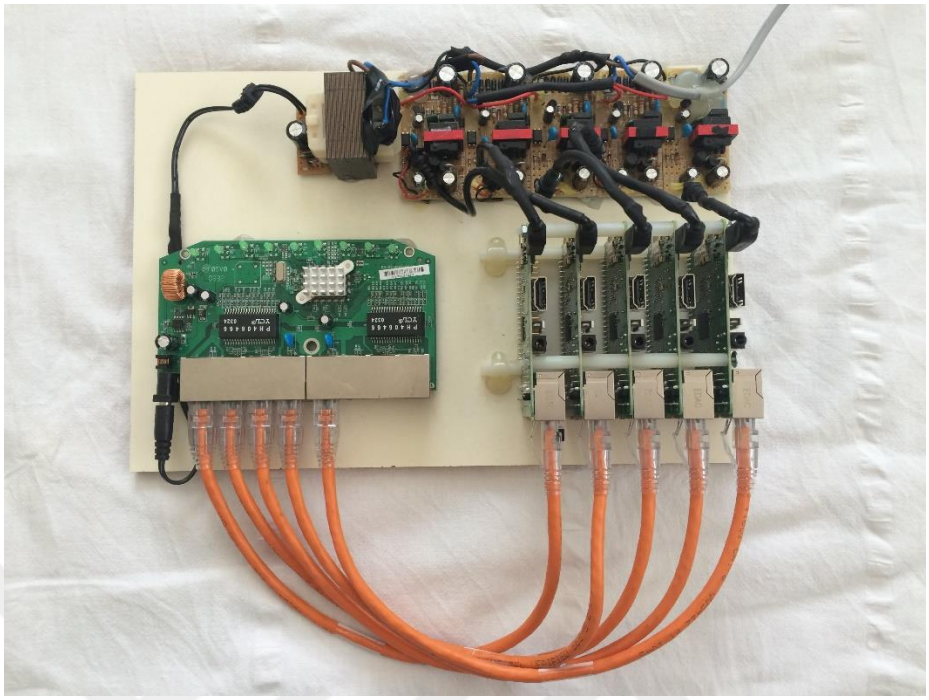


Şekil 4.Ağ ayarlarının tutulduğu interfaces dosyasının içeriği.

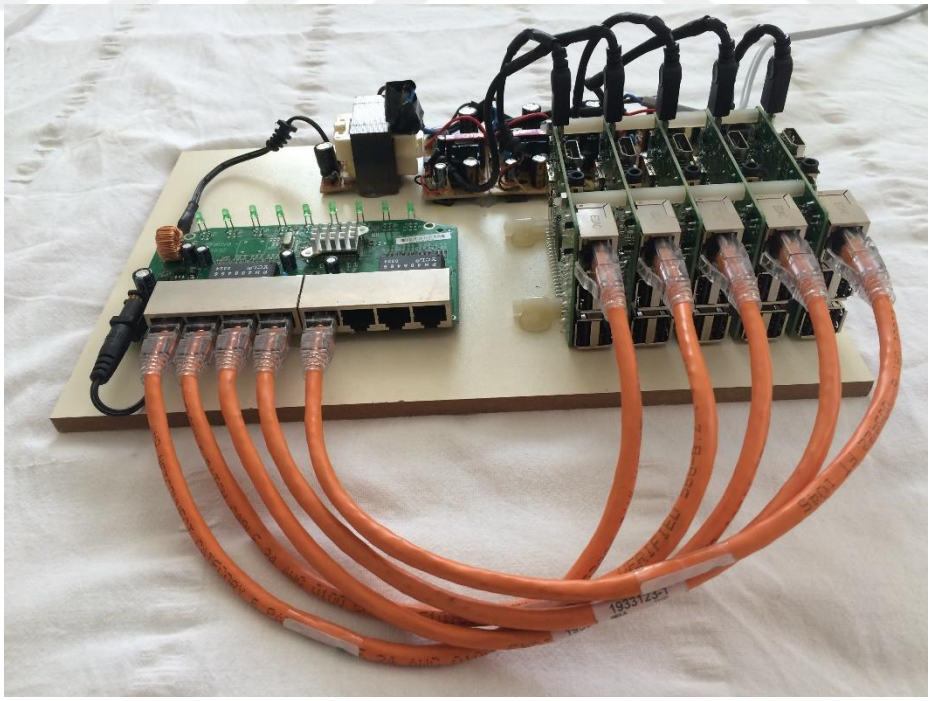
3.4 Test Sisteminin Tasarlanması

Bu tez çalışması için 5 adet Raspberry Pi 2 Model B cihazı kullanılmaktadır. Yapıya istenilen sayıda ek Raspberry Pi cihazı eklenebilir. Sistemin oluşturulması aşamasında 5 adet Raspberry Pi, 5 adet 2A 5V adaptör, 1 adet 6 port network switch, 1 adet 9V switch adaptörü ve işletim sistemi kurulumu aşamasında ihtiyaç duyulması sebebi ile HDMI bağlantılı monitör kullanılmaktadır.

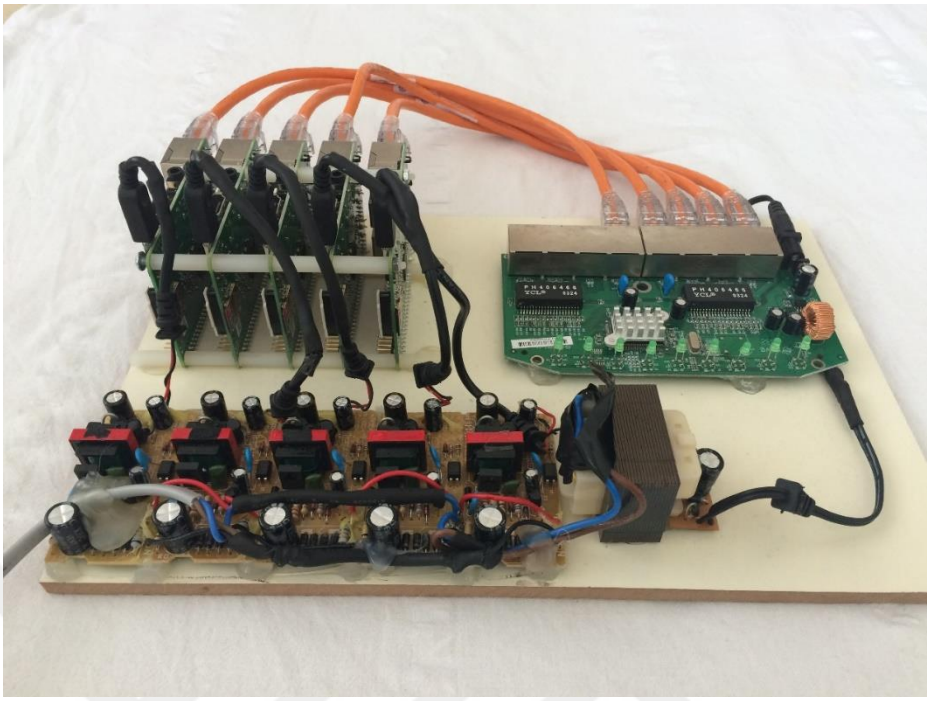
Sabit bir zemin üzerine parçaların bağlantıları yapılarak sistem çalışır hale getirilmektedir. Sistemin genel görünümü şekil 5, şekil 6 ve şekil 7'de gösterilmektedir. Sistemin mimari çizimi şekil 8' deki gibidir. Raspberry Pi cihazlarının her birinden birer data kablosu ile ağ switch' ine bir bağlantı yapılmaktadır. Bu cihazların hepsi aynı switch üzerindeki ağ üzerinde çalışabilecek duruma getirilmesi gerekmektedir. Switch' in boş portlarını kullanarak cihazlara internet erişimi sağlanmaktadır. Boş portlar kullanılarak bilgisayar üzerinden yine bu sisteme ait ağa bağlanılarak gerekli çalışmalar yapılmaktadır.



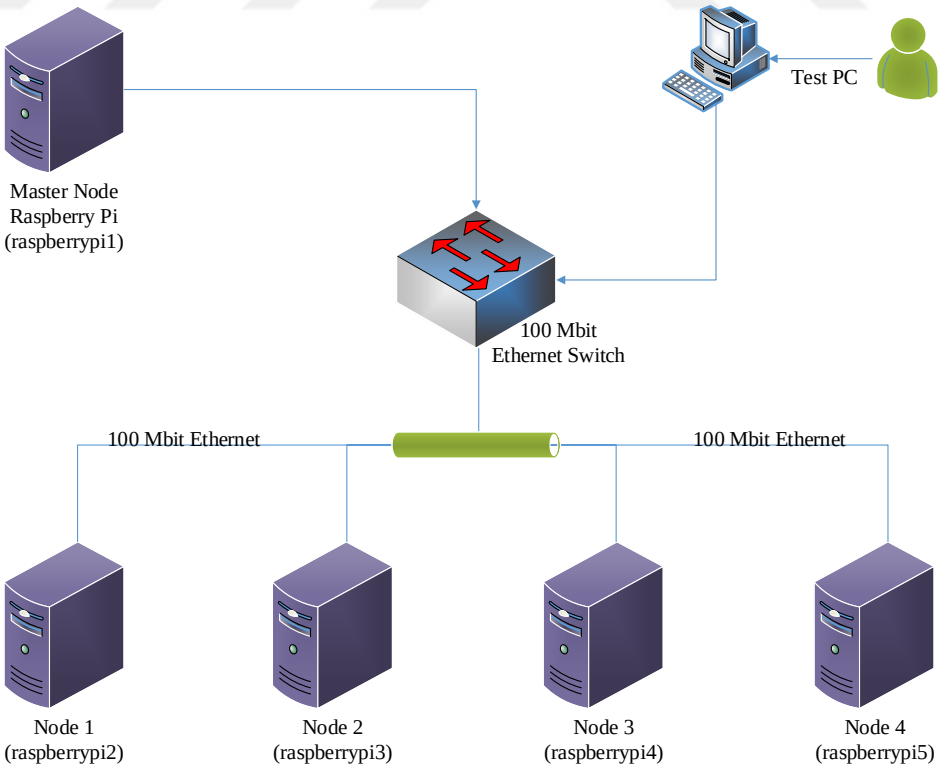
Şekil 5.Tez kapsamında hazırlanan cihazın genel görünümü.



Şekil 6.Tez kapsamında hazırlanan cihazın genel görünümü.



Şekil 7.Tez kapsamında hazırlanan cihazın genel görünümü.



Şekil 8:Test sisteminin mimarisi.

4. YAZILIM ALTYAPISI

4.1 Büyük Veri

4.1.1 Büyük Veri Kavramı

Büyük Veri tanımı sadece diskte çok fazla yer kaplayan veri anlamına değil, aynı zamanda geleneksel yöntem ve araçlarla işlenemeyen veri anlamına da gelmektedir. Teknolojinin ilerlemesi, internetin gelişmesi ve sosyal medyanın her gün artan kullanım hızı sayesinde bilginin gücünün öne çıkması iş yapma şekillerinin kökten değişmesine sebep olmaktadır. Şirketler bu yeni zamanda bir adım öne geçebilmek için farklı yöntemler kullanarak öne çıkmaları gerekmektedir. Bu sebeple artık en ufak bir bilginin bile ne kadar önemli olduğu anlaşılmaktadır [7].

Büyük veri kavramının yaygınlaşması rekabetin bu kadar çok olduğu günümüzde firmaların bir adım daha ileriye gidebilmesi ve farklılık yaratmak istemeleri olmaktadır. Bunun yanı sıra sadece firmaların kazançları, satışlarının arttırılması gibi ticari yönden değil güvenlik birimlerinin, istihbarat birimlerinin veya şehircilik alanında da bu verilerin kullanımı büyük avantajlar sağladığı görülmektedir. Burada belirtilmeyen daha birçok alanda büyük veri ile elde edilebilen bilgilerden fayda sağlandığı görülmektedir.

Büyük verileri oluşturan kaynakları ise hayatın her alanında görmek mümkündür. Bu büyük verileri internette gezinirken, araç kullanırken, telefonların kullanımında, alışveriş yapılırken, bir market içerisinde reyonlar arasında dolaşırken bile üretilmektedir.

Teknolojinin ilerlemesi ile birlikte hem sistemlerin hızlanması hem de kapasitelerinin artması gün geçtikçe daha fazla olmaktadır fakat bununla birlikte daha fazla kaynağa ihtiyacın olması sebebi ile fiyatlar düşse bile yine de bu kadar büyük verilerin depolanabileceği sistemlerin kurulması oldukça maliyetli olmaktadır.

2000 yılında tüm dünyada 800,000 petabyte büyüklüğünde veri saklandığı belirtilmektedir. Örneğin Twitter her gün 12 TB, Facebook 500 TB ve bazı kurumlar her gün her saat TB'larca veri saklamaktadır [8].

Örnek olarak büyük bir internet servis sağlayıcı veya bir arama motoru hizmeti veren firma ele alınırsa günlük üretilebilecek tıklanma işlemlerine ait verinin ne kadar büyük olabileceği tahmin edilebilmektedir. Bu kadar büyük ve hızlı bir şekilde akan veriyi ayrıca yapısal olmaması durumunu da göz önüne alırsak bu verileri saklamak ve bunlar üzerinden analiz yapılması işlemlerini geleneksel sunucu mimarileri ve yapıları ile sağlamak hem çok yüksek maliyetler getirecektir hem de yapısal olmayan bu verilerin geleneksel sistemler üzerinden işlenemeyeceğini göstermektedir. Büyük verileri ancak büyük veri için hazırlanmış sistemler ve bu sistemlerin getirdiği çözümleri kullanarak ayrıca bu sistemler ile yapısal olmayan bu veriler işlenebilir ve fayda sağlanabilir.

Google' ın başarıları arasındaki en önemli farklardan birisi büyük verilerin işlenmesi için klasik yöntemleri kullanmak yerine ihtiyaç duyduğu teknolojileri kendisi geliştirerek büyük bir başarı yakalamıştır. Google milyarlarca internet sayfasının verisini kendi geliştirdiği Google File System üzerinde tutmaktadır. Google bu verileri işlemek için kullandığı veri tabanı sistemi ise Big Table sistemidir. Büyük verileri dağıtık mimariler üzerinde işlemek için MapReduce mimarisinden faydalanmaktadır. Bu teknolojilerin hepsi düşük maliyetli binlerce bilgisayarın bir araya gelerek oluşturduğu kümeler üzerinde çalışmaktadır.

Google, Amazon, Yahoo gibi firmalar geliştirdiği teknolojiler ve yaptıkları çalışmalar ile ilgili akademik makaleler yayınlamaktadırlar. Bunların en güzel örnekleri genelde Apache projeleri olarak ortaya çıkan Lucene, Solr, Hadoop, HBase gibi projeler ve her biri büyük veriyi iyi bir şekilde işleyebilen ve yönetebilen başarılı projelerdir.

Bilgi teknolojileri alanında çalışmakta olan bir şirket sistemlerinde yapılan her işleme ait hareketleri kaydedip, "hangi hatalar birbirleriyle ilişkili", "hangi problem sistemin performansını ne kadar etkiliyor" gibi sorulara cevap bulmaktadır. Ayrıca bu bilgiler ışığında fayda sağladığı belirtilmektedir. Bir firma internet sitelerini gezen

kullanıcılarının tüm tıklama işlemlerini kaydederek “hangi ürünleri gezdi”, “hangi ürünleri sepete atıp satın almadan işlemi bitirdi” gibi sorulara cevap bulabilir ve buna göre kişiye özel kampanyalar düzenleyebilir. Bir banka, müşterilerin hareketlerinden dolandırıcılık teşebbüsünü tespit edebilir. E-posta servis sağlayıcı tüm e-postaları analiz edilerek hangilerinin spam olduğu tespit edebilmektedir [9].

Büyük veri üzerinde çalışmalar başlatıldığında karşılaşılan problemleri aşağıdaki gibi açıklanmaktadır;

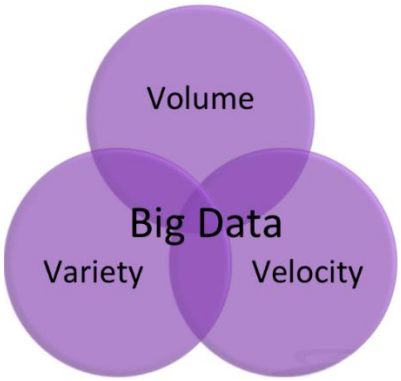
- Verinin nasıl saklanacağı problemi
    - Büyük verinin saklanması için kullanılacak donanım altyapısı
    - Büyük veriyi saklayacak geleneksel veri tabanı çözümlerinin yetersizliği

- Verinin işlenmesi problemi
    - İşleme sırasındaki hafıza problemleri (geleneksel sistemlerdeki Memory yetersizliği)
    - İşleme sırasındaki süre problemleri (sürekli akan gerçek zamanlı uygulamalardaki sürenin kritiklik durumu)

- Verinin yapısal olmaması problemleri
    - Büyük verilerin her alandan gelebilmesi ve standart olmaması sebebi ile geleneksel veri tabanları üzerine bu verilerin aktarılamaması

Yukarıdaki maddeler kapsamında, büyük veri kavramını ve hangi veriler büyük veridir sorusuna cevap bulunabilmesi için genel olarak veri üzerinde 3 farklı boyutta inceleme yapılmaktadır.

Şekil 9’ da bulunan grafik büyük veriyi tanımlamak amacı ile geliştirilen 3V formülünü göstermektedir.



Şekil 9.Büyük Veriyi tanımlayan 3V maddelerinin grafiksel görünümü [10].

- Volume (hacim) : Verinin kapladığı alan.
- Velocity (hız) : Verinin değişim veya birikme hızı.
- Variety (çeşitlilik) : Verinin geldiği kaynakların çeşitliliği (email, facebook, videolar, resimler, ses kayıtları vb.).

4.1.2 Büyük Veri İçin Kullanılan Araçlar

Büyük veri için kullanılan birçok araç ve uygulama bulunmaktadır. Open source veya yazılım firmaları tarafından lisanslı olarak geliştirilenler de bulunmaktadır. Tablo 2' de kategorize edilmiş bazı uygulamaların listesi gösterilmektedir.

Altyapı ve Dosya Sistemi
Apache Hadoop
Apache MacReduce
Apache Storm
Veri Tabanı
Apache Hbase
MongoDB
Programlama ve Veriye Erişim
Apache Hive
Apache Pig

Tablo 2.Büyük veri için kullanılan araçların ve uygulamaların listesi.

4.2 Apache Hadoop

4.2.1 Hadoop Mimarisi

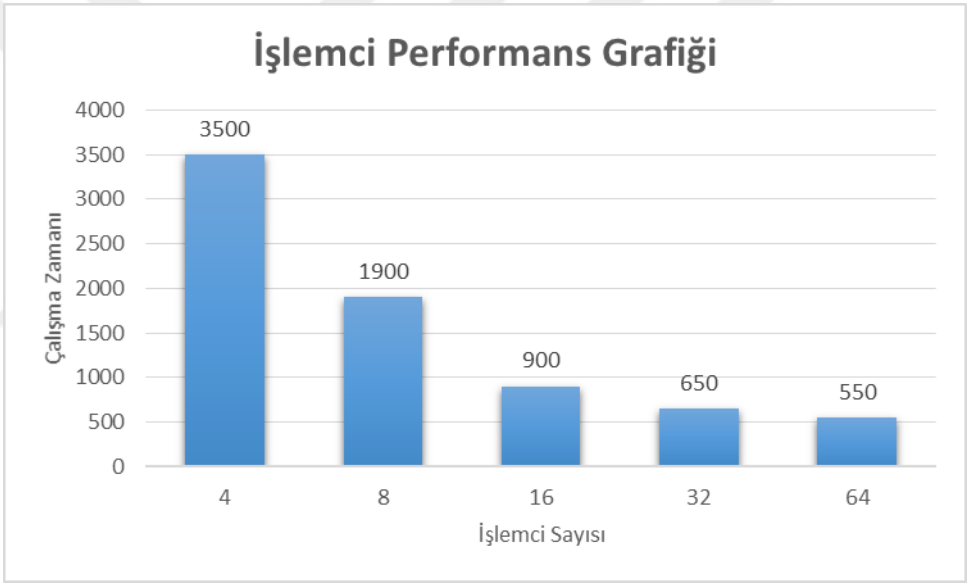
Apache Hadoop çok sayıda hatta sayı sınırı olmadan sıradan sunucuları kullanarak oluşturulan küme (cluster) yapısı üzerinde büyük verilerin işlenmesi için kullanılacak uygulamalara altyapı sağlayan ve Hadoop Distributed File System (HDFS) olarak adlandırılan bir dağıtık dosya sistemidir. Bu tanıma ek olarak Apache Hadoop üzerindeki bileşenlerden olan MapRaduce ile birlikte Java ile geliştirilmiş açık kaynaklı bir kütüphanedir.

Hadoop dosya sistemi olan HDFS sayesinde sıradan sunucuların diskleri kullanılarak bir araya getirilerek bunlardan büyük ve tek bir disk alanı oluşturulmaktadır. Bu sayede çok büyük boyuttaki birçok dosya bu dosya sisteminde saklanabilir, yedeklenebilir ve bütün bunlar verinin güvenliğini de göz önüne alarak yapılabilmektedir. Bu dosyalar bloklar halinde (varsayılan 128 MB) birden fazla ve farklı sunucu üzerine (istenilen sayıda kopya olarak) dağıtılarak RAID benzeri bir yapıyla yedeklenmektedir. Bu sayede veri kaybının önüne geçilmektedir. Hadoop HDFS, NameNode ve DataNode alt servisleri ile oluşmaktadır.

NameNode ana (master) süreç olarak blokların sunucular üzerindeki dağılımından, yaratılmasından, silinmesinden, bir blokta sorun meydana geldiğinde yeniden oluşturulmasından ve her türlü dosya erişiminden sorumludur. Kısacası HDFS üzerindeki tüm dosyalar hakkındaki bilgiler (metadata) NameNode tarafından saklanır ve yönetilir. Her kümede yalnızca bir adet NameNode olabilir [11].

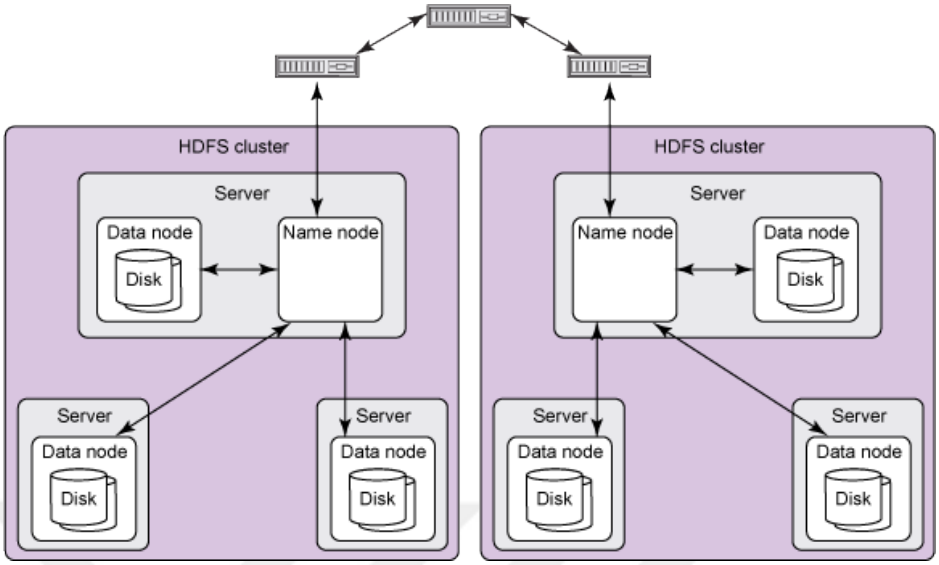
DataNode işlevi blokları saklamak olan işçi (slave) süreçtir. Her DataNode kendi yerel diskindeki veriden sorumludur. Ayrıca diğer DataNode'lardaki verilerin yedeklerini de barındırır. DataNode'lar küme içerisinde birden fazla olabilir [12]. Hadoop MapReduce işlemi ise büyük boyuttaki ve çok sayıdaki dosyaların düğümler üzerine dağıtılarak işlenmesini sağlamaktadır. Reduce işlemi ise Map işleminden sonra oluşan sonuçların toplanarak bir araya getirilmesi ve sonuçların çıkartılması aşamasıdır.

Bu işlemler yapılırken Hadoop' un avantajlarından bir tanesi de Map ile dağıtılan işlerdeki dosyaların trafiğinin ağ üzerinden gitmemesidir. Yukarıda açıklandığı gibi HDFS dosya sistemine atılan bir dosya belirlenen kopya sayısına göre ilgili veri düğümler üzerine atılmaktadır başlatılan bir işe ait dosyalar yerel disklerde olduğundan işlemin çalışması sırasında sürekli olarak ağ üzerinden bir veri trafiği geçmemiş olmaktadır. Bir diğer avantajı ise birden fazla işi aynı anda işleyebilmesi ve sonucu tek bir Reduce işlemi ile tamamlayabilmesidir. Yani şekil 10' da ki grafikte olduğu gibi Hadoop kümesindeki düğüm sayısı arttıkça performansı da doğrusal olarak artmaktadır.



Şekil 10.Hadoop işlemlerinin CPU sayısına göre performans grafiği [13].

MapReduce, JobTracker ve TaskTracker süreçlerinden oluşur. JobTracker yazılan MapReduce programının küme üzerinde dağıtılarak çalıştırılmasından sorumludur [14]. Başlatılan bir işe ait işlerin düğümler üzerinde atanması ve çalıştırılmasından sorumludur. Bu işlerin çalışması sırasında oluşabilecek bir problem durumunda işin farklı bir düğüm ile devam ettirilmesi işlemlerini yerine getirmektedir.



Şekil 11.Hadoop HDFS sistem mimarisi [15].

4.2.2 Hadoop Bileşenleri

Apache Hadoop sistemini büyük ölçekli firmalardan Facebook, Yahoo, Amazon, Twitter, Linkedin gibi birçok firma kullanmaktadır. Bu firmaların ortak özelliği ise büyük verileri analiz etmek amacıyla Hadoop sistemini kullanmalarıdır. Ayrıca Hadoop projesini geliştiren ücretli ve lisanslı sürümlerini yayınlayan firmalar bulunmaktadır. Bu konuda eğitim ve danışmanlık hizmetleri sunan Hortonworks, Cloudera, MapR gibi firmalar mevcuttur.

Bunlar dışında Hadoop projesi büyük verileri işleme konusundaki diğer projelere bir çatı görevi görmektedir. Hadoop bir "ecosystem" olarak da adlandırılmaktadır. Bu kapsamda geliştirilen diğer projeler ise Avro (serialization) sistemi, Cassandra (yüksek erişilebilir, ölçeklenebilir NoSQL veritabanı), HBase (Hadoop üzerinde çalışan, büyük veriler için ölçeklenebilir, dağıtık NoSQL veri tabanı), Hive (büyük veriler üzerinde iş zekası sistemi), Mahout (ölçeklenebilir yapay öğrenme (machine learning) ve veri madenciliği kütüphanesi), Pig (paralel hesaplamalar için yüksek düzeyli bir veri akışı dili ve yürütme kütüphanesi), ZooKeeper (dağıtık uygulamalar için yüksek ölçekli koordinasyon uygulaması) projeleri şeklindedir [16]. Bu tez çalışmasında Apache

Hadoop sistemine ek olarak Hadoop bileşenlerinden, Apache Pig ve Apache Hive uygulamaları incelenmiş ve kullanılmaktadır.

4.2.3 Hadoop HDFS

Hadoop HDFS; verilerin kaydedilmesi için kullanılan dağıtık (birden çok bilgisayar üzerinden çalışan) bir dosya sistemidir. HDFS bilgisayar sistemlerinin geleneksel sistemlerdeki küme (cluster) yapılarındaki gibi birebir aynı olması gerekmemektedir bu yapı sayesinde farklı özelliklerde farklı işletim sistemlerinde farklı boyutlarda ve mimarilerde çalışan sistemler kullanılabilmektedir. Hadoop HDFS'nin en büyük özelliklerinden biri ise sıradan birçok sunucunun diskleri bir araya getirilerek büyük boyutlara sahip tek bir sanal disk oluşturulabilmesidir. Bu sayede ucuz diskler ile sınırsız kapasitelerin elde edilmesi mümkündür. HDFS'nin asıl ortaya çıkış sebebini bu özelliği oluşturmaktadır. Bu yüzden pek çok büyük internet sağlayıcıları Hadoop kullanmaktadır.

4.2.4 Hadoop Sisteminin Avantajları

- Ölçeklenebilir olması; Yeni düğümler sisteme devamlı eklenebilir ve bunların hepsinde aynı sürüm sistemin çalışması gibi bir zorunluluk bulunmamaktadır.

- Düşük maliyet; Hadoop mimarisi standart sunucular ile kurulabilmektedir.

- Esnek; Hadoop şema gerektirmez, her türlü veri kullanılabilir.

- Hata Toleranslı; MapReduce mimarisi sayesinde, hata alan bir job olursa bunu otomatik olarak düzeltmektedir.

4.2.5 HDFS NameNode ve DataNode yazılımları

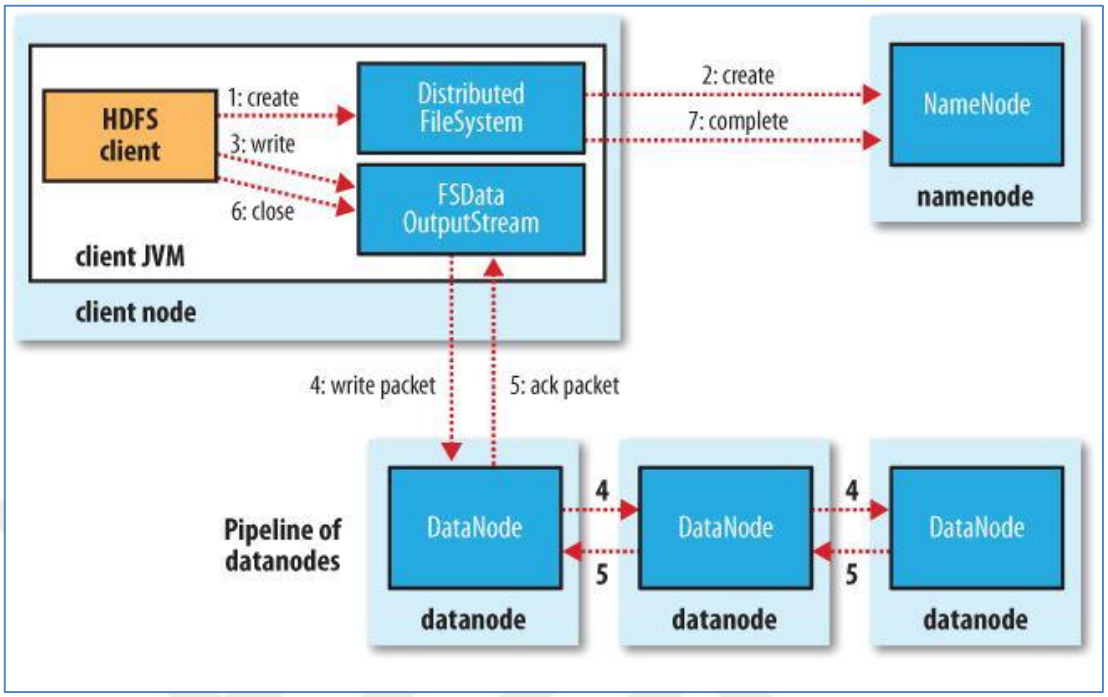
HDFS dosya sistemi iki farklı servisten meydana gelmektedir bunlar; dosya sistemi yönetici servisinde bulunan NameNode ve istemci servisinde bulunan DataNode

yazılımlarıdır. NameNode yazılımı, dosya sisteminde kayıtlı bulunan tüm dosya ve klasörlere ait dosya sistemi ağacını tutar ve yönetir. NameNode servisi dosyalardan kaç tane kopya olduğunu bunların hangi sistemler üzerinde bulunduğuna dair bilgileri üzerinde bulundurmaktadır ve bu bilgileri yönetmektedir. Her dosyaya ait kayıt bilgisi NameNode tarafından yerel diskte tutulur. NameNode ayrıca bir dosyaya ait blokların bazılarının tutulduğu DataNode'lara ait bilgileri de tutar. Yerel sunuculardaki blok adreslerini yerel olarak tutmaz sadece hangi bloğun hangi DataNode tarafından tutulduğuna dair bilgileri saklamaktadır.

DataNode servisi kendi üzerinde bulunan blokların saklanması, eklenme, değiştirilmesi, silinmesi ve adres yapısının oluşturulması gibi işlemlerin yapılmasından sorumludur. DataNode, düğümlerde bulunan disk blok işlemlerinin yürütüldüğü, yapılan işlemlerin durumu ve blok listelerini NameNode servisine gönderen servistir. NameNode servisi bütün dosya sistemini yönettiğinden bozulması durumunda bütün dosya sistemi ve dosyalar kaybolacaktır. Hadoop sisteminde bunun önüne geçebilmek için iki mekanizma geliştirilmiştir. Birinci yöntem; NameNode'un bütün dosya sistemi ağaç verisini belli aralıklarla imajını almasıdır, ikinci yöntem ise; paralelde çalışan ikinci bir NameNode servisi çalıştırmaktır. Bu ikinci (Secondary) NameNode bilgilerini yedekleme amaçlı çalışır ve genellikle ayrı bir makinede bulunması önerilmektedir.

4.2.6 HDFS girdi çıktı mekanizması (HDFS I/O)

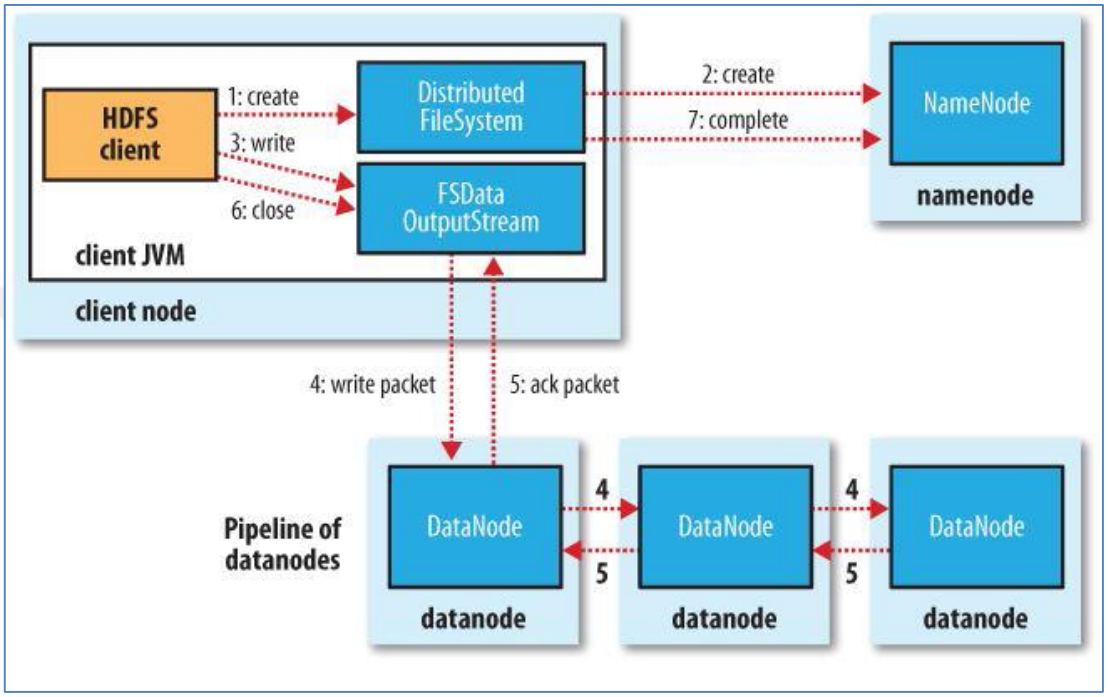
Kullanıcı bir dosyayı HDFS sistemi üzerinden erişmek istediği zaman bununla ilgili istekte bulunduktan sonra HDFS sisteminin bunu gerçekleştirme akışı şekil 12'deki gibidir.



Şekil 12.HDFS sisteminde dosya okuma istekleri için akış şeması [17].

Kullanıcı HDFS sisteminde istek yaptığında open() işlemi ile DistributedFileSystem nesnesinin NameNode ile haberleşmesini başlatır. Bunu gerçekleştirmek için Remote Procedure Call (RPC) [18] mekanizmasını kullanır. Böylelikle istenilen dosyaya ait blokların bulunduğu DataNode'lara ait adresleri NameNode geri döndürür. Bu sayede kullanıcının istediği bilgilerin hangi sistemler üzerinde bulunduğuna dair bilgiler alınmış olur. Bu gelen bilgiler DataNode'lardaki blokları ilgili adreslerinden okumak için FSDataInputStream üzerinden read() fonksiyonu çağrılır. Bu işlemi her blok bilgisini okumak için tekrarlayarak yapar. Tüm blok bilgileri okunduktan sonra FSDataInputStream üzerinde close() fonksiyonu kullanarak veri akışı işlemi durdurulur. Eğer veriyi DataNode'lardan okuma sırasında problem oluşursa okunacak blok bilgisini içeren diğer düğümler üzerinden bu blok tekrardan okunur. Eğer blok verisinin doğruluk bilgisi kontrol edilerek bozuk olduğu anlaşılır ise, bu durum NameNode'a bildirilir ve yedeklenmiş blok bilgisi (Replication) ile bozuk blok verisi güncellenir [19].

Kullanıcılar tarafından HDFS sistemine dosya yazma işlemleri akış şeması şekil 13' de ki gibidir. İşlemler sırası ile dosyanın oluşturulması, verinin dosyaya yazılması ve sonrasında dosyanın kapatılmasıdır.



Şekil 13.HDFS sisteminde dosya yazma istekleri için akış şeması [20].

Kullanıcı tarafından dosya DistributedFileSystem nesnesi üzerinden çağrılarak oluşturulur. DistributedFileSystem NameNode servisine bir RPC mesajı ile dosya sisteminde yeni bir dosya oluşturulması işlemini iletir [21]. Bu aşamada ilk oluşturulan dosya bilgisi blok verisi içermemektedir. NameNode servisi eğer aynı isimde dosya yoksa istemcinin yazma yetkilerini kontrol ederek bu isteği gerçekleştirmektedir. Bu parçalar daha sonra bloklar halinde sistemde bulunan DataNode servislerine kaydedilmektedir. Bu blokların çoğaltılarak kopyalanması işlemi bir boru hattı (pipeline) mekanizmasına göre olur. İlk DataNode bloğu kaydeder ve ikinci DataNode servisine gönderir ve servis isteği karşılayarak kaydetme işlemlerini yerine getirmektedir. Bu işlemden sonra diğer servislere kaydedilmesi için istekleri gönderilmektedir [22].

4.3 MapReduce Mimarisi

MapReduce özellikle dağıtık mimari sistemler üzerinde büyük verilerin hızlı bir şekilde işlenebilmesi ve analiz edilmesine imkân sağlayan bir yazılım sistemidir. 2004 yılında Google tarafından duyurulan bu sistem 1960'lı yıllarda geliştirilen fonksiyonel programlamadaki Map ve Reduce fonksiyonlarını temel alınması ile geliştirilmiştir. Büyük verilerin işlenmesi işlemlerinde bu iki fonksiyon kullanılmaktadır. Büyük boyuttaki bir veriyi işleyebilmek için çok yüksek donanıma sahip sunucular kullanmak yerine sıradan sunuculardan oluşan bir küme (cluster) sistemi üzerinde MapReduce yardımıyla aynı işlem çok daha hızlı bir şekilde gerçekleştirilebilir. Google'ın diğer arama motorları arasında fark yaratmasını sağlayan temel teknolojilerden birisi MapReduce teknolojisidir. Google haricinde sosyal medyanın çok fazla kullanılmaya başlandığı günümüzde Facebook, Linkedin, Twitter gibi çok fazla veri toplayan işlem hacmi hızlı ve çok yüksek olan şirketler de bu sistemi kullanmaktadır. Son 2 yılda web üzerinde üretilen verilerin geçen 10 seneden daha fazla olduğu düşünürsek ilerleyen zamanlarda bu sistemin çok daha fazla öne çıkacağı anlaşılmaktadır.

4.3.1 Map Fonksiyonu

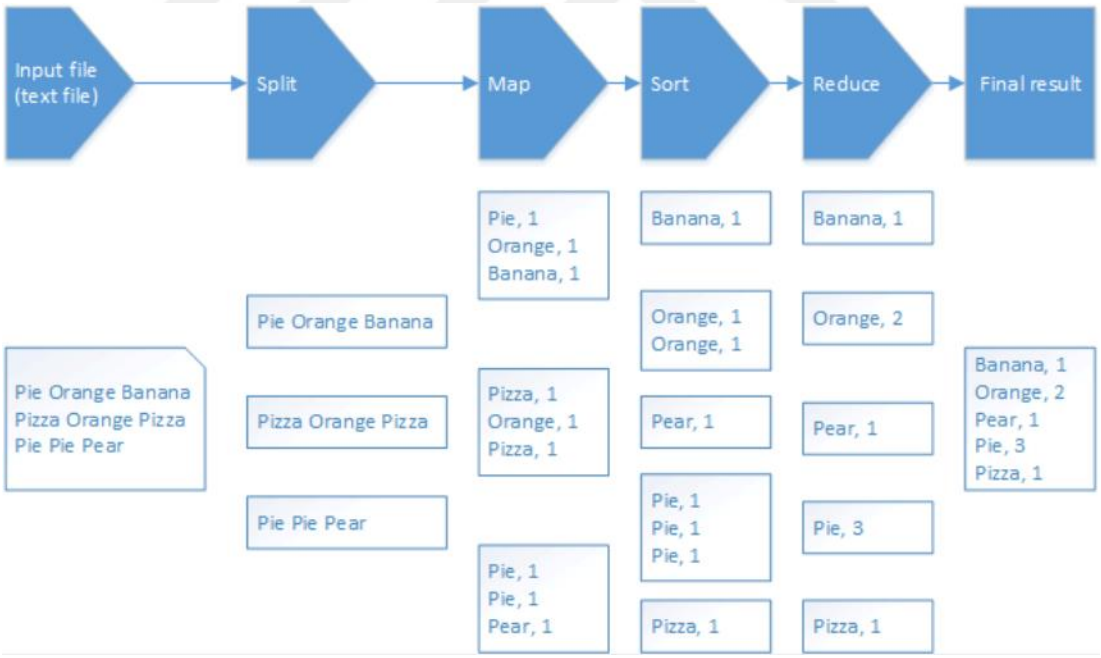
Map işlemi aşamasında ana (master) ve düğüm(node) servisleri işlenecek verileri alıp daha ufak parçalara ayırarak istemci sunuculara dağıtmaktadır. İstemci sunucular bu işleri tamamladıkça görevi dağıtan sonucunu master sunucuya geri göndermektedir. Map işlemini sistem tüm düğümlere dağıtabildiği için birbirinden bağımsız olarak paralel şekilde çalışabilmektedir bu sayede işlemler sistemin paralellik sayısına göre daha hızlı bir şekilde tamamlanabilmektedir.

4.3.2 Reduce Fonksiyonu

Reduce aşamasında ise tamamlanan görevler işin mantığına göre birleştirilerek ilgili sonuç elde edilmektedir. Kısaca Map aşamasında analiz edilen veri içerisinden almak istenilen veriler çekilir, Reduce aşamasında ise bu çekilen veri üzerinde istenilen analiz işlemleri gerçekleştirilir. Reduce aşamasında ise aynı anahtara sahip veriler paralel

olarak işlenebilir. Bu işlemi ilişkisel veri tabanında yapılan bir işlem ile karşılaştırırsak; Map aşaması select cümlesi ile ilgili alanların seçilmesi ve where ile ilgili filtreleme işlemlerini yapmaya, Reduce işlemi ise ilişkisel veri tabanında sorgulamalar yapılırken kullanılan count, sum, having, avg, min, max gibi ilgili veri üzerinde hesaplama yapma işlemine benzemektedir. Bu bilgileri görsel olarak ifade edersek şekil 14'deki gibidir. Şekil 14'de bu tez çalışmasının performans testlerinde kullanılan "Word Count" uygulamasının MapReduce iş akış şeması görünmektedir.

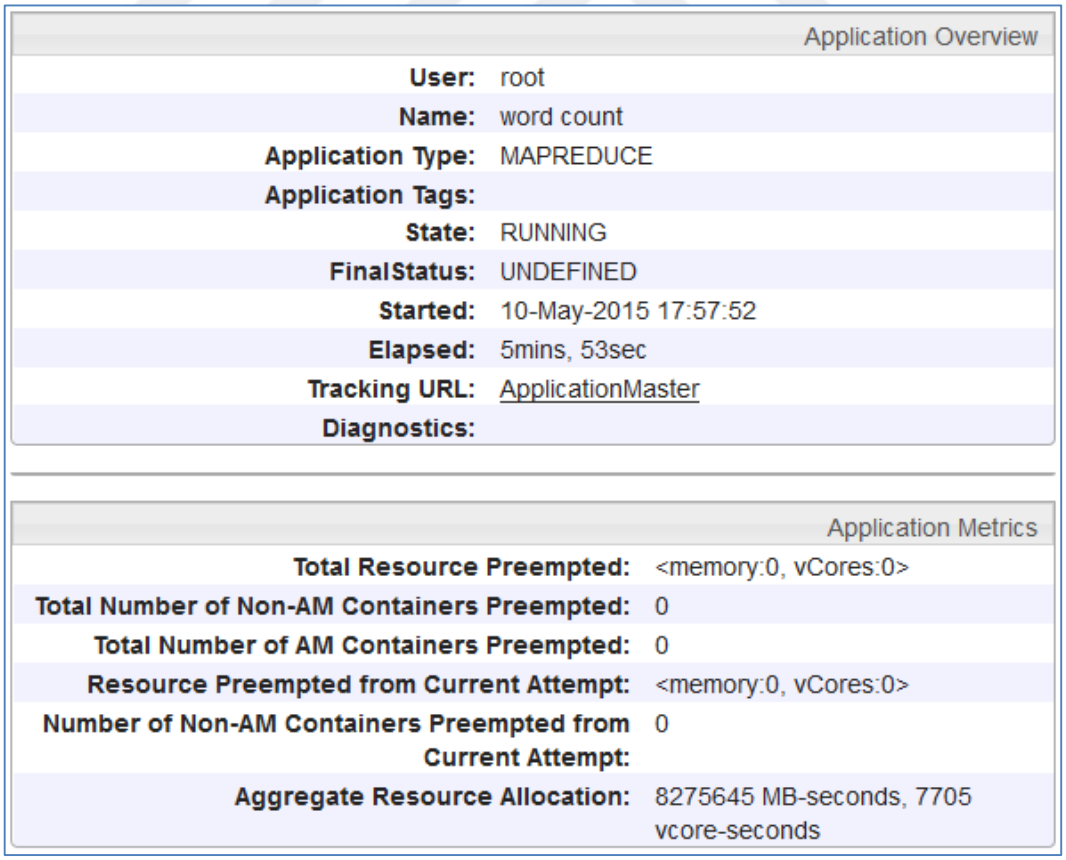
Şekil 14' de gösterildiği gibi başlatılan bir işe ait dosyaların nasıl bölündüğü, Map işleminin paralel olarak çalışması ve ayrı iş parçacıkları olarak kendi içlerinde işlemleri yapmaları sonrasında Reduce ile birleştirme işlemlerinin akışı görülmektedir. "Word Count" örneği sistemde bulunan metin dosyalarındaki kelimelerin sayılarını vermektedir.



Şekil 14.MapReduce "Word Count" uygulaması için akış şeması [23].

4.3.3 MapReduce Görevi Çalıştırma

Kullanıcı Hadoop üzerinde bir işi çalıştırmak istediğinde komut satırından hadoop komutunu ardından JAR parametresini kullanarak çalıştıracağı koda ait dosyanın adı ile birlikte giriş dosya bilgileri girilerek başlatılmaktadır. "hadoop jar /home/pi/hadoop/hadoop-2.6.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /TestDir /TestDir.Out" komut satırının çalıştırılması ile birlikte Hadoop'un bu işleme ait komut satırına yazdırdığı mesajlar görülebilir. Bu çıktıda zaman bilgileri ile birlikte görevin sonlanması sonucunda map ve reduce işleyicilerinin istatistikleri sergilenmektedir. Şekil 15'de yukarıdaki komut satırı ile başlatılan bir göreve ait istatistik ve durum bilgileri görünmektedir. Bu sayfaya web tarayıcısı üzerinden erişilmektedir. Standart hadoop kurulumunda JobTracker web gui erişimi 8088 numaralı portundan yapılmaktadır.



Application Overview

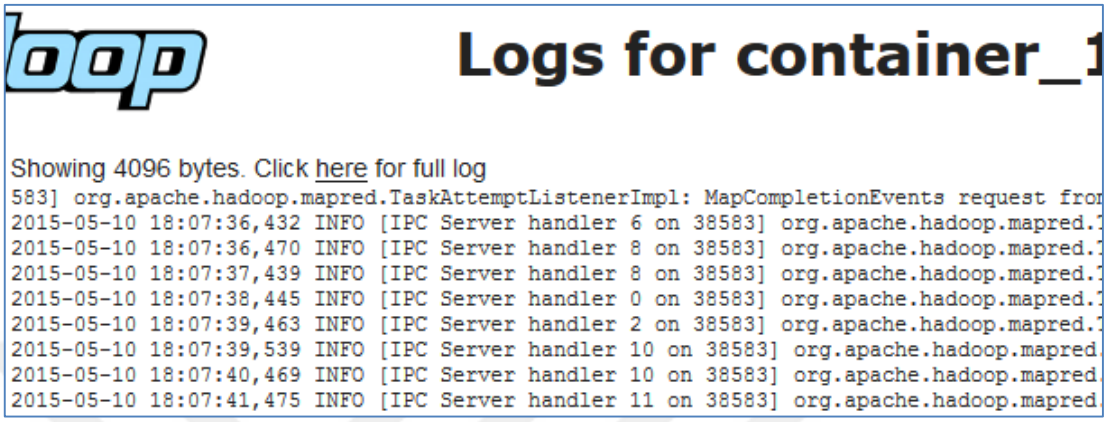
User:	root
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
State:	RUNNING
FinalStatus:	UNDEFINED
Started:	10-May-2015 17:57:52
Elapsed:	5mins, 53sec
Tracking URL:	ApplicationMaster
Diagnostics:	

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	8275645 MB-seconds, 7705 vcore-seconds

Şekil 15.Hadoop JobTracker uygulamasında çalışan bir göreve ait bilgiler.

Web uygulamasını kullanarak çalışan görevlere ait işlem loglarına bu loglar sayesinde işlemlerin nasıl tamamlandığını, hata alması veya uyarlara erişim sağlanabilmektedir.



Şekil 16.Hadoop JobTracker uygulamasında çalışan göreve ait loglar.

Çalışan görev devam ederken terminal ekranına bilgi mesajları düşmektedir. Bu mesajlarda işin tamamlanma yüzdesi gibi bazı bilgiler yazdırılmaktadır. Aşağıdaki örnek log ise mapraduce işlemi tamamlandıktan sonra sonuç raporu olarak terminal ekranına yazdırılmaktadır. Görevin bitimi sonrası gelen bu raporda işlemin ne kadar sürdüğü işlem sırasında yapılan okuma, yazma boyutu paralel map işlem sayısı, reduce sayısı, bu işlemler sırasında hata alınmış ise bunların sayıları gibi birçok veri raporda yer almaktadır.

```
File System Counters

FILE: Number of bytes read=14458256
FILE: Number of bytes written=34294282
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=61327091
HDFS: Number of bytes written=240900
HDFS: Number of read operations=153
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters

Failed map tasks=8
```

```
Launched map tasks=58
Launched reduce tasks=1
Other local map tasks=8
Data-local map tasks=50
Total time spent by all maps in occupied slots
(ms)=9766835
Total time spent by all reduces in occupied slots
(ms)=679859
Total time spent by all map tasks (ms)=9766835
Total time spent by all reduce tasks (ms)=679859
Total vcore-seconds taken by all map tasks=9766835
Total vcore-seconds taken by all reduce tasks=679859
Total megabyte-seconds taken by all map
tasks=10001239040
Total megabyte-seconds taken by all reduce
tasks=696175616
Map-Reduce Framework
Map input records=1049850
Map output records=10636300
Map output bytes=102732700
Map output materialized bytes=14458550
Input split bytes=5191
Combine input records=10636300
Combine output records=979550
Reduce input groups=19591
Reduce shuffle bytes=14458550
Reduce input records=979550
Reduce output records=19591
Spilled Records=1959100
Shuffled Maps =50
Failed Shuffles=8
Merged Map outputs=50
GC time elapsed (ms)=404952
CPU time spent (ms)=771500
Physical memory (bytes) snapshot=7591542784
Virtual memory (bytes) snapshot=48137740288
Total committed heap usage (bytes)=6106701824
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=2
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=61321900
File Output Format Counters
Bytes Written=240900
```

Bu rapora web tarayıcısı üzerinden de erişim mümkündür ayrıca son Hadoop versiyonunda JobHistory servisi sisteme eklenmiştir. JobHistory servisi sayesinde geçmiş görevlere ait loglara erişim ve istatiksel bilgilere erişim sağlanabilmektedir.

4.4 Apache Pig

Apache Hadoop üzerinde prosedürler şeklinde çalışan bir data akışı yazmayı sağlayan veri işleme platformudur. Veri işlemleri ile ilgilenen kullanıcılara Hadoop'un güçlü, dağıtık ve esnek yapısına ileri seviye java kodları yazmadan daha üst bir katmandan erişim imkânı sağlayan bir yapıdır. Pig Latin olarak adlandırılan basit bir dile sahip açık kaynak kodlu projedir. Pig'in iki önemli bileşeninden Pig Latin dilini kullanarak kullanıcılar veri akışı scriptleri yazabilmektedir. Pig bunları Hadoop üzerinde MapReduce kodlarına çevirerek çalıştırmaktadır. Pig'in en büyük getirisi Java bilmeyen kullanıcılara da Hadoop üzerinde üst seviye prosedürler şeklinde çalışan bir script dili ile veriyi işleme imkânı vermesi ve Map Reduce programları yazmaya gerek kalmadan işlemleri yapabilmeye imkân sağlayan bir yazılım olmasıdır [24].

Yazılımcı tarafından yazılan Pig Latin kodları otomatik olarak MapReduce işlerine dönüştürülmektedir ve Hadoop üzerinde çalıştırılmaktadır. Pig kodu yazılımcısı yazdığı Pig Latin scriptinin arka tarafta sistem üzerinde kaç adet MapReduce koduna çevrildiğini, kaç kez derleneceğini, paketlenip çalıştırılarak sonuçların getirildiğini fark etmemektedir. Bu işlemlere ait bilgilere işlem loglarına kontrol edilerek ilgili incelemeler ve hata ayıklanması yapılabilmektedir.

Pig kodlarını çalıştırmak için 3 farklı yöntem kullanılmaktadır. Script ile kullanımı, Piglatin komutlarından oluşan pig script'leri oluşturmak ve bunları bir ara yüz veya komut satırından çalıştırmak örnek olarak "`$ pig -F pig_ornek.pig`" şeklindedir. Shell yani komut satırından kullanım, Grunt, Pig'in interaktif shell ortamıdır bu arayüz ile komutlar direk yazılarak kullanılabilir. Embedded olarak kullanımı**,** Pig programları Java kodlarının içinden de kullanılabilir. Pig ara yüzüne bağlanarak şekil 17'deki gibi örnek bir kod çalıştırıldığında, "pig_deneme.csv" isimli dosyadaki "SiraNo" alanında 3' den büyük olan kayıtlar ekrana gelmektedir.

```
grunt> A = LOAD 'pig_deneme.csv' USING PigStorage() AS (SiraNo:int, AdSoyad:chararray);
grunt> B = FILTER A BY SiraNo > 3;
grunt> C = FOREACH B GENERATE *;
grunt> DUMP C;
```

Şekil 17.Apache Pig örnek sorgulama kodu.

Şekil 17' deki Pig kodu çalıştırıldıktan sonra şekil 18'daki gibi terminal ekranına sonuçlar gelmektedir. Terminal üzerinden Pig ara yüzüne bağlantı yapılarak çalıştırılır ise sonuçlar terminal ekranına yazılacaktır. Terminal ekranına yazdırılan bu ilgileri ayrıca bir dosyaya aktarılabilir. Terminal ekranı haricinde Pig yazılımına java veya farklı yazılım platformlarından da erişim sağlanabilmektedir.

```
Job DAG:
job_201209132133_0001

2012-09-13 21:54:09,008 [main] INFO  org.apache.pig.backend.hadoop.exe
2012-09-13 21:54:09,021 [main] INFO  org.apache.hadoop.mapreduce.lib.i
2012-09-13 21:54:09,021 [main] INFO  org.apache.pig.backend.hadoop.exe
1
(1,Ahmet Y.)
(2,Mehmet T.)
(3,Selen F.)
(4,Elif Y.)
(5,Deniz Y.)
grunt>
```

Şekil 18.Apache Pig örnek sorgulama sonucu.

Sonuçlara ait bilgiler HDFS dosya sisteminde çalışan iş parçacığına ait bir dizin altında da saklanmaktadır bu işlemi hadoop ve mapreduce yazılımları otomatik olarak yapmaktadır.

4.5 Apache Hive

Apache Hive yazılımı SQL benzeri bir ara yüz yardımıyla Hadoop üzerinde Java kullanmadan sorgulama ve analiz işlemlerini yapmak amacıyla Facebook tarafından geliştirilmiş, daha sonrasında Apache' ye devredilmiş açık kaynaklı kodlu bir projedir [25]. Özellikle Veri Ambarı uygulamalarını Hadoop kümeleri üzerinde geliştirebilmek için Hive projesi birçok firma tarafından kullanılmaktadır. Hive kullanarak Hadoop sistemi üzerinde belirli bir formata uygun şekilde bulunan dosyaları, örnek olarak

“.csv” dosyalarını tanımlayıcı bilgileri ile birlikte işleyerek tablo gibi kullanılabilir. Hive bu bilgileri saklayarak çalıştırılan SQL benzeri sorguları MapReduce kodlarına çevirerek bu dosyalar üzerinde select, join işlemlerini gerçekleştirmektedir. Bu sayede çok büyük miktarda veriyi SQL gibi Hadoop üzerinde paralel olarak sorgulama olanağı bulunmaktadır [26].

Hive konsol ara yüzünden kullanılabildiği gibi “-e” parametresi ile script şeklinde hazırlanmış dosyalar ile de kullanılabilmektedir. Hive kodlaması SQL’ e çok benzer cümleler ile yapılmaktadır. Bu sayede daha kolay bir kullanım sunmaktadır. Hive üzerinde örnek bir tablo oluşturma işlemine ait kodlama şekil 19’daki gibi olmaktadır.

```
hive> CREATE EXTERNAL TABLE hive_deneme
          (SiraNo INT, AdSoyad STRING)
          ROW FORMAT DELIMITED FIELDS
          TERMINATED BY '\t'
          LINES TERMINATED BY '\n'
          STORED AS TEXTFILE
          LOCATION '/hadoop/hadoop-data/hive-example';
OK
Time taken: 0.232 seconds
```

Şekil 19.Apache Hive örnek tablo oluşturma kodu.

Bu kod bloğu ile bir tablo oluşturma isteği yapılmaktadır. Bu tablonun int ve String tipinde iki kolondan oluştuğu, dosyanın tab karakterleri ile ayrıldığı, satırların \n karakteri ile bittiğini ve Hadoop üzerinde “/hadoop/hadoop-data/hive-example” klasörü üzerinde bulunduğu belirtilmektedir. Hive üzerinde oluşturulan tablolar şekil 20’ de gösterilen SQL yazım dili gibi sorgulanabilmektedir.

```
hive> select * from hive_deneme;
OK
1   Ahmet Y.
2   Mehmet T.
3   Selen F.
4   Elif Y.
5   Deniz Y.
Time taken: 0.314 seconds
```

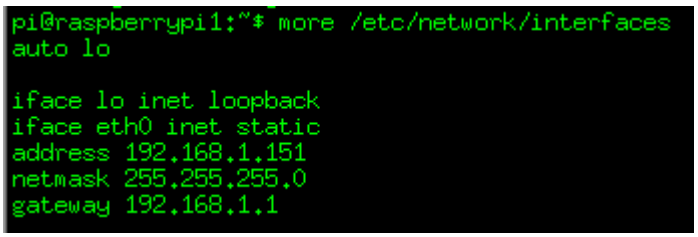
Şekil 20.Apache Hive ile tablo sorgulama işlemi.

4.6 Raspberry Pi Üzerine Apache Hadoop ve Bileşenlerinin Kurulumu

Raspberry Pi üzerine Hadoop kurulumuna başlamadan önce cihazlar üzerindeki bazı sistem ayarlarının yapılması gerekmektedir. Bu tez çalışmasında sistem üzerindeki kullanıcı root olarak seçilmiştir. İstenilirse Hadoop sistemi için ayrı bir kullanıcı tanımı yapılabilir bunun için önce işletim sistemi üzerinde bir kullanıcı tanımlayıp sonra Hadoop' un istediği yetkilerin verilmesi gerekmektedir. Eğer kapalı devre bir sistem kuruluyor ise (örnek olarak bu tez çalışması gösterilebilir) yetki sorunlarını aşmak için root user kullanımı tercih edilebilir bu sayede işletim sistemi tarafında bazı klasörleri tanımlamaya ve bu klasörlere atanacak kullanıcı yetkilerini belirlemeye gerek kalmamaktadır.

4.6.1 Tek Düğüm (Single Node)

Tek sunucu üzerinde Hadoop' un tüm servislerini çalıştırabilir. Tek sunucu olmasına rağmen sunucudaki işlemcinin çekirdek (CPU Core) sayına bağlı olarak kendi içerisinde yine paralel olarak sistem çalışacaktır. Tek sunucu kurulumunda ilk olarak sistemin ağ ayarlarının yapılması gerekmektedir. Bu tez çalışmasının **"3.6.4 Ağ Ayarları"** bölümünde açıklandığı şekilde sistemin ağ ayarlarının Şekil 21'de ki gibi yapılması gerekmektedir.



Şekil 21.Ağ adresi ayarları.

Sistemin isim (hostname) bilgisi "raspberrypi1" olarak düzenlenmiş ve tanımlama "/etc/hostname" dosyası içerisinde yapılmaktadır. Raspberry Pi üzerine kurulmakta olan Raspbian işletim sistemi kendi içerisinde java paketleri yüklü olarak gelmektedir. Bu tez çalışmasında işletişim sistemi ile gelen Java sürümü yerine Oracle' ın ARM

işlemciler için yayınladığı ve daha güncel olan sürüm tercih edilmektedir. Oracle firmasının yayınladığı ARM tabanlı işlemciler için Java sürümü "http://www.oracle.com/technetwork/java/javase/downloads/jdk8-arm-downloads-2187472.html" adresinden indirilmektedir.

Oracle' ın ARM işlemciler için yayınladığı Java sürümü derlenmiş olduğu için ayrıca derleme işlemlerinin yapılmasına gerek yoktur. Java indirildikten sonra sıkıştırılmış dosya istenilen bir dizin altında açılır. İşletim sisteminde Java sürümü 1.8.0 şeklindedir Oracle üzerinden indirilen sürüm ise 1.8.3_33 şeklindedir. Java 1.8.0_33 sürümünün tercih edilmesi sayesinde bazı hataların alınmaması sağlanmaktadır.

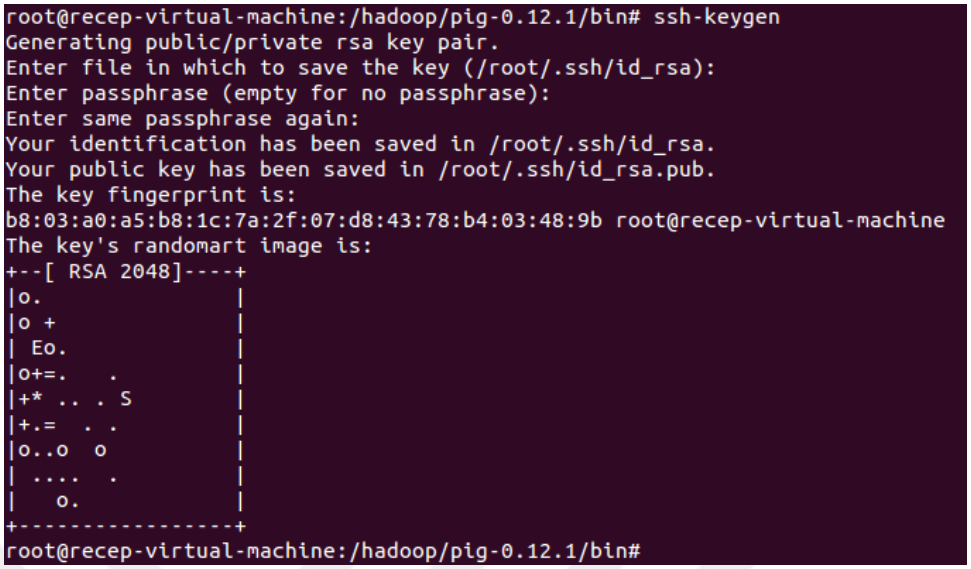
Bu tez çalışmasında kullanılan Hadoop sürümü 2.6.0 şeklindedir. Bu sürüm Java 1.8 sürümü ile istikrarlı çalışmaktadır.

Eğer daha düşük bir Hadoop sürümü kullanılacak ise o zaman sürüme uygun bir Java ile çalışılması gerekmektedir. Örneğin Hadoop 1.x.x versiyonu Java 1.6 ile çalışmaktadır bu durumda Raspberry Pi sistemi ile gelen Java 1.8 sürümü kullanılmamaktadır yukarıda bahsedildiği şekilde işletim sistemi ile gelen Java yerine ilgili sürümün yüklenmesi gerekmektedir.

Hadoop sistemi çalışırken kendi içerisinde birçok alt servis ve bunların kullandığı portları sistem üzerinde açmaktadır bu işlemi yaparken sunucu üzerine SSH bağlantısı kurması gerekmektedir. SSH bağlantısının şifre sorulmadan yapılabilmesi gerekmektedir. SSH ile şifresiz bağlantı işlemi servislerin otomatik olarak başlatılması aşamasında kullanılmaktadır. SSH servisinin nasıl aktif edileceği bu yazının giriş kısmında açıklanmaktadır.

SSH üzerinden şifresiz bağlantı için aşağıdaki adımların takip edilmesi gerekmektedir.

- SSH-RSA Key dosyasının hazırlanması; "ssh-keygen" komutu ile RSA key dosyasını oluşturulmalıdır. Dosya "/root/.ssh/id_rsa" şeklinde oluşmaktadır.



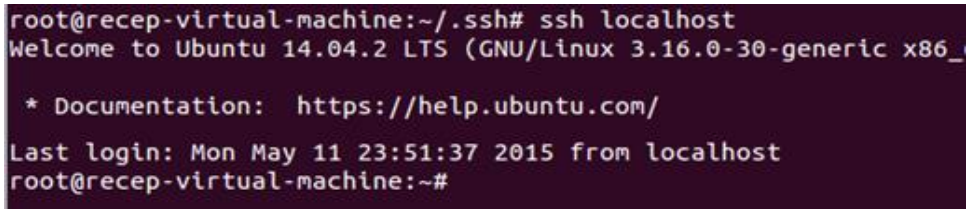
Şekil 22.SSH-RSA Key dosyasının oluşturulması.

- SSH Key dosyasına otomatik bağlantı yapılacak user bilgisinin eklenmesi; "ssh-copy-id -i root@localhost" komutunu kullanarak bu işlem gerçekleştirilmektedir.

- Root user kullanılacak ise "/etc/ssh/sshd_config" dosyasında "Authentication" bölümüne aşağıdaki satırların eklenmesi gerekmektedir.

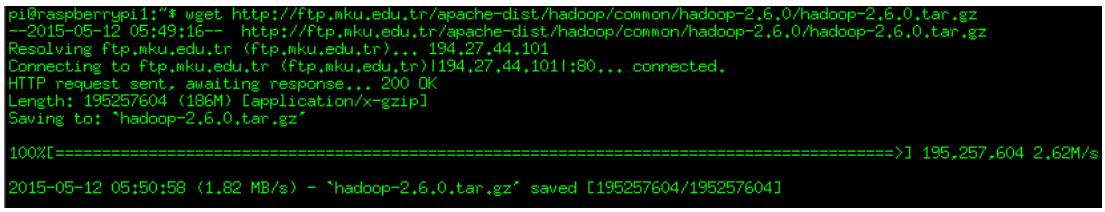
  #PermitRootLogin without-password
  PermitRootLogin yes

- Yapılan ayarların test edilmesi için "ssh localhost" komutu çalıştırılmalıdır. Bu komut çalıştığında şifre sorulmadan yeni SSH bağlantısı açılmaktadır.



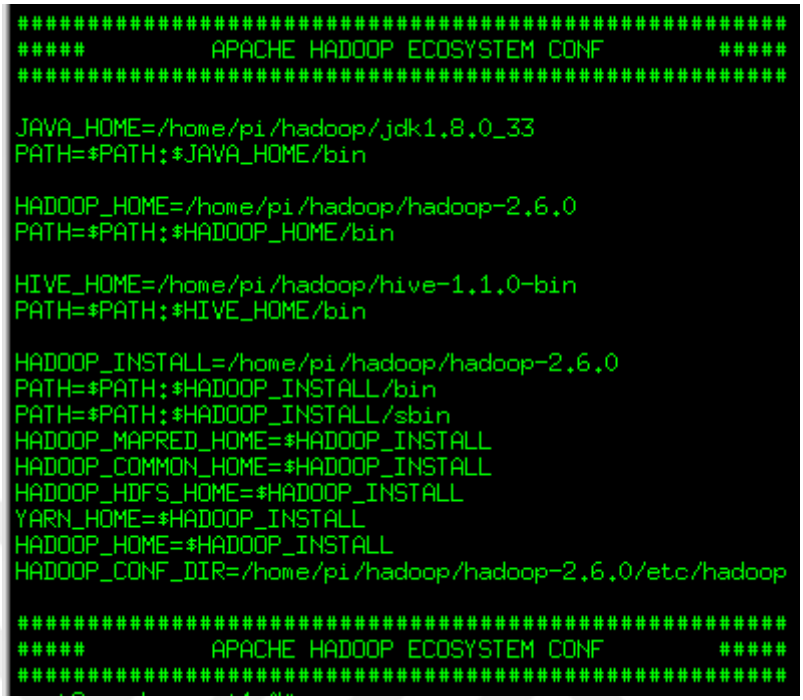
Şekil 23."ssh localhost" komutunu kullanarak şifresiz bağlantı işlemi.

Hadoop kurulum işlemleri için kullanılacak Hadoop sürümünün sistem üzerine indirmesi gerekmektedir. Bu tez çalışmasında Hadoop 2.6.0 sürümü kullanılmaktadır. Hadoop sürümünü "wget http://ftp.mku.edu.tr/apache-dist/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz" komutlarını kullanarak sistem üzerine indirilebilir.



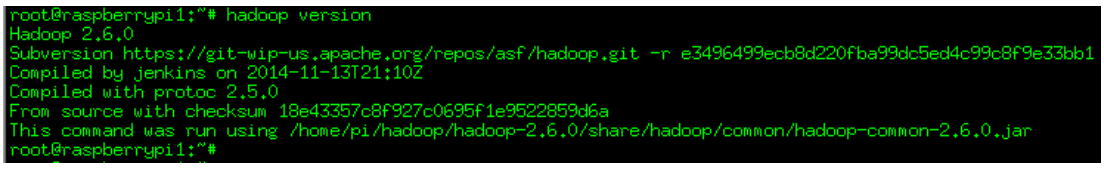
Şekil 24.Hadoop sürümünün sistem üzerine indirilmesi.

İndirilen sıkıştırılmış dosyayı "tar -xvzf hadoop-2.6.0.tar.gz -C / /home/pi/hadoop/hadoop-2.6.0" komutunu kullanarak Hadoop yazılımını hangi dizine kurulumu yapılacak ise ilgili dizin altına açılmalıdır. Hadoop' un Java yazılımını kullanabilmesi için sistemde bazı tanımların (Environment Variables) yapılması gerekmektedir. Şekil 25'deki tanımların kalıcı olarak yapılması gerekmektedir. Sistem yeniden başladığında otomatik olarak bu ayarları kullanabilmesi için bu tanımları root user home dizini altında bulunan ".bashrc" ve ".profile" dosyaları içerisine eklenmelidir. Bu tanımlar içerisinde Java' nın bulunduğu dizin, Hadoop ve bileşenlerinin bulunduğu dizin ve ilerleyen sayfalarda bahsedilecek olan Pig ve Hive kurulum dizinlerinin bilgileri yer almaktadır. Bu tanımlar aktif olduğunda sistem üzerinde hangi dizinde olunursa olunsun Hadoop komutlarını terminalde direk yazarak kullanabilir. Eğer sistem üzerindeki Java sürümü kullanılacak ise JAVA_HOME değişkeninin "export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")" şeklinde tanımlanması gerekmektedir.



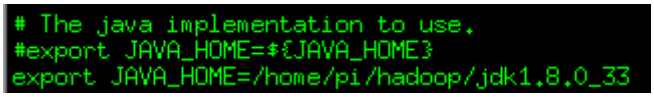
Şekil 25.Hadoop için gerekli olan “Environment Variables” tanımları.

Tanımlar yapıldıktan sonra tanımların test edilmesi için “hadoop version” komutu kullanılarak Hadoop’ un aktif olma durumu kontrol edilebilir.



Şekil 26.“hadoop version” sürüm bilgisinin alınması.

Hadoop sistem ayarlarının bulunduğu dosya “hadoop-env.sh” şeklindedir. Bu dosya içerisinde Java ayarlarının şekil 27’deki gibi yapılması gerekmektedir.



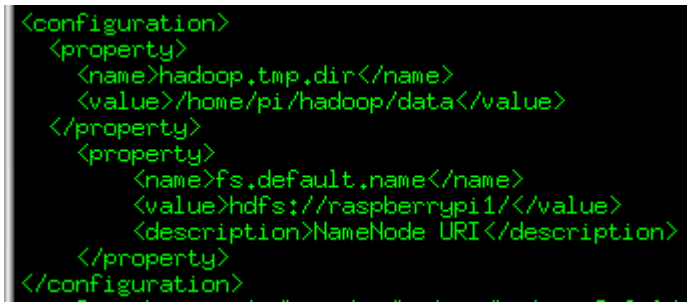
Şekil 27.Hadoop Java ayaları.

Hadoop konfigürasyon dosyası olan “hadoop-env.sh” içerisindeki “HADOOP_CONF_DIR” ayarının “export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/home/pi/hadoop/hadoop-2.6.0/etc/hadoop"}” şeklinde yapılması gerekmektedir. Kullanılan sistemin Raspberry Pi 2 cihazlarında oluşması sebebi ile cihazdaki memory kapasitesi 1 GB şeklindedir. Memory ayarının bu kapasiteye uygun olarak “hadoop-env.sh” içerisinde “export HADOOP_HEAPSIZE=250” şeklinde tanımlanması gerekmektedir.

Buraya kadar yapılan ayarlar Hadoop sistem ayarları ile ilgiliydi, aşağıda devam eden ayarlar ise Hadoop yazılımına ait ayarlardır. Bu ayarlar çalışılacak dizin, portların seçilmesi ve geçici (temp) olarak kullanılacak dizinlerin belirlenmesi gibi ayarları içermektedir.

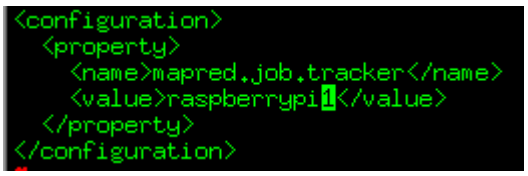
Bu ayarlar için Hadoop yazılımının 3 ana yapılandırma dosyası bulunmaktadır. Bunlar Hadoop’ un çalışacağı dizin, port ve sisteme ait bilgilerin yer aldığı “core-site.xml” dosyası, MapReduce işlemlerine ait ayarların bulunduğu “mapred-site.xml” dosyası ve dosya sistemine (HDFS File System) ait ayarların bulunduğu “hdfs-site.xml” dosyasıdır.

“core-site.xml” dosyasının içeriği şekil 28’ deki gibi yapılması gerekmektedir.



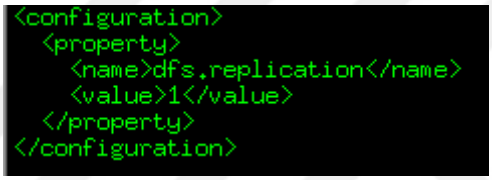
Şekil 28.“core-site.xml” dosyasının içeriği.

“mapred-site.xml” dosyasının içeriği şekil 29’ da ki gibi yapılması gerekmektedir.



Şekil 29.“mapred-site.xml” dosyasının içeriği.

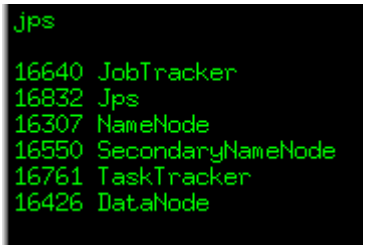
“hdfs-site.xml” dosyasının içeriği şekil 30’ da ki gibi yapılması gerekmektedir.



Şekil 30.“hdfs-site.xml” dosyasının içeriği.

“hdfs-site.xml” dosyasında bulunan “dfs.replication” değeri “1” olarak ayarlanmıştır. Bu ayar Hadoop sisteminde bir dosyanın kaç kopyasının olmasını belirler bu başlık altında tek sunucu (Single Node) üzerinde kurulum anlatıldığı için kopya 1 şeklinde yapılmıştır. Sistemdeki makinelerin sayısına bağlı olarak kopya sayısı değiştirilmektedir. Örneğin 10 adet makine var bunlardan 5 tanesine kopya alınması sağlanabilir.

Hadoop sistemi ile ilgili ayarlar tamamlandıktan sonra HDFS dosya sisteminin oluşturulması için formatlanması gerekmektedir bunun için “hadoop namenode -format” komutunu kullanılır. Format komutu ile HDFS dosya sistemi oluşturulmaktadır. Bu ayarlardan sonra Hadoop çalışır duruma gelecektir. Hadoop servislerini çalıştırmak için “start-all.sh” komutunu kullanılmaktadır ve servisler açıldıktan sonra “jps” komutu ile çalışan servislerin listesi alınabilmektedir. Servisleri durdurmak için “stop-all.sh” komutu kullanılmaktadır. Hadoop’ un düzgün çalışması olması için 5 adet servisin çalışmış olması gerekmektedir. “jps” komutunun çıktısı şekil 31’ deki gibi olması gerekmektedir.



Şekil 31.“jps” komutu ile çalışan Hadoop servislerinin görüntülenmesi.

Bu aşamaya gelindiğinde Hadoop sistemi çalışır duruma gelmiş olmaktadır sistemi test etmek için “hadoop dfs -put /tmp/test.txt /” komutunu kullanarak “/tmp/test.txt” dosyasını Hadoop root dizine kopyalanabilir eğer hata alınmadı ise HDFS dosya sistemine dosya kopyalama işlemi gerçekleştirilmiştir. “hadoop dfs -ls /” komutu ile HDFS dosya sisteminde root dizininin listesi alınabilmektedir. “hadoop jar /hadoop/hadoop-1.2.1/hadoop-examples-1.2.1.jar wordcount /test.txt /test.out” komut satırını kullanarak “test.txt” dosyasında bulunan kelime sayısı öğrenilebilir ve bu komutlar kullanılarak sistem test edilebilir.

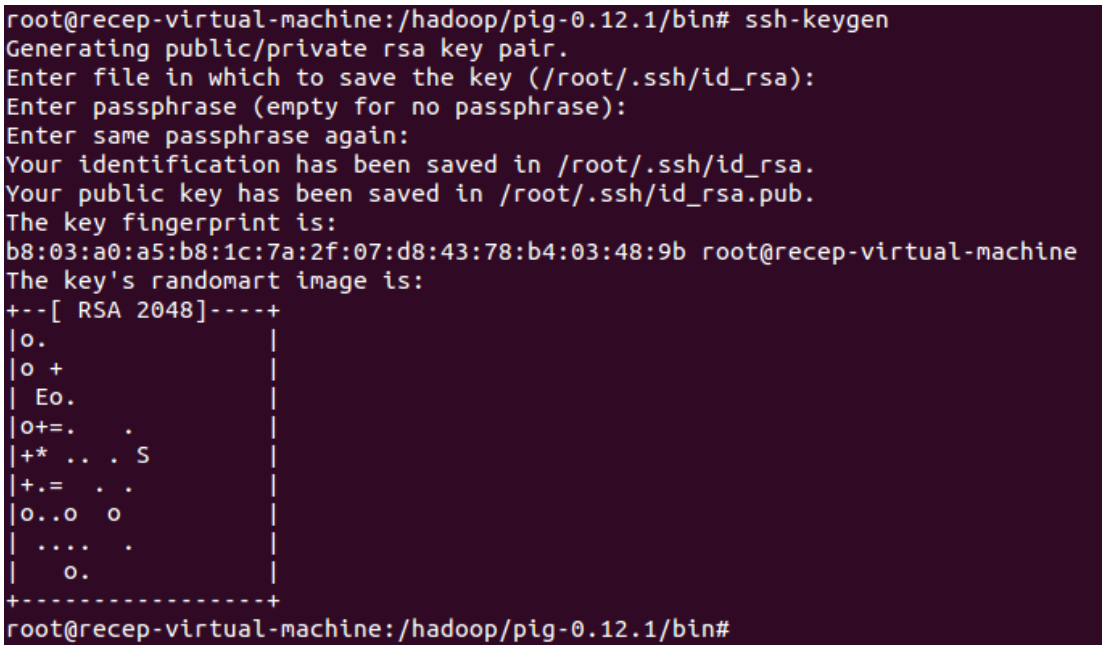
4.6.2 Küme Sistemi (Cluster Node)

Çoklu makineye (Multi (Cluster) Node) kurulum işlemi için tek sunucu (Single Node) kurulumundaki işlemlerin aynı şekilde yapılması gerekmektedir. Bu tez çalışmasında tek sunucuya kurulum yapıldıktan sonra diğer 4 sunucuya kurulum için önce tüm sunuculara aynı dizinler ve ayarlar kopyalanmıştır. Hadoop cluster kurulumda bazı ayarlarda değişiklikler yapılmaktadır bu değişikliklerin aşağıdaki gibi sırası ile yapılması gerekmektedir. Cluster sistemine eklenecek tüm sunucular sunucu isimleri ile birbirini çözmesi gerekmektedir. Bu işlem iki şekilde yapılabilir; birincisi ağ içerisine bir DNS sunucusu kurulması ve bu DNS sunucusu üzerine tüm sunuculara ait kayıtların girilmesi ayrıca sunucular üzerinde de DNS IP ayarının DNS sunucusu olarak düzenlenmesi, ikincisi ise DNS sunucusu kurulumuna gerek kalmadan sunuculardaki “/etc/hosts” dosyasına tüm sunucuların kayıtlarının girilmesi. Şekil 32’ da tüm sunuculara ait girilmiş olan kayıtlar görünmektedir bu düzenleme tüm sunucularda aynı şekilde yapılmalıdır.

Şekil 32.“/etc/hosts” dosyasına tüm sunuculara ait kayıtların girilmesi.

SSH üzerinden şifresiz bağlantı için aşağıdaki adımların takip edilmesi gerekmektedir.

- SSH-RSA Key dosyasının hazırlanması; “ssh-keygen” komutu ile RSA key dosyası oluşturulur. Dosya “/root/.ssh/id_rsa” şeklinde oluşacaktır. Bu işlem her düğüm üzerinde yapılmalıdır ve bütün düğümler birbirlerine şifresiz olarak bağlanabilmesi gerekmektedir.



Şekil 33.SSH-RSA Key dosyasının oluşturulması.

- SSH Key dosyasına otomatik bağlantı yapılacak user bilgisinin eklenmesi; Aşağıdaki komut satırları her sunucuda çalıştırılmalıdır.

  “ssh-copy-id -i root@raspberrypi1”
  “ssh-copy-id -i root@raspberrypi2”
  “ssh-copy-id -i root@raspberrypi3”
  “ssh-copy-id -i root@raspberrypi4”
  “ssh-copy-id -i root@raspberrypi5”

- Root user kullanılacak ise “/etc/ssh/sshd_config” dosyasında “Authentication” bölümüne aşağıdaki satırların eklenmesi gerekmektedir.

  #PermitRootLogin without-password
  PermitRootLogin yes

- Yapılan ayarların test edilmesi için aşağıdaki satırların her sunucuda çalıştırılması ge şifresiz olarak girişlerin yapılması gerekmektedir.

  “ssh root@raspberrypi1”
  “ssh root@raspberrypi2”
  “ssh root@raspberrypi3”
  “ssh root@raspberrypi4”
  “ssh root@raspberrypi5”

Cluster kurulumunda bir sunucun ana (master) diğerlerinin ise yardımcı (slave) olmaları gerekmektedir. Bu sebeple cluster kurulumda Hadoop konfigürasyon dosyalarının bulunduğu dizin altına “masters” isminde bir dosya oluşturup içerisine master sunucunun hostname bilgisinin eklenmesi slaves isminde bir dosya oluşturup içerisine slave olacak sunucuların eklenmesi gerekmektedir.

Master sunucu aynı zamanda slave sunucu işlevini de görebilir eğer bu şekilde kullanılacak ise master sunucu hostname bilgisi hem masters hem de slaves dosyasına

eklenmesi gerekmektedir. Bu tez çalışmasında beş Raspberry Pi cihazının bir tanesi master dört tanesi ise slave olarak kullanılmıştır. Hadoop konfigürasyon dosyalarının düzenlenmesi ise aşağıdaki gibi olmalıdır.

“core-site.xml” dosyasının içeriği şekil 34’deki gibi yapılması gerekmektedir.

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/pi/hadoop/data</value>
  </property>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://raspberrypi1/</value>
        <description>NameNode URI</description>
    </property>
</configuration>
```

Şekil 34.“core-site.xml” dosyasının içeriği.

“hdfs-site.xml” dosyasının içeriği şekil 35’ deki gibi yapılması gerekmektedir.

```
<configuration>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///home/pi/hadoop/data/namenode</value>
        <description>NameNode directory for namespace and transaction logs storage.</description>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>4</value>
    </property>
    <property>
        <name>dfs.permissions</name>
        <value>false</value>
    </property>
    <property>
        <name>dfs.datanode.use.datanode.hostname</name>
        <value>false</value>
    </property>
    <property>
        <name>dfs.namenode.datanode.registration.ip-hostname-check</name>
        <value>false</value>
    </property>
    <property>
         <name>dfs.namenode.http-address</name>
         <value>raspberrypi1:50070</value>
         <description>Your NameNode hostname for http access.</description>
    </property>
    <property>
         <name>dfs.namenode.secondary.http-address</name>
         <value>raspberrypi1:50090</value>
         <description>Your Secondary NameNode hostname for http access.</description>
    </property>
</configuration>
```

Şekil 35.“hdfs-site.xml” dosyasının içeriği.

“hdfs-site.xml” dosyasında bulunan “dfs.replication” değeri 4 olarak ayarlanmıştır. Bu ayar Hadoop sisteminde bir dosyanın kaç kopyasının olmasını belirler bu başlık altında cluster sistem kurulumu anlatıldığı için kopya 4 şeklinde yapılmıştır. 1 adet master 4 adet slave sunucusu bulunmaktadır. 4 adet kopya yapılacağı için HDFS dosya sisteminde atılan her dosya her sunucuda birer kopya olarak saklanacaktır.

“mapred-site.xml” dosyasının içeriği şekil 36’ da ki gibi yapılması gerekmektedir.

```
<configuration>
<property>
  <name>mapred.child.java.opts</name>
  <value>-Xmx768m</value>
</property>
<property>
  <name>mapreduce.job.jvm.numtasks</name>
  <value>10</value>
</property>
<property>
  <name>mapred.reduce.parallel.copies</name>
  <value>20</value>
</property>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
<property>
  <name>mapred.job.tracker</name>
  <value>raspberrypi1</value>
</property>
<property>
  <name>mapreduce.map.speculative</name>
  <value>false</value>
  <final>true</final>
</property>
<property>
  <name>mapreduce.reduce.speculative</name>
  <value>false</value>
  <final>true</final>
</property>
<property>
  <name>mapreduce.tasktracker.map.tasks.maximum</name>
  <value>20</value>
</property>
<property>
  <name>mapreduce.tasktracker.reduce.tasks.maximum</name>
  <value>20</value>
</property>
<property>
  <name>mapreduce.jobtracker.restart.recover</name>
  <value>false</value>
</property>
</configuration>
```

Şekil 36.“mapred-site.xml” dosyasının içeriği.

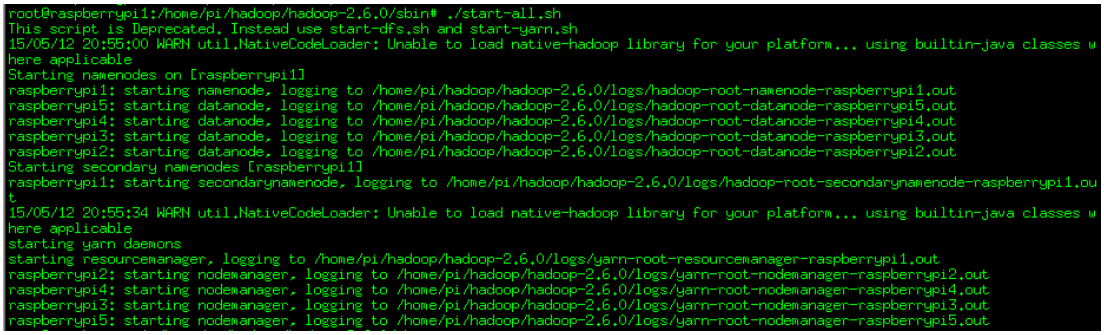
Cluster kurulumda yukarıda üç ana konfigürasyon dosyası dışında diğer dosyalarda da düzenlemeler yapılmaktadır. Bunlar sırasıyla aşağıdaki gibidir.

“yarn-site.xml” dosyasının içeriği şekil 37’ deki gibi yapılması gerekmektedir.

Şekil 37.“yarn-site.xml” dosyasının içeriği.

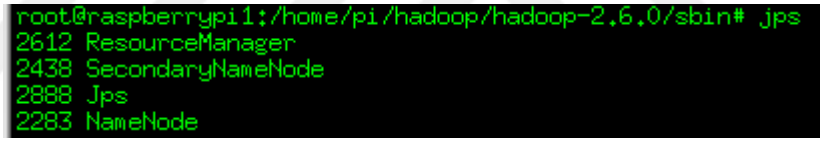
Bu konfigürasyon dosyalarındaki ayarın hepsi tüm sunucularda aynı olmalıdır. Hadoop sistemi ile ilgili ayarlar tamamlandıktan sonra HDFS dosya sisteminin oluşturulması için formatlanması gerekmektedir bunun için “hadoop namenode -format” komutunun kullanılması gerekmektedir. Format komutu ile HDFS dosya sistemi oluşturulmaktadır. Eğer tek düğümlü (Single Node) bir sistem kullanırken cluster bir sisteme geçilecek ise HDFS dosya sisteminin yeniden oluşturulması gerekmektedir.

Ayarların düzenlenmesinden sonra “./start-all.sh” komutunu kullanarak Hadoop servislerini başlatabilir. Servislerin çalışmasına ilişkin görünüm şekil 38’ de ki gibidir bu görüntüde hem master sunucunun hem de slave sunucularındaki servislerin başlatılması işlemleri görünmektedir.

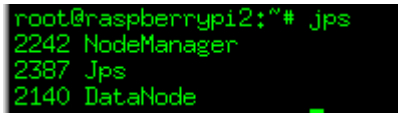


Şekil 38.Hadoop sistemindeki servislerin başlatılması.

Hadoop' un düzgün çalıştığının kontrol edilmesi için master sunucuda 3 adet slave sunucularda ise 2 adet servisin çalışmış olması gerekmektedir. Master sunucu için "jps" komutunun çıktısı şekil 39'da ki gibi, slave sunucular için "jps" komutunun çıktısı şekil 40' da ki gibi olması gerekmektedir.



Şekil 39.Master sunucu üzerinde çalışan Hadoop servislerinin görüntülenmesi.



Şekil 40.Slave sunucularında çalışan Hadoop servislerinin görüntülenmesi.

Bu bölüme gelindiğinde Hadoop sistemi çalışır duruma gelmiş olmaktadır. Sistemi test etmek için "hadoop dfs -put /tmp/test.txt /" komutunu kullanarak "/tmp/test.txt" dosyasını Hadoop root dizine kopyalanabilir. Eğer hata alınmadı ise HDFS dosya sistemine dosya kopyalama işlemi başarılı şekilde yapılmaktadır. Cluster kurulumu için kopya sayısını 4 adet olarak verilmektedir. Bu sebeple kopyalanan dosya her sunucuda birer tane olacak şekilde 4 kopya olarak HDFS dosya sisteminde yedeklenmektedir. "hadoop dfs -ls /" komutu ile HDFS dosya sisteminde root dizininin

listesi alınabilmektedir. “hadoop jar /hadoop/hadoop-1.2.1/hadoop-examples-1.2.1.jar wordcount /test.txt /test.out” komut satısını kullanarak “test.txt” dosyasında bulunan kelime sayısı öğrenilir ve bu komutlar ile sistem test edilebilir. Cluster sistemde yapılan herhangi bir işlem için sistem otomatik olarak işi paralel olarak çalıştırmaktadır fakat örnek olarak küçük bir dosya veya sadece bir adet dosya için işlem yapıldı ise sistem bunu paralel olarak çalıştırılmasının maliyetinin yüksek olacağını hesapladığında ilgili iş parçasını sadece yedek düğümlere (slave node) atamaktadır. Cluster sisteminde paralel çalışmayı test etmek için bir dizin altına 50-100 adet metin dosyası konularak “word count” uygulaması çalıştırılabilir. Sistem bu kadar çok sayıdaki dosyaları işleyebilmek için dosyaları düğümlere bölecek ve paralel olarak çalıştıracaktır. Bu işlem yazının yukarıdaki MapReduce bölümünde detaylı olarak anlatılmaktadır.

Hadoop sisteminin servislerine ait port bilgileri tablo 3’ de gösterilmektedir. Hadoop servislerinin çalıştırmak ve durdurmak için kullanılan scriptler ve açıklamaları tablo 4’ de açıklanmaktadır. Hadoop konfigürasyon dosyaları ve açıklamaları tablo 5 ‘ de gösterilmektedir.

Servis (Daemon)	Port
NameNode	50070
Secondary NameNode	50090
JobTracker	50030
DataNode(s)	50075
TaskTracker(s)	50060

Tablo 3.Servislerin port bilgiler.

Script	Açıklama
start-dfs.sh	NameNode, Secondary NameNode ve DataNode Servislerinin başlatılması
stop-dfs.sh	NameNode, Secondary NameNode ve DataNode Servislerinin durdurulması
start-mapred.sh	JobTracker ve TaskTracker Servislerinin başlatılması
stop-mapred.sh	JobTracker ve TaskTracker Servislerinin durdurulması

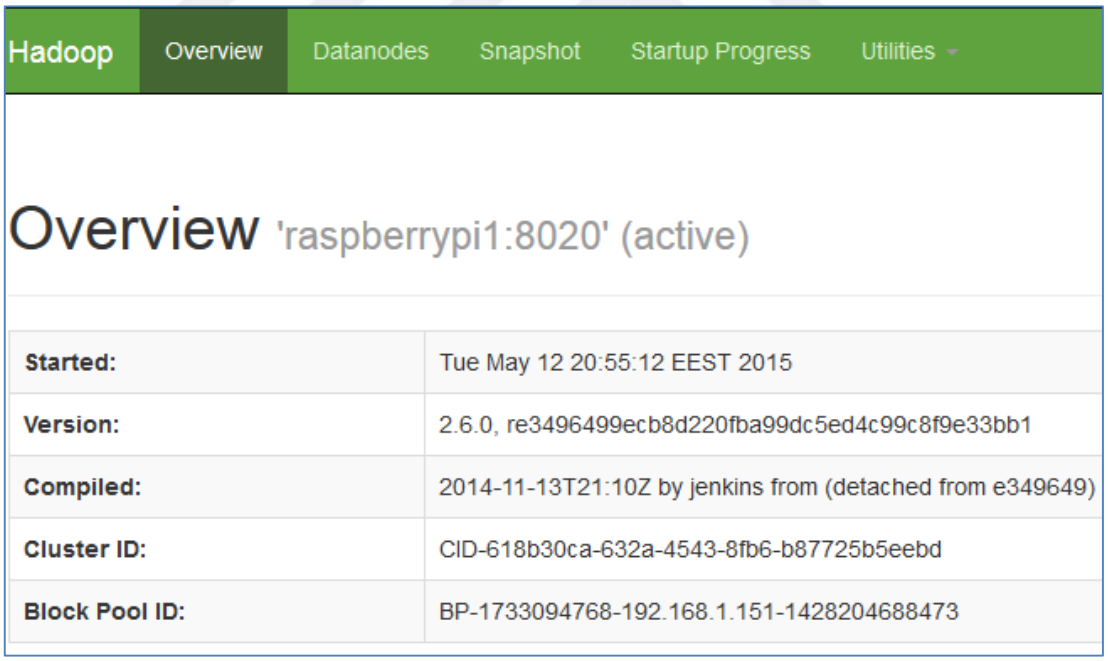
Tablo 4.Hadoop sistemine ait script dosyalarının bilgileri.

Konfigurasyon Dosya İsmi	Açıklama
/etc/hadoop/core-site.xml	NameNode ve JobTracker için genel ayarlar
/etc/hadoop/hdfs-site.xml	HDFS file system için ayarlar
/etc/hadoop/mapred-site.xml	MapReduce servisi ve job' lar için ayarlar
/etc/hadoop/hadoop-env.sh	Yapılandırma ayarları. Java, SSH vb.
/etc/hadoop/master	Master node tanım dosyası (Cluster kurulumda gereklidir.)
/etc/hadoop/slaves	Slave sunucuların tanım dosyası (Cluster kurulumda gereklidir.)

Tablo 5.Hadoop sistemine ait konfigürasyon dosyaları.

Hadoop sisteminde kurulum ile birlikte gelen web uygulamaları bulunmaktadır. Bu web ara yüzleri ile sistemde ait birçok bilgi alınabilmektedir. Web uygulamalarına erişim adresleri aşağıdaki gibidir.

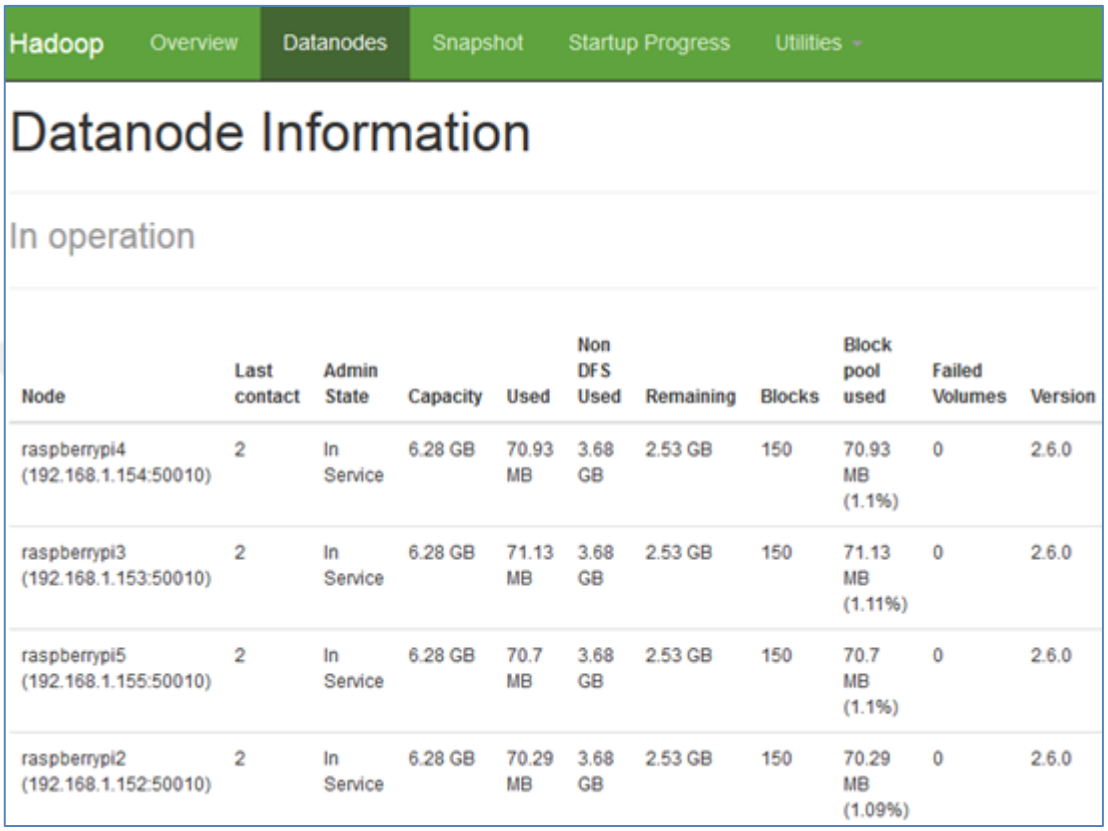
http://192.168.1.151:50070 – NameNode web ara yüzü şekil 41' deki gibidir.



Şekil 41.Hadoop NameNode web ara yüzü sayfası.

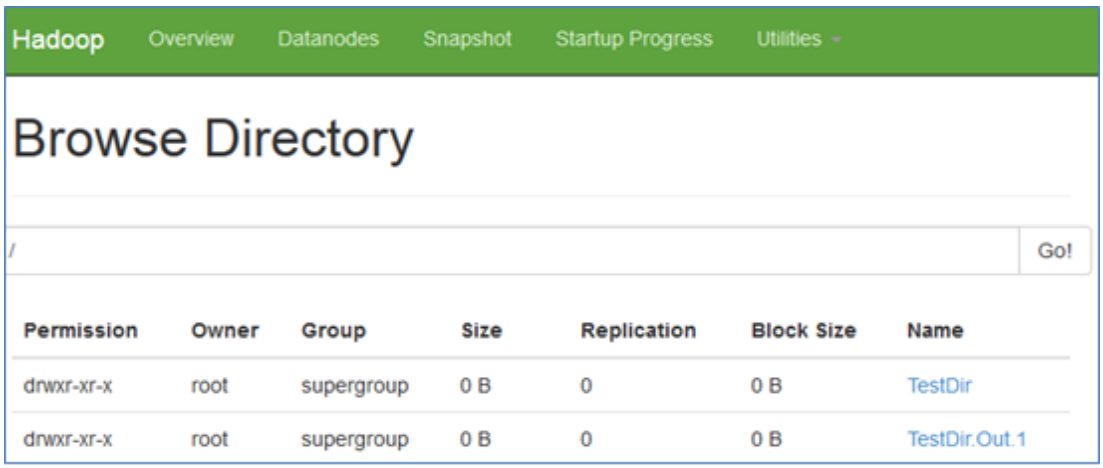
NameNode web ara yüzünü kullanarak sistemin genel bilgisini ve durumunu, bağlı olan düğümleri ve bu düğümlerin durumlarını (aktif, pasif vb.), dosya sisteminin kapasite kullanımını, sistemdeki loglar hakkında bilgiler alınabilmektedir ayrıca dosya

sistemine erişim yapılabilmektedir. Dosya sistemine erişim ile web ara yüzü üzerinden dosyaların bulunduğu düğümlerin bilgisini ve dosyanın indirilmesi mümkündür.



Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
raspberrypi4 (192.168.1.154:50010)	2	In Service	6.28 GB	70.93 MB	3.68 GB	2.53 GB	150	70.93 MB (1.1%)	0	2.6.0
raspberrypi3 (192.168.1.153:50010)	2	In Service	6.28 GB	71.13 MB	3.68 GB	2.53 GB	150	71.13 MB (1.11%)	0	2.6.0
raspberrypi5 (192.168.1.155:50010)	2	In Service	6.28 GB	70.7 MB	3.68 GB	2.53 GB	150	70.7 MB (1.1%)	0	2.6.0
raspberrypi2 (192.168.1.152:50010)	2	In Service	6.28 GB	70.29 MB	3.68 GB	2.53 GB	150	70.29 MB (1.09%)	0	2.6.0

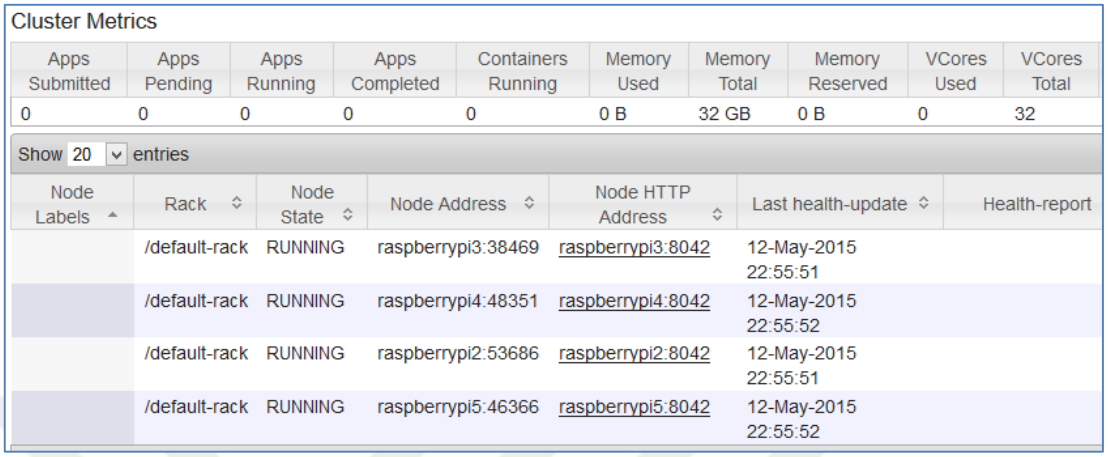
Şekil 42.Hadoop NameNode web ara yüzü "Datanode Information" sayfası.



Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	0	0 B	TestDir
drwxr-xr-x	root	supergroup	0 B	0	0 B	TestDir.Out.1

Şekil 43.Hadoop NameNode web ara yüzü "Browse Directory" sayfası.

http://192.168.1.151:8088 – Yarn web ara yüzü şekil 44’ deki gibidir.



Şekil 44.Hadoop Yarn web ara yüzü sayfası.

http://raspberrypi3:8042 – Adresinden tüm düğümler üzerindeki uygulamalara ait bilgiler görünmektedir. Bu adres her düğüm üzerinde ayrı ayrı olarak çalışmaktadır.

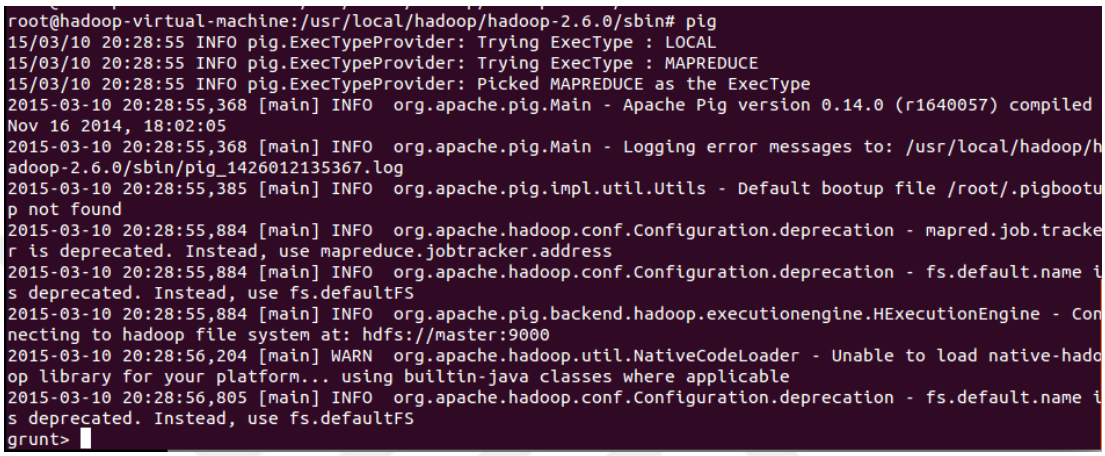
4.6.3 Apache Pig Kurulumu

Apache Pig kurulumu ARM sistemlerde ve x86 mimarileri için aynı şekildedir farklı mimariler için ayrı sürümler çıkarılmamıştır mevcut sürüm tüm sistemlerde çalışmaktadır. Bu tez çalışmasında Apache Pig kurulumu için güncel sürüm olan 0.14.0 sürümü kullanılmaktadır. Pig’ in kurulumunun yapılacağı dizine örnek olarak “/usr/local/hadoop/pig-0.14.0/” dizini altında indirilen kurulum açılmalıdır. Mevcut sürümü “http://ftp.itu.edu.tr/Mirror/Apache/pig/pig-0.14.0/” adresinden indirilmektedir. Pig’ in komutlarına her yerden erişebilmek için PATH değişkenine Pig’ e ait dosyaların bulunduğu dizinin şekil 45’ deki gibi eklenmesi gerekmektedir.

```
# Pig Ayarlari ================================================
PIG_HOME=/usr/local/hadoop/pig-0.14.0/
PATH=$PATH:$PIG_HOME/bin
```

Şekil 45.Apache Pig ayarları.

Pig' e ait ayarların tamamlanmasından sonra Pig konsoluna bağlantı yapılabilir. Komut satırında pig yazarak pig uygulaması çalıştırılmaktadır. Pig konsolu açıldığında komut satırında şekil 46' da ki gibi "grunt>" belirecektir bu aşamada pig uygulaması çalışır duruma gelmektedir.



Şekil 46.Apache Pig uygulamasına bağlanılması.

Apache Pig için örnek uygulamalar Pig ile ilgili bölümde gösterilmektedir.

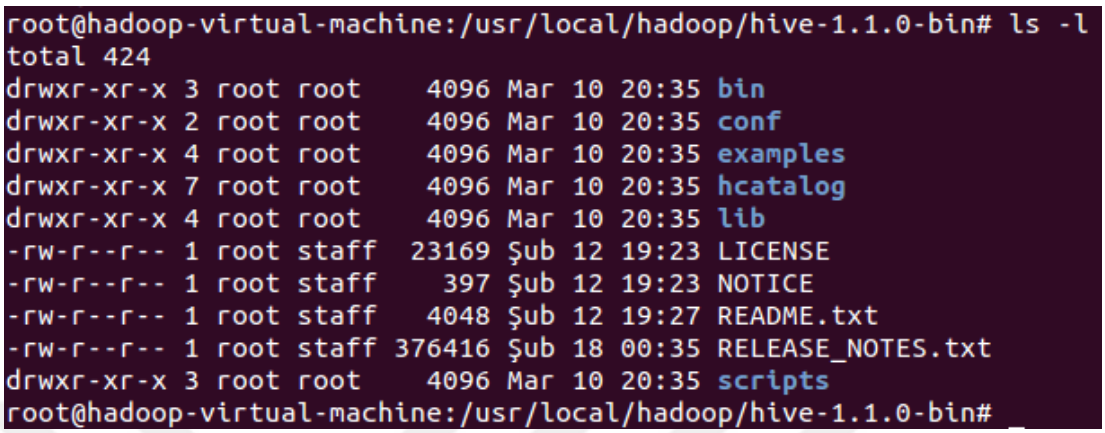
4.6.4 Apache Hive Kurulumu

Apache Hive kurulumu Apache Pig uygulamasındaki gibi ARM ve x86 mimarileri için aynı şekildedir farklı sistem mimarileri için ayrı sürümler çıkarılmamıştır mevcut sürüm tüm sistemlerde çalışmaktadır. Bu tez çalışmasında Apache Hive kurulumu için güncel sürüm olan 1.1.0 sürümü kullanılmaktadır. Hive kurulumu için "http://ftp.mku.edu.tr/apache-dist/hive/hive-1.1.0/apache-hive-1.1.0-bin.tar.gz" adresinden Hive ile ilgili paketi indirilip "/usr/local/hadoop/hive-1.1.0-bin/" altına açılır. Hive için PATH ayarları şekil 47' deki gibi yapılması gerekmektedir.

```
HIVE_HOME=/usr/local/hadoop/hive-1.1.0-bin/
PATH=$PATH:$HIVE_HOME/bin
```

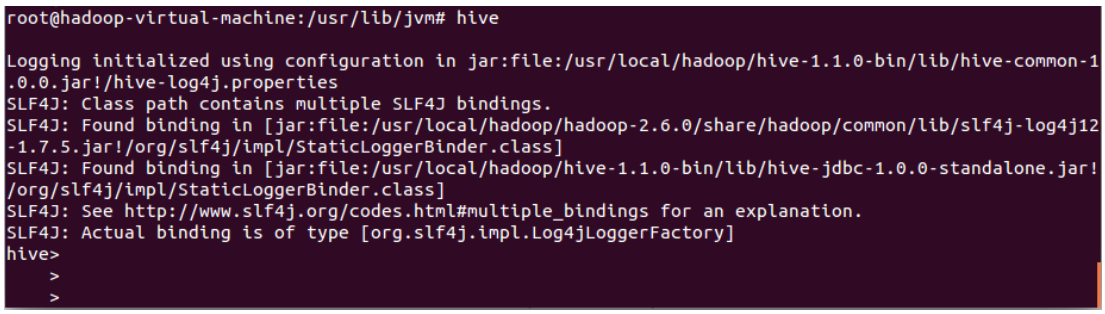
Şekil 47.Apache Hive kurulum dizini ve PATH ayarları.

Hive dizinini altındaki dosya ve dizin yapısı şekil 48' deki gibi oluşmaktadır.



Şekil 48.Apache Hive uygulamasının dizin ve dosyaları.

Hive çalışırken oluşan metadata bilgisini hangi dizin içerisinden Hive' ı çalıştırırsak ilgili klasöre altına yazmaktadır. Bu sebeple hive_datawarehouse isimli bir klasör oluşturup workspace mantığında her zaman bu klasör içinde çalışmak geçmiş bilgilerinde bir yerde toplanması bakımında daha kullanışlı olmaktadır. Bunun için gerekli dizinlerin ve hakların tanımlanması gerekmektedir. Bu tanımlar yapıldıktan sonra "hive" komutu ile Hive uygulaması çalıştırılabilir.



Şekil 49.Apache Hive uygulamasının başlatılması.

Apache Hive uygulaması çalıştığında Hive konsolu ekrana gelecektir, Hive konsolu "hive>" komut satırı ile başlamaktadır. Apache Hive için örnek uygulamalar Hive ile ilgili bölümde gösterilmiştir.

4.7 Sistemlerin Performans Bilgilerinin Toplanması

4.7.1 Sar Monitoring Paketinin Kullanımı

Sar komutu sisteme ait anlık CPU kullanımını vermektedir bu bilgiyi verirken kullanıcıların ve sistemin ne kadar kaynak tükettiğini de ayrıca vermektedir. Sar komutu disk için I/O değerlerini de vermektedir. Sar komutu "`apt-get install sysstat`" komutları ile sisteme yüklenebilmektedir ve Raspberry Pi sistemlerinde çalışmaktadır. Sar komutu geçmiş bilgiyi de saklayabilmektedir. Sar komutu bir çok parametre alabilmektedir genel kullanımı "sar 2 10" şeklindedir buradaki 2 zaman aralığı 10 ise kaç adet olacağını belirler. Örnek sar komut çıktısı şekil 50' deki gibidir. Sar komutundan gelen değerler farklı bir dosyaya aktarılabilir bu sayede grafiksel olarak raporlama veya analizler yapılabilir.

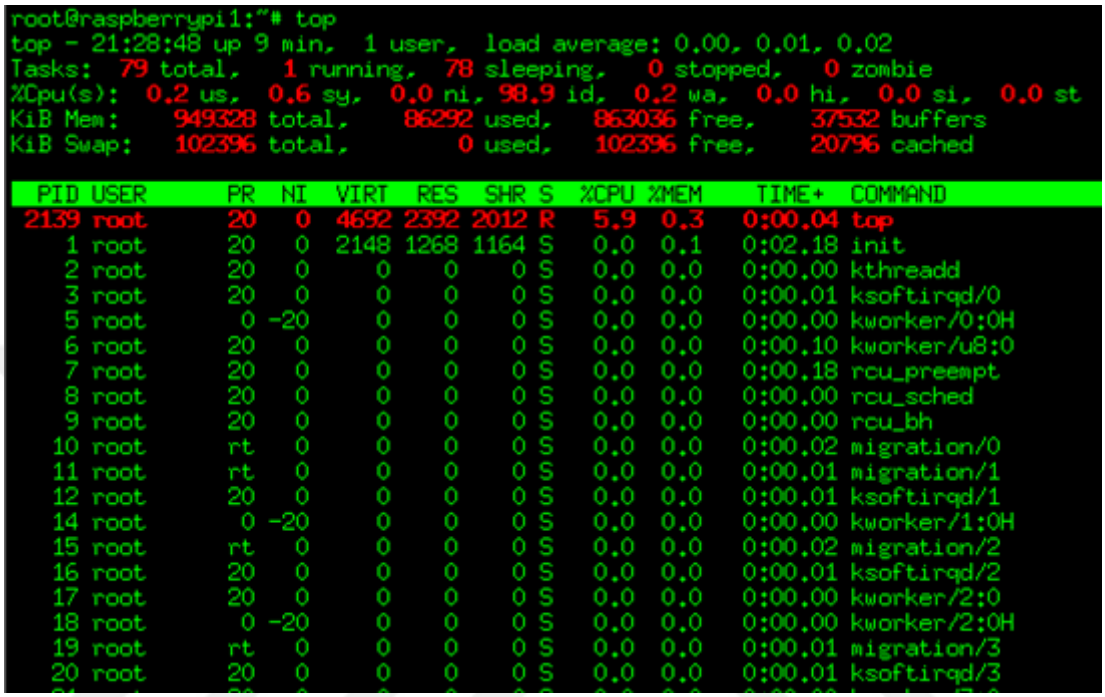
```
root@raspberrypi1:~# sar 1 10
Linux 3.18.7-v7+ (raspberrypi1)         05/13/2015      _armv7l_        (4 CPU)

09:20:41 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
09:20:42 PM     all      0.00      0.00      0.25      0.00      0.00     99.75
09:20:43 PM     all      0.00      0.00      0.25      0.00      0.00     99.75
09:20:44 PM     all      0.00      0.00      0.25      0.00      0.00     99.75
09:20:45 PM     all      0.00      0.00      0.25      0.00      0.00     99.75
09:20:46 PM     all      0.00      0.00      0.25      0.00      0.00     99.75
09:20:47 PM     all      0.00      0.00      0.50      0.00      0.00     99.50
09:20:48 PM     all      0.25      0.00      0.25      0.00      0.00     99.50
09:20:49 PM     all      0.25      0.00      0.00      0.00      0.00     99.75
09:20:50 PM     all      0.00      0.00      0.25      0.00      0.00     99.75
09:20:51 PM     all      0.25      0.00      0.00      0.00      0.00     99.75
Average:        all      0.08      0.00      0.23      0.00      0.00     99.70
root@raspberrypi1:~#
```

Şekil 50.Sar komut çıktısı.

4.7.2 Top Monitoring Paketinin Kullanımı

Top komutu sisteme ait CPU, Memory, Swap ve Proses bilgilerini vermektedir. Anlık olarak çalışır, üst kısımda kaynak kullanımlarını alt kısmında ise aktif prosesleri ve bunların kaynak tüketimlerini vermektedir. Top komutu Raspberry Pi işletim sisteminde yüklenmiş olarak gelmektedir. "top" yazıp çalıştırılmaktadır istenilirse

parametreler ile güncelleme zaman aralığı değiştirilebilir. Zaman aralığı verilmez ise 1 saniye aralıklar ile güncelleme yapmaktadır.



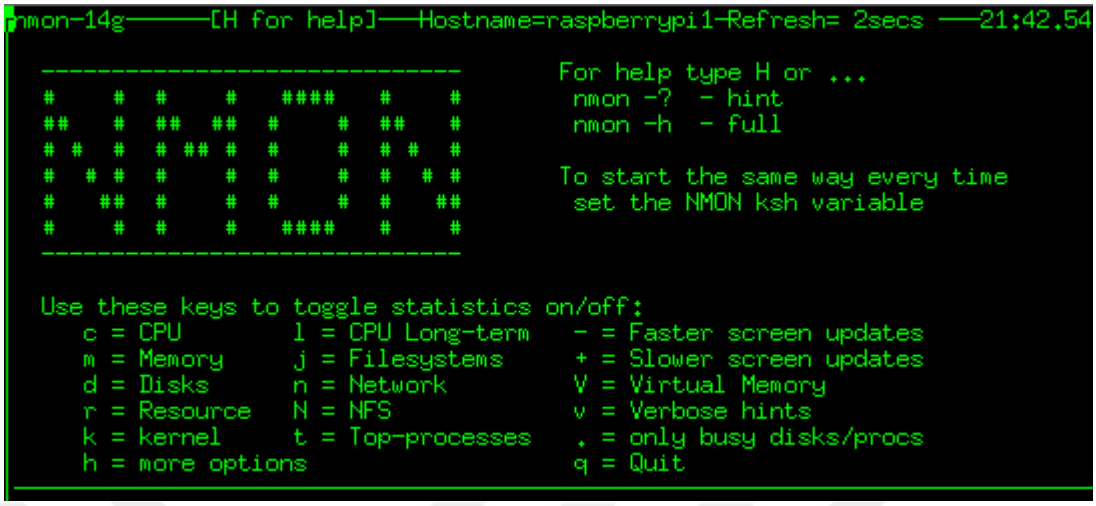
Şekil 51.Top komut çıktısı.

4.7.3 Nmon Monitoring Uygulamasının Kullanımı

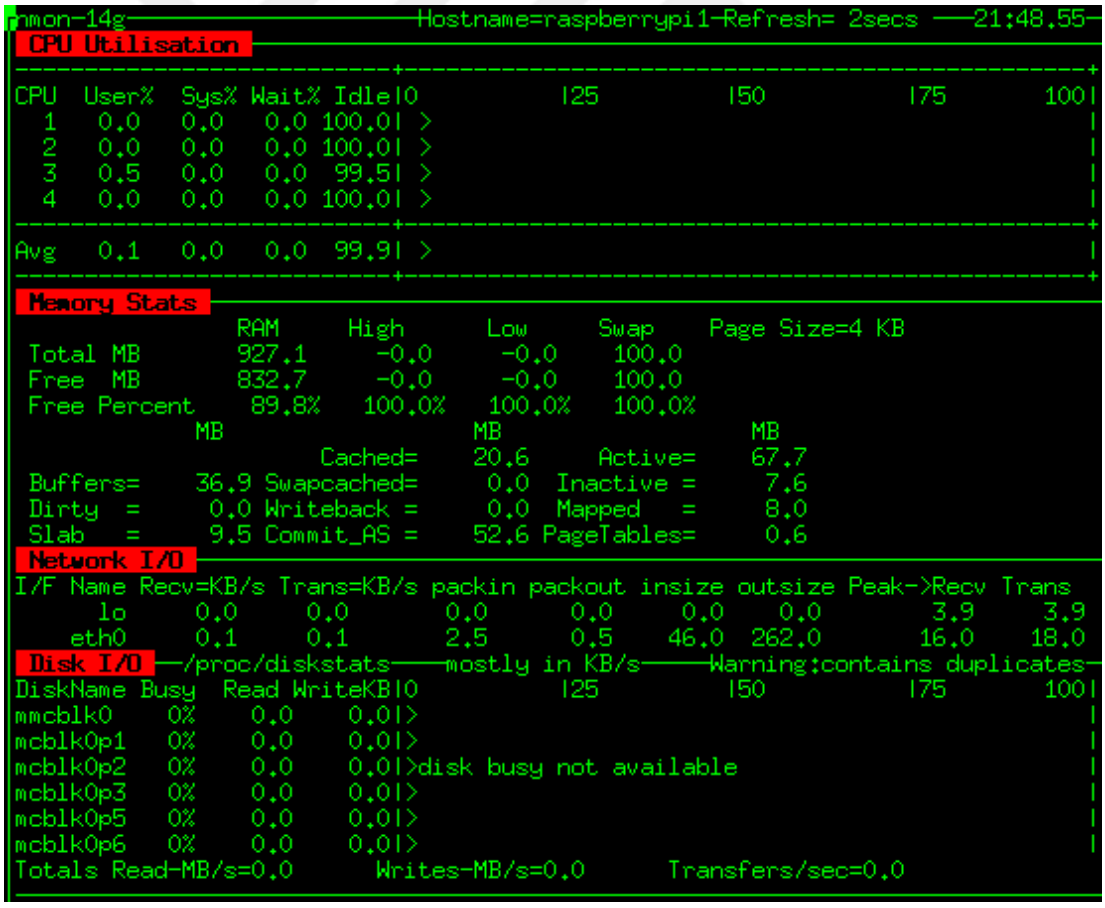
Nmon açık kaynak kodlu olarak geliştirilen ve terminal üzerinde çalışabilen detaylı sistem kaynak izleme uygulamasıdır. Raspberry Pi sisteminde önceden yüklenmiş olarak gelmemektedir. ARM mimarisi için derlenmiş sürümü bulunmaktadır. "http://sourceforge.net/projects/nmon/files/nmon_pi.zip" adresinden paketlerini indirip "/bin/" içerisine kopyalandığında "nmon_pi" komutu ile çalışmaktadır. Nmop uygulaması tablo 6' da ki kaynak bilgilerini verebilmektedir.

c = CPU	l = CPU Long-term	r = Resource	N = NFS
m = Memory	j = Filesystems	k = kernel	t = Top-processes
d = Disks	n = Network		

Tablo 6.Nmop uygulamasının izlediği kaynakların bilgisi.



Şekil 52.Nmop uygulamasının giriş ekranı.



Şekil 53.Nmop kaynak kullanımlarının gösterimi ekranı.

4.7.4 Ganglia Monitoring Uygulamasının Kullanımı

Ganglia uygulaması detaylı kaynak ve performans izleme uygulamasıdır. Web tabanlı ara yüzü bulunmaktadır. Sunucu tarafında yüklenen client paketi ile kaynak bilgilerini toplamaktadır. Sunucular tarafında sadece bir servis (gmond) çalışmaktadır. Web ara yüz uygulaması herhangi bir sunucuya kurulabilmektedir. Bu tez çalışmasında Raspberry Pi cihazlarının tümüne gmond paketi kurulmaktadır. Ganglia web uygulaması ise farklı bir sanal sunucuya kurulmaktadır. Şekil 54'deki komutların bilgileri toplayacak master sunucuda çalıştırılması ile web ara yüz uygulama kısmı kurulmaktadır. Ganglia uygulamasının web ara yüzü PHP ile geliştirildiği için sunucuya Apache ve PHP yüklenmesi gerekmektedir.

```
sudo apt-get install apache2 php5 libapache2-mod-php5 php5-json
sudo apt-get install ganglia-frontendweb gmetad gmetad rrdtool ganglia-monitor
```

Şekil 54.Nmop web uygulamasının kurulum komutları.

Kurulum işlemi tamamlandıktan sonra "/etc/ganglia/gmetad.conf" dosyasında bazı düzenlemelerin yapılması gerekmektedir. Bunlar sırası ile aşağıdaki gibidir.

- Sisteme ait tanımlama bilgileri;

```
cluster {

  name = "my cluster"     ## Cluster name

  owner = "unspecified"

  latlong = "unspecified"

  url = "unspecified"

}
```

- Sistemin master sunucuya bağlantı bilgileri;

Master Sunucu için;

```
udp_send_channel  {
 #mcast_join = 239.2.11.71 ## comment out
 host = localhost
 port = 8649
 ttl = 1
}
```

Düğümler için;

```
udp_send_channel {
 #mcast_join = 239.2.11.71  ## Comment
 host = 192.168.1.252  ## IP address of master node
 port = 8649
 ttl = 1
}
```

- Master sunucusun port bilgilerinin ayarlanması; bu bölüm sadece master sunucuda olması gerekmektedir. Düğümlerde bu bölümün "#" ile kapatılması gerekmektedir.

```
udp_recv_channel {

 #mcast_join = 239.2.11.71 ## comment out

 port = 8649

 #bind = 239.2.11.71 ## comment out

}
```

- Ağ bağlantısında kullanılacak Port ayarları;

```
tcp_accept_channel {

 port = 8649

}
```

Bu ayarlar yapıldıktan sonra önce master sunucudaki uygulamanın çalıştırılması gerekmektedir. Şekil 55' deki komut satırı kullanılarak web uygulaması, şekil 56' da ki komutlar ile düğümlerdeki Ganglia servisi çalıştırılmaktadır.

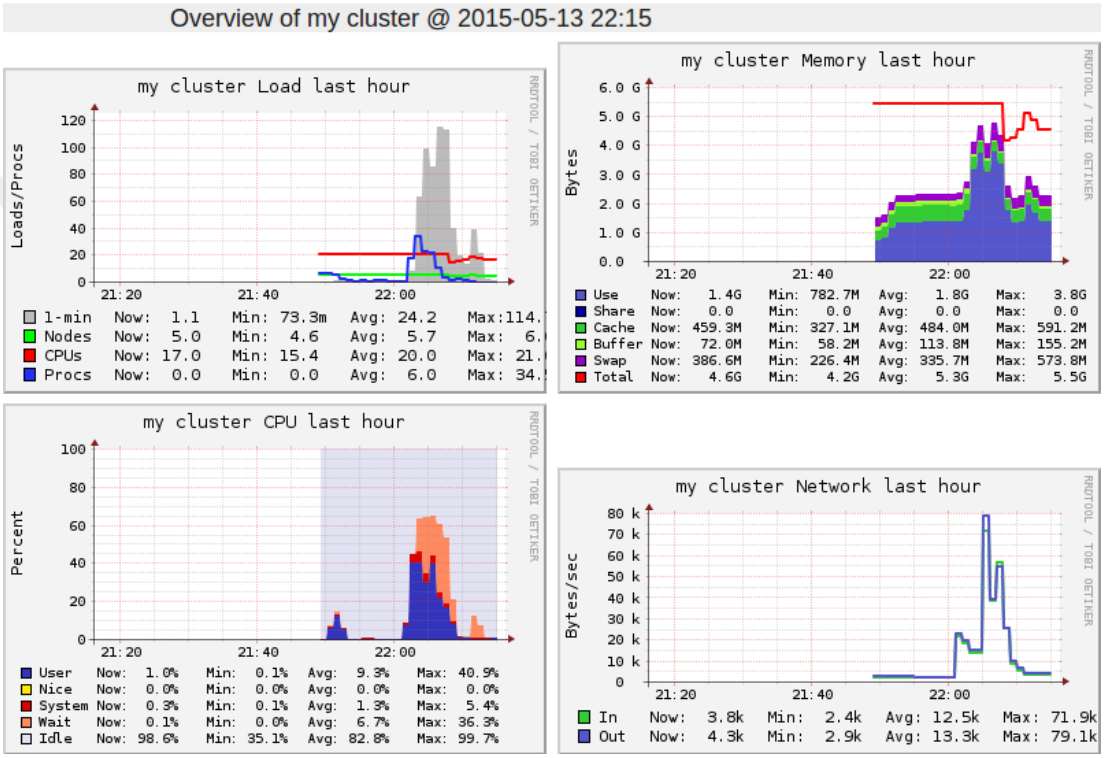
```
sudo service ganglia-monitor restart && sudo service gmetad restart && sudo service apache2 restart
```

Şekil 55.Nmop web uygulamasının kurulum komutları.

```
sudo service ganglia-monitor restart
```

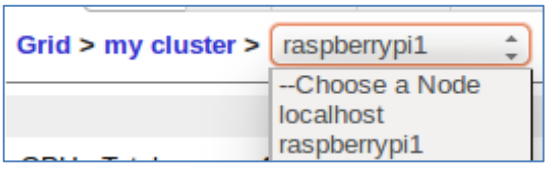
Şekil 56.Nmop web uygulamasının kurulum komutları.

Ganglia uygulamasında yukarıdaki ayarların dışında çok farklı ayarlarda yapılabilmektedir. Daha fazla detayda ayarlamalar için "Monitoring with Ganglia" isimli kitabın incelenmesi önerilmektedir. Şekil 57' deki ekran görüntüsü Ganglia giriş sayfasını göstermektedir. Düğümlerden bilgiler geldikçe anlık olarak kaynak kullanımlarına ait bilgiler tablolarda ve grafikler üzerinde görünmektedir.



Şekil 57.Ganglia web uygulaması giriş sayfası.

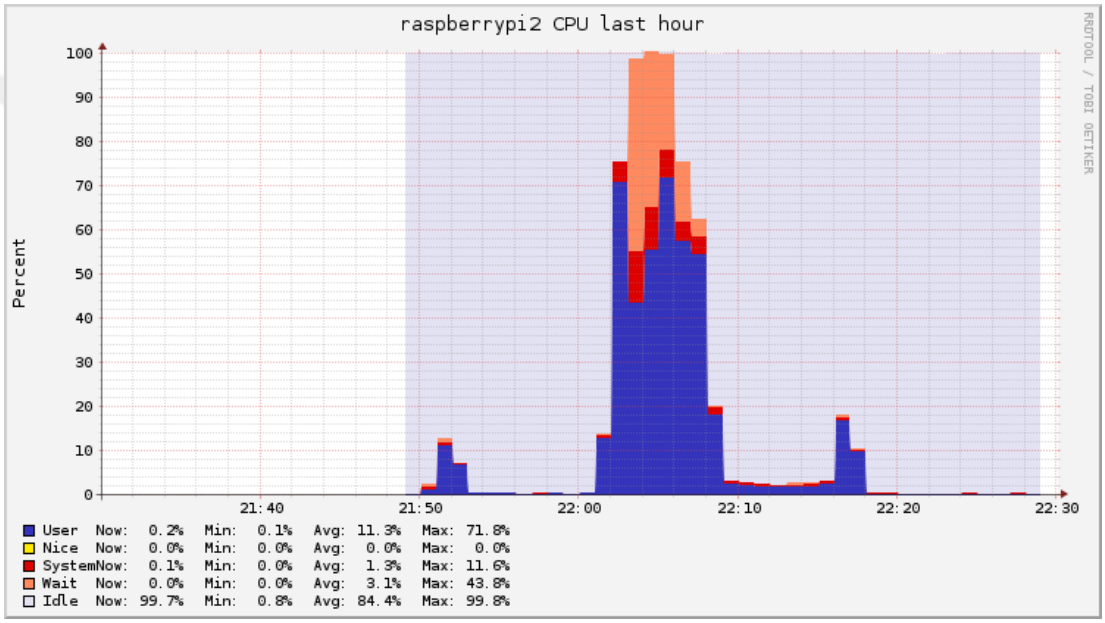
Ganglia giriş sayfasında tüm bağlı düğümlerin oluşturduğu toplam kaynak kullanımı gösterilmektedir. Şekil 58' deki gibi tek sunucuya ait bilgilerde seçilebilir.



Şekil 58.Sunucu seçim.

Aşağıda yer alan Şekil 59 ve Şekil 60’ da ki gibi detaylı ve daha kısa zamanlı raporlar alınabilmektedir. Uygulama bu bilgileri geriye dönük olarak da kullanılacak şekilde saklamaktadır.

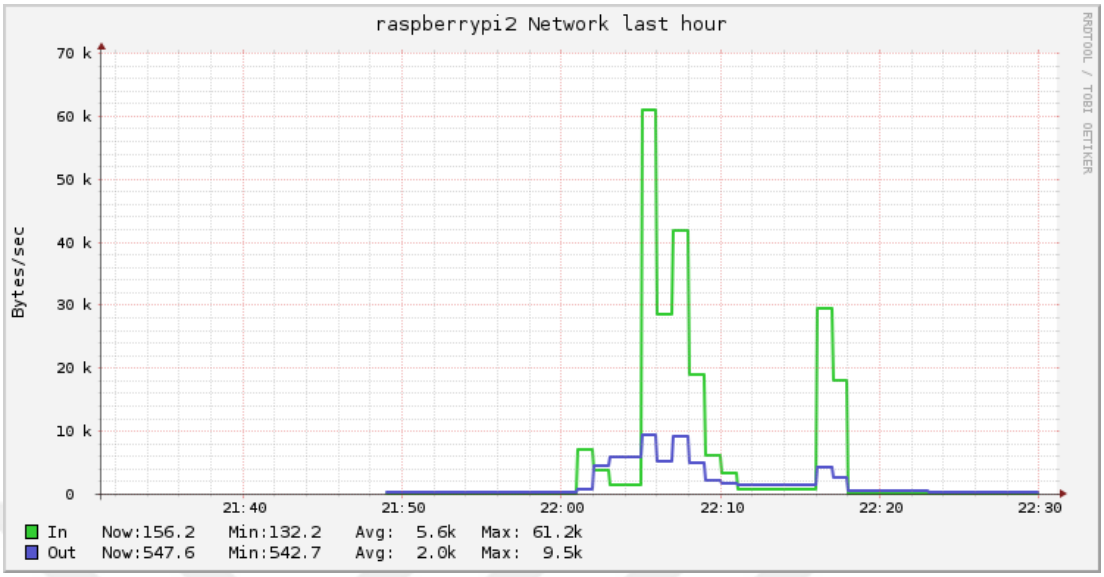
Şekil 59’ da “raspberrypi2” cihazına ait son bir saatlik işlemci kullanımı bilgisi görünmektedir. Bu grafikte görüldüğü gibi MapReduce uygulaması çalıştırıldığı zaman aralığında işlemci %100 performans ile kullanılmaktadır.



Şekil 59.Sunucuya ait CPU kaynak kullanım bilgisi.

Bu tez çalışmasındaki öncelikli hedef işlemcilerin maksimum seviyede kullanabilmek olduğu için uygulama ve Hadoop konfigurasyonu bu şekilde yapılmaktadır. Ganglia uygulaması işlemci bilgisi dışında sisteme ait birçok kaynak için bilgi sağlamaktadır.

Şekil 60’ da ki grafik uygulama çalışma zamanındaki network kullanımı bilgisini göstermektedir.



Şekil 60.Sunucuya ait ağ kaynak kullanım bilgisi.

5. SİSTEMİN TEST EDİLMESİ

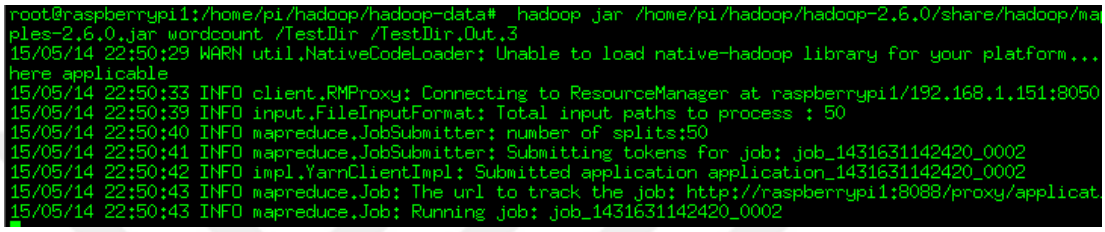
5.1 Kullanılan Veri Setleri

Test işlemleri Raspberry Pi cihazlarının CPU güçlerinin en üst seviyede kullanacak şekilde planlanmıştır bu sebeple yüksek paralel işlemlerin çalışmasına yönelik veri setleri oluşturulmuştur. Veri setleri kitapların metin dosyalarını seçerek oluşturulmuştur. Amaç bu metin şeklindeki kitap dosyalarında kelimelerin kaç kez tekrarlandığının hesaplanmasıdır. Test işlemindeki öncelik ise bu uygulama için MapReduce işleminin paralel çalışarak tüm sistemlerdeki yük dağılımını ve işlemin tamamlanma zamanı olacaktır.

Test öncesi hazırlık işlemleri aşağıdaki gibidir. Öncelikle temin edilen metin dosyalarının Hadoop HDFS dosya sistemine kopyalanması gerekmektedir. “smallfile.txt” isminde 1.2 MB boyutundaki dosyadan 50 adet kopya hazırlanmıştır. Bu dosyalar “hadoop dfs -put /home/pi/hadoop/hadoop-data/dosyalar/* /TestDir/” komutu ile HDFS dosya sisteminde oluşturulan “TestDir” dizini altına kopyalanmıştır. Test için kullanılan “wordcount” uygulaması Hadoop kurulumu ile birlikte “hadoop-

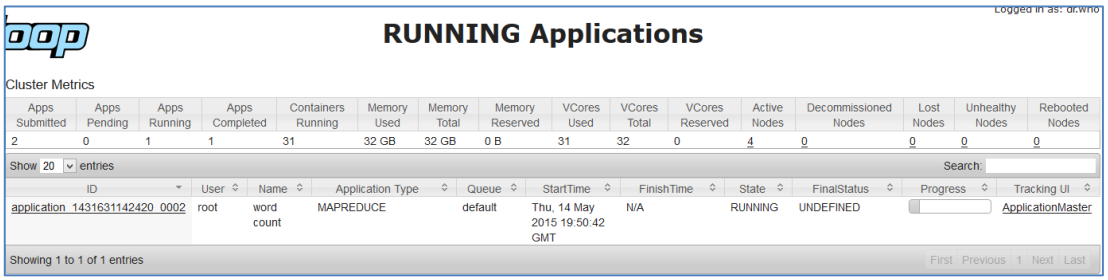
mapreduce-examples-2.6.0.jar" içerisinde gelmektedir. Aşağıdaki komut satırı çalıştırılarak uygulama başlatılmaktadır. Uygulama çalışmaya başladığında şekil 61' deki gibi ekran görünümü oluşmaktadır.

"hadoop jar /home/pi/hadoop/hadoop-2.6.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /TestDir /TestDir.Out"

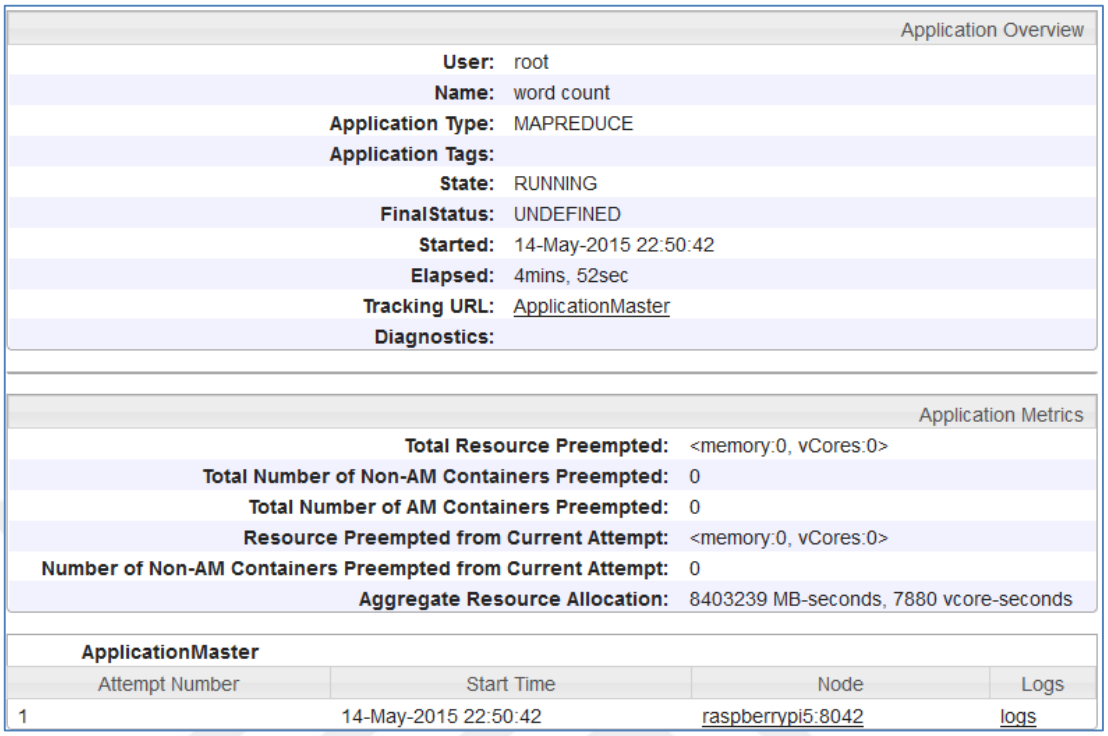


Şekil 61.Test işlemi için "wordcount" örnek uygulama çalıştırılması.

Çalışan uygulamaya ait bilgileri web ara yüzünden de takip edilebilmektedir. Şekil 62' da sistem, kaynaklara ve çalışan uygulamaya ait bilgiler, şekil 63' de ise uygulamaya ait detaylı bilgiler yer almaktadır.

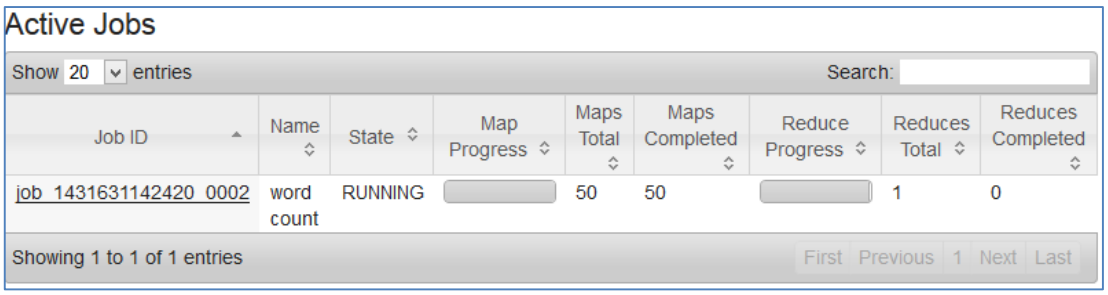


Şekil 62.JobTracker web ara yüzü.



Şekil 63.Çalışan uygulamaya ait detaylı bilgiler.

Şekil 64' de sistem üzerinde çalışan tüm aktif uygulamalar ve bu uygulamalar için MapReduce işlemlerinin paralellik durumları ve uygulamaların tamamlanma oranları görünmektedir.



Şekil 64.Çalışan aktif uygulamalar.

5.2 Dağıtık Çalıştırılan Uygulama

Hadoop sisteminin Java tabanlı olması sebebi ile üzerinde çalıştırılmakta olan MapReduce uygulamaları da temel yapı olarak Java platformunda geliştirilmektedir. Hadoop sisteminin farklı yazılım dillerini de destekleme özelliği sayesinde C, Python, .Net gibi yazılım platformları ile ayrıca Apache Pig ve Apache Hive kullanılarak yüksek seviyede MapReduce uygulamaların yazılması mümkündür.

Bu tez çalışmasında Hadoop kurulumu içerisinde hazır olarak gelen "WordCount" Java uygulaması kullanılmaktadır. Bu uygulama ile HDFS sisteminde bulunan ve uygulamaya parametre olarak verilen metin dosyaları içerisindeki kelimelerin sayıları bulunabilmektedir. Kullanılan "WordCount" Java veya benzeri bir dilde yazılması kolay bir örnek olarak görünmesine rağmen verilecek veri seti boyutunun 1 Petabyte olması durumunda normal bir bilgisayar ile ve kullanılacak herhangi bir yazılım dili ile bunu işleyebilmenin çok maliyetli olacağı bilinmektedir. Kullanılan "WordCount" örneği ile çok büyük boyutlu bir metin dosyasını Hadoop kümesi üzerinde MapReduce yapısı kullanılarak işlemek mümkündür. Örnek uygulamada Map ve Reduce fonksiyonlarının kullanılması için Mapper ve Reducer sınıflarını kullanarak üretilmiş birer sınıf kullanılmaktadır. Koddaki main bölümü sınıf uygulamalarının çalışması için gereken başlangıç parametrelerinin alınması işlemlerini yerine getirmektedir.

Uygulama çalışmaya başladığında verilen metin dosyalarını satır satır okuyarak Mapper sınıfının map metodunda bulunan value parametresine aktarmaktadır. Map metodu satırlardaki kelimeleri ayırarak context objesine aktarmakta ve değerini 1 olarak yazmaktadır. Örnek olarak tablo 7'deki gibi bir metin dosyası kullanılması durumunda value nesnesinin map metodundan sonraki çıktısı tablo 8' deki gibi olmaktadır.

Maltepe Üniversitesi
Maltepe Üniversitesi Fen Bilimleri Enstitüsü
Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü

Tablo 7.Örnek Metin Dosyası.

Maltepe	1
Üniversitesi	1
Maltepe	1
Üniversitesi	1
Fen	1
Bilimleri	1
Enstitüsü	1
Maltepe	1
Üniversitesi	1
Fen	1
Bilimleri	1
Enstitüsü	1
Bilgisayar	1
Mühendisliği	1
Bölümü	1

Tablo 8.Metin Dosyasının işlenmesi aşamasında Map işlemine ait sonuç.

Map işleminde sonra aynı kelimler bir araya getirilerek Reducer sınıfının reduce metoduna aktarılmaktadır. Bu aşamada kelimeler reduce sınıfındaki key nesnesine aktarılmaktadır. Values nesnesine ise bu kelimelerin değerleri yazılmaktadır. Örnek olarak Mapper tablo 8' deki örnekte yer alan metindeki "Maltepe" kelimesi reduce metoduna aktarıldığında values nesnesi içerisinde 3 adet 1 değerindeki sayı bulunmaktadır. Reduce metodu bu değerleri toplayarak context nesnesine yazmaktadır. Reduce işlemi tamamlandığında sonuç tablo 9' da ki gibi olmaktadır.

Maltepe	3
Üniversitesi	3
Fen	2
Bilimleri	2
Enstitüsü	2
Bilgisayar	1
Mühendisliği	1
Bölümü	1

Tablo 9.Metin Dosyasının işlenmesi aşamasında Recude işlemine ait sonuç.

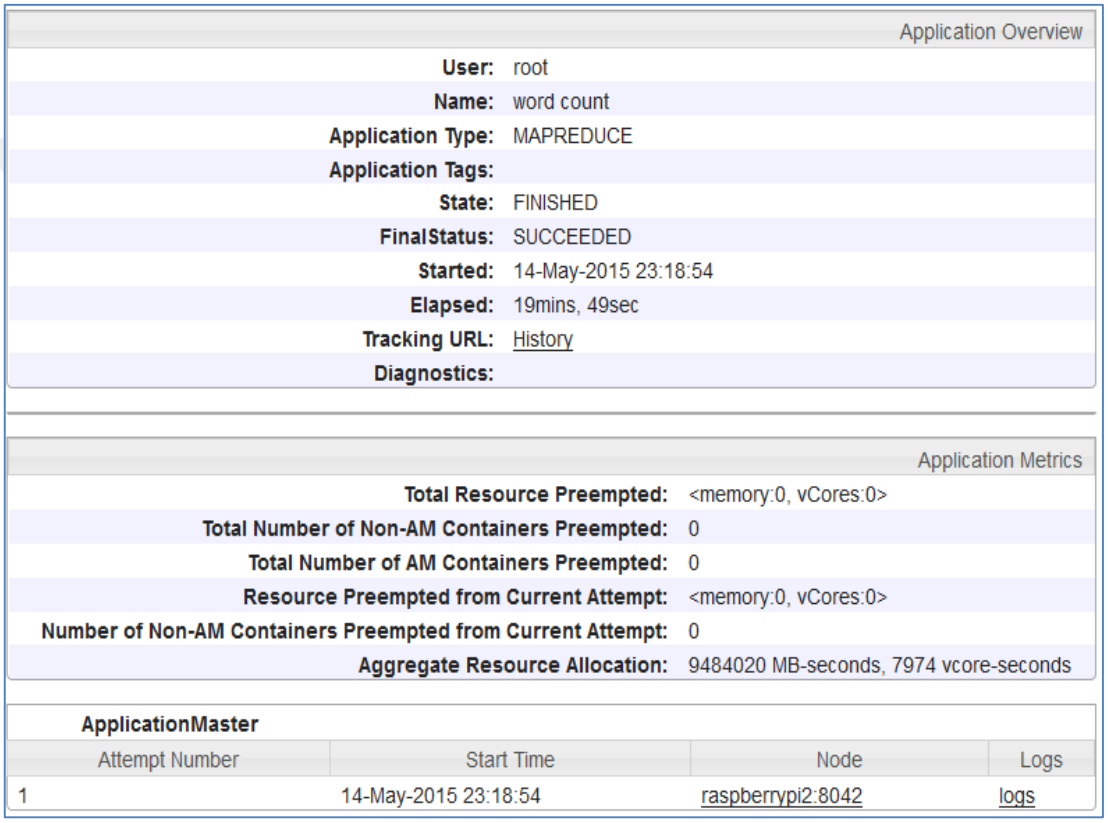
Java ile geliştirilmiş "WordCount" uygulamasına ait kodların tamamı aşağıdaki gibidir.

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat
;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
;

public class WordCount {

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

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

 public static class Reduce extends Reducer<Text,
IntWritable, Text, IntWritable> {
```

```
    public void reduce(Text key, Iterable<IntWritable>
values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
 }

 public static void main(String[] args) throws
Exception {
    Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

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

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new
Path(args[0]));
    FileOutputFormat.setOutputPath(job, new
Path(args[1]));

    job.waitForCompletion(true);
 }

}
```

5.3 Test işleminin 1 Düğüm Açık Diğerleri Kapalı Şekilde Yapılması

Wordcount uygulaması tek düğüm (Single Node) üzerinde çalıştırıldığında toplam 19 dk.49 sn.' lik bir sürede tamamlanmıştır. Uygulama çalışma süresinde sistem üzerindeki kaynaklara ait kullanım bilgileri Ganglia uygulaması ile toplanmıştır. Şekil 65' de uygulamanın çalışma zamanına ait bilgiler yer almaktadır.



Application Overview

User:	root
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	14-May-2015 23:18:54
Elapsed:	19mins, 49sec
Tracking URL:	History
Diagnostics:	

Application Metrics

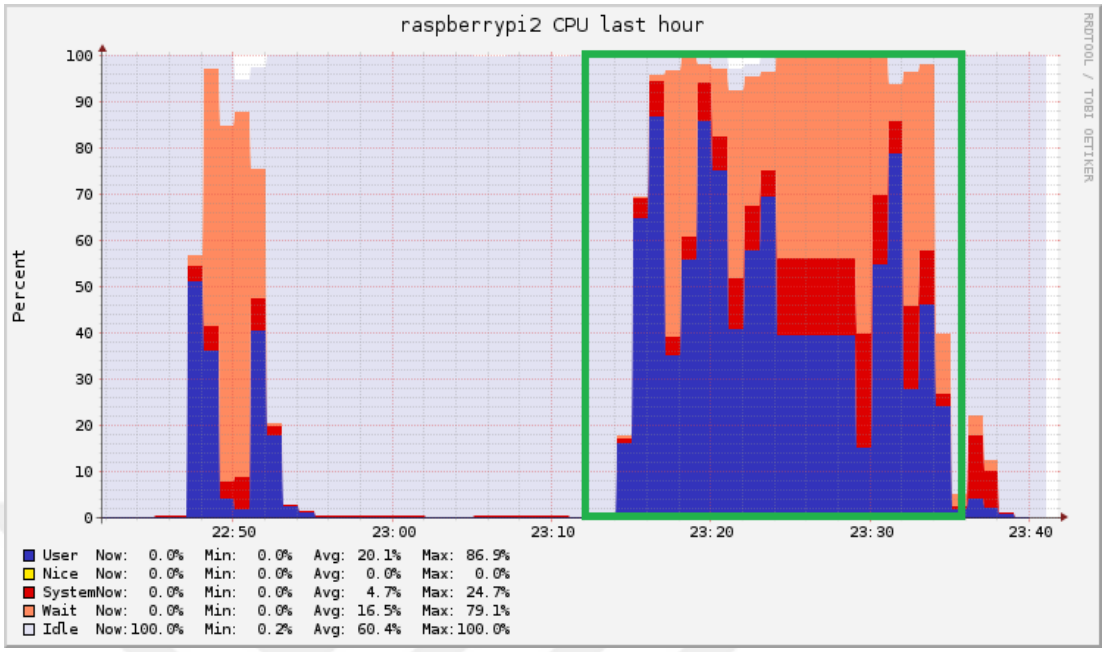
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	9484020 MB-seconds, 7974 vcore-seconds

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	14-May-2015 23:18:54	raspberrypi2:8042	logs

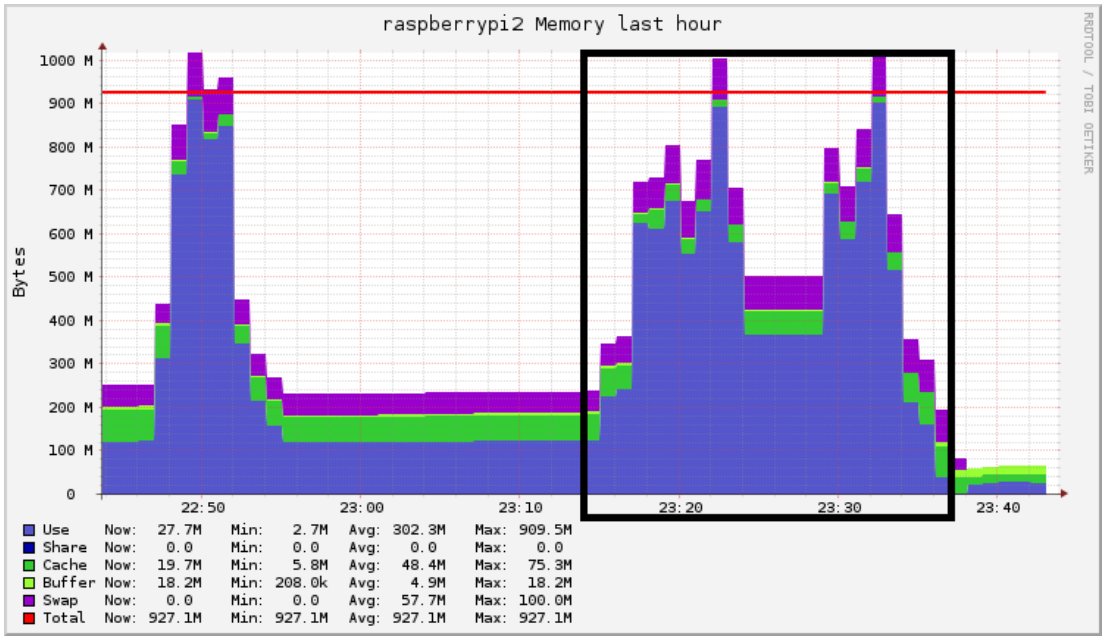
Şekil 65.Uygulamaya ait bilgiler.

Şekil 66' da ki grafik uygulamanın çalışma zamanı içerisindeki işlemci kullanımını göstermektedir. Uygulama çalıştığı süre boyunca işlemciyi %100 verimde kullandığı görülmektedir.



Şekil 66.Test sırasında sistemin CPU kullanımı bilgisi.

Şekil 67' deki grafik uygulamanın çalışma zamanı içerisindeki memory kullanımını göstermektedir.



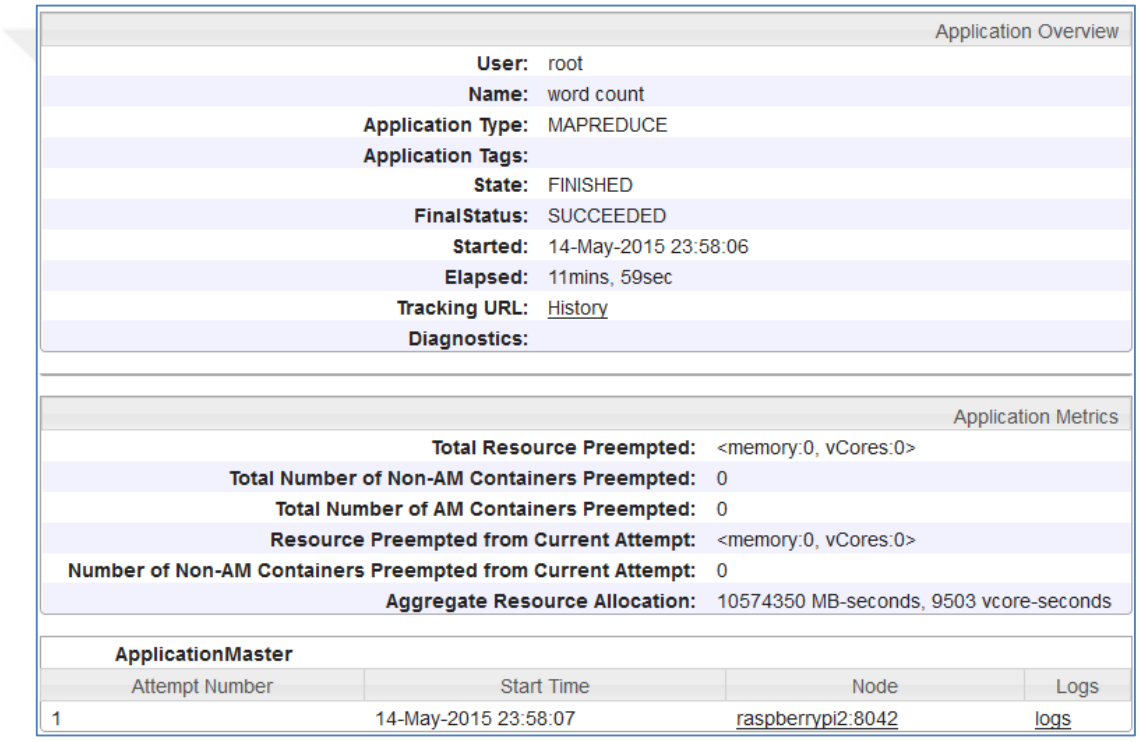
Şekil 67.Test sırasında sistemin Memory kullanımı bilgisi.

Ganglia üzerinden çok daha fazla sayıda detay rapor alınabilmektedir bunlar network, disk, swap gibi raporlardır. Bu test işleminde CPU kaynaklı inceleme yapılmaktadır. Ganglia uygulaması dışında farklı komut veya uygulamalar ile sistemin incelenmesi mümkündür şekil 68' de "sar" komutu ile işlemci kullanımı ve kullanıcıların bu işlemleri kullanma oranları görünmektedir.

Şekil 68.Test sırasında sistemin Sar komutu ile CPU kullanımı bilgisi.

5.4 Test işleminin 2 Düğüm Açık Diğerleri Kapalı Şekilde Yapılması

Wordcount uygulaması iki düğüm üzerinde çalıştırıldığında toplam 11 dk. 59 sn.' lik bir sürede tamamlanmaktadır. Bu test işleminde uygulama cihazlar arasında paralel olarak çalışmaktadır. Aşağıda grafikler ile aktif sunuculardaki işlemci kullanımları tek tek gösterilmektedir. Uygulama çalıştığı süre boyunca sistem üzerindeki kaynak kullanım bilgileri Ganglia ile toplanmaktadır. Kaynak kullanımına ait detaylı raporlar aşağıdaki şekillerde gösterilmektedir.



Application Overview

User:	root
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	14-May-2015 23:58:06
Elapsed:	11mins, 59sec
Tracking URL:	History
Diagnostics:	

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	10574350 MB-seconds, 9503 vcore-seconds

ApplicationMaster

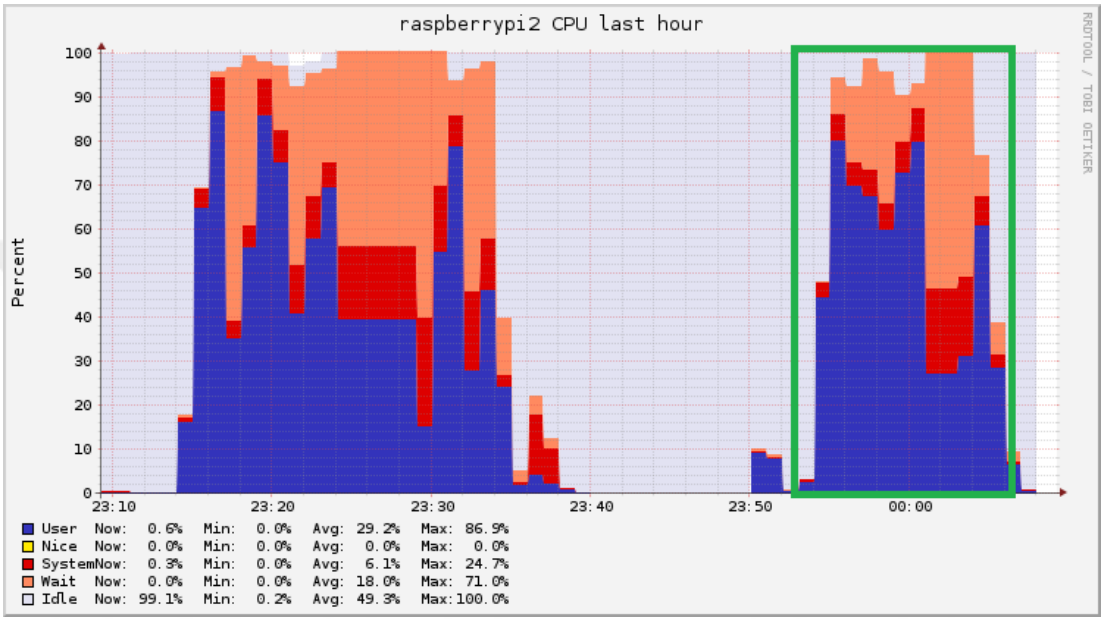
Attempt Number	Start Time	Node	Logs
1	14-May-2015 23:58:07	raspberrypi2:8042	logs

Şekil 69.Uygulamaya ait bilgiler.

Uygulama küme (Cluster) mimarisinde çalıştırdığında amaç küme içerisindeki sistemlerdeki kaynakların tamamının verimli şekilde kullanılmasıdır. Uygulamanın çalışma süresi içerisindeki durumu görebilmek için aşağıda sırası ile her iki sistemdeki kaynak kullanımları ayrı ayrı gösterilmektedir. Şekil 70' de kümede bulunan 1.sisteme ait işlemci kullanımı bilgisi görünmektedir. 1.sistemde uygulama çalışma zamanında işlemci kullanımı %100 olarak görünmektedir.

1.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;

Şekil 70’ de 1.sistem için uygulama çalışma sırasındaki işlemci kullanımı bilgisi görünmektedir. Maksimum seviyede işlemci kullanımı sağlanmıştır.

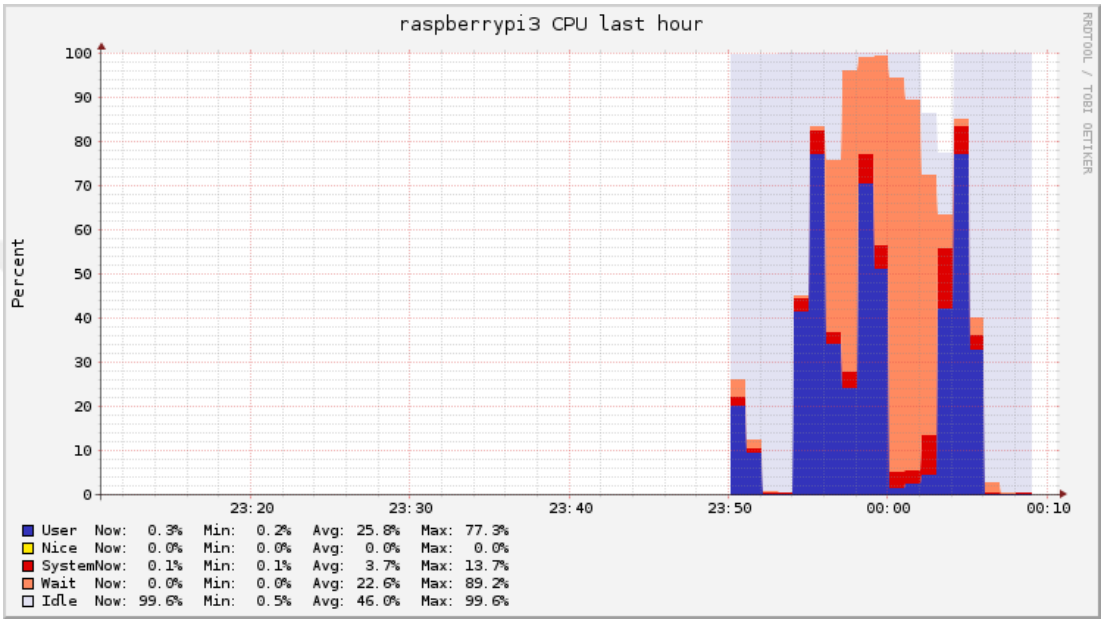


Şekil 70.Test sırasında sistemin CPU kullanımı bilgisi.

Bu grafik üzerinden uygulama çalışması sırasında işlemcinin %100 kullanımının alt detayları da görünmektedir. Bu grafikte işlemcinin ne kadarlık kısmını kullanıcı (user), ne kadarlık kısmını işletim sisteminin (system) ve ne kadarlık kısmının bekleme (wait) durumunda olduğu ayrıca görünmektedir.

2.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;

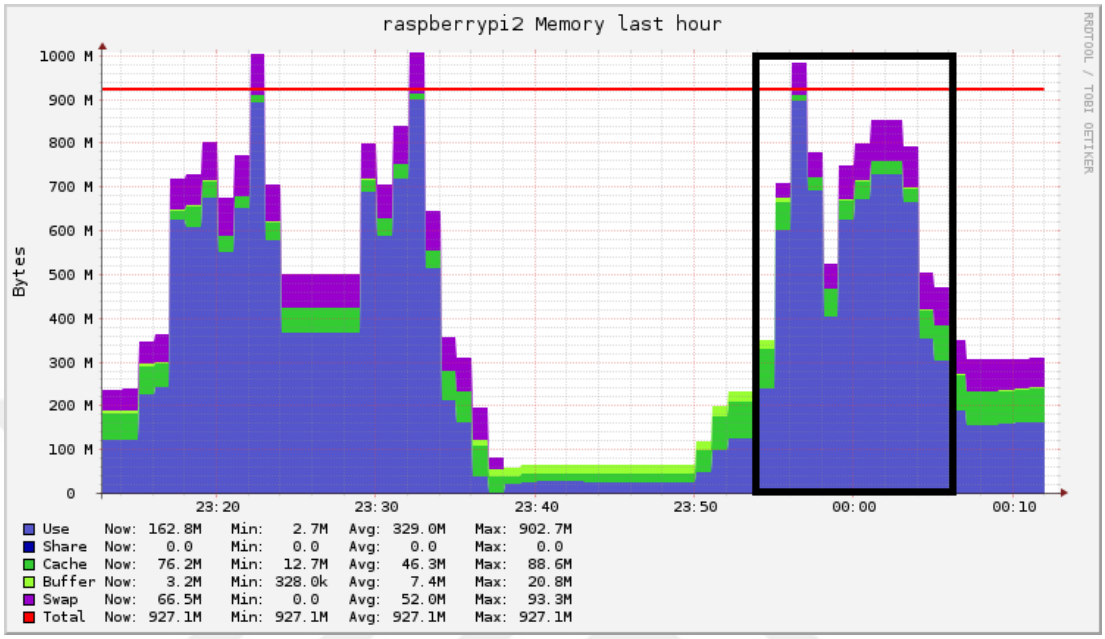
Şekil 71' de küme sistemindeki 2.sistem için uygulama çalışma sırasındaki işlemci kullanımı bilgisi görünmektedir. Maksimum seviyede işlemci kullanımı sağlanmıştır.



Şekil 71.Test sırasında sistemin CPU kullanımı bilgisi.

Uygulama çalıştığı süre boyunca küme içerisinde her iki sistem için işlemcileri %100 verimde kullandığı görülmektedir. İşlemcinin %100 kullanımı yapılan tüm testlerde aynı çıkmaktadır bu sayede sistemin istikrarlı olduğu görünmektedir.

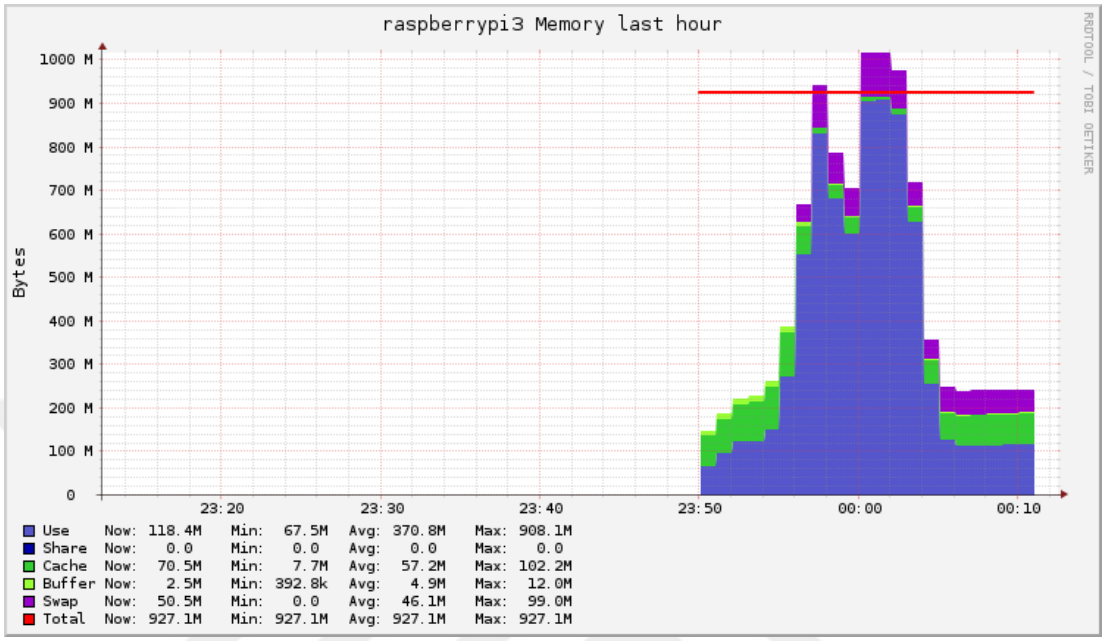
1.Düğüm İçin Uygulama Çalışma Süresinde Bellek (Memory) Kullanımı;



Şekil 72.Test sırasında sistemin Bellek (Memory) kullanımı bilgisi.

Uygulama çalışma sırasında çalıştığı düşüm üzerinde kullandığı memory miktarını bu grafikte görülmektedir. Bu tez kapsamında test için kullanılan uygulama daha çok işlemci gücünü kullandığı için memory kullanımı yüksek seviyelerde seyretmemektedir.

2.Düğüm İçin Uygulama Çalışma Süresinde Bellek (Memory) Kullanımı;

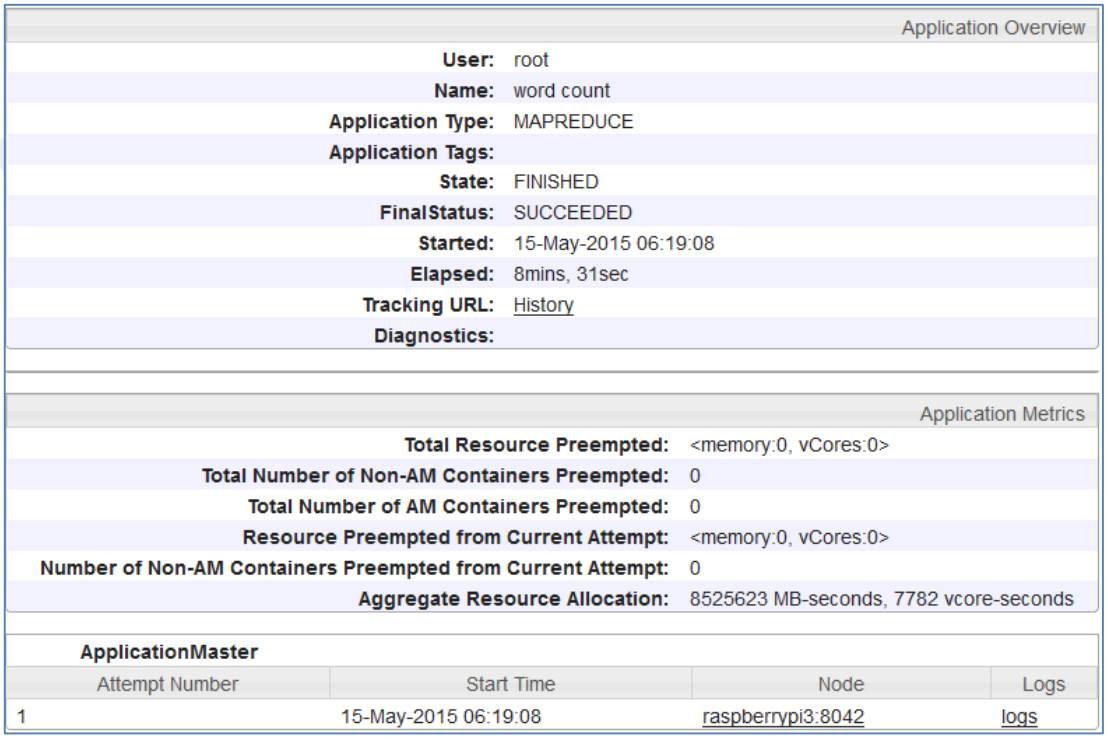


Şekil 73.Test sırasında sistemin Bellek (Memory) kullanımı bilgisi.

Uygulama çalışma sırasında diğer düğümler içinde memory kullanımı aynı seviyelerde seyrettiği görünmektedir.

5.5 Test işleminin 3 Düğüm Açık Diğerleri Kapalı Şekilde Yapılması

Wordcount uygulaması üç adet düğüm üzerinde çalıştırıldığında toplam 8 dk. 31 sn.' lik bir sürede tamamlanmaktadır. Uygulama çalışma süresinde sistem üzerindeki kaynaklara ait kullanım bilgileri Ganglia ile toplanmaktadır. Kaynak kullanımına ait raporlar aşağıda gösterilmektedir.



Application Overview

User:	root
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	15-May-2015 06:19:08
Elapsed:	8mins, 31sec
Tracking URL:	History
Diagnostics:	

Application Metrics

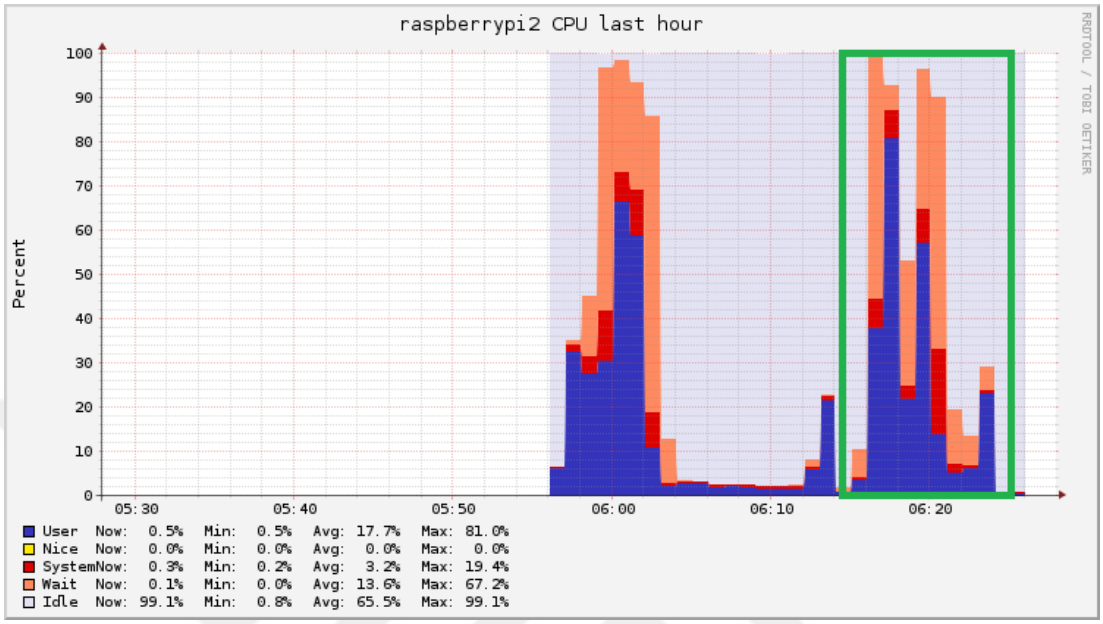
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	8525623 MB-seconds, 7782 vcore-seconds

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	15-May-2015 06:19:08	raspberrypi3:8042	logs

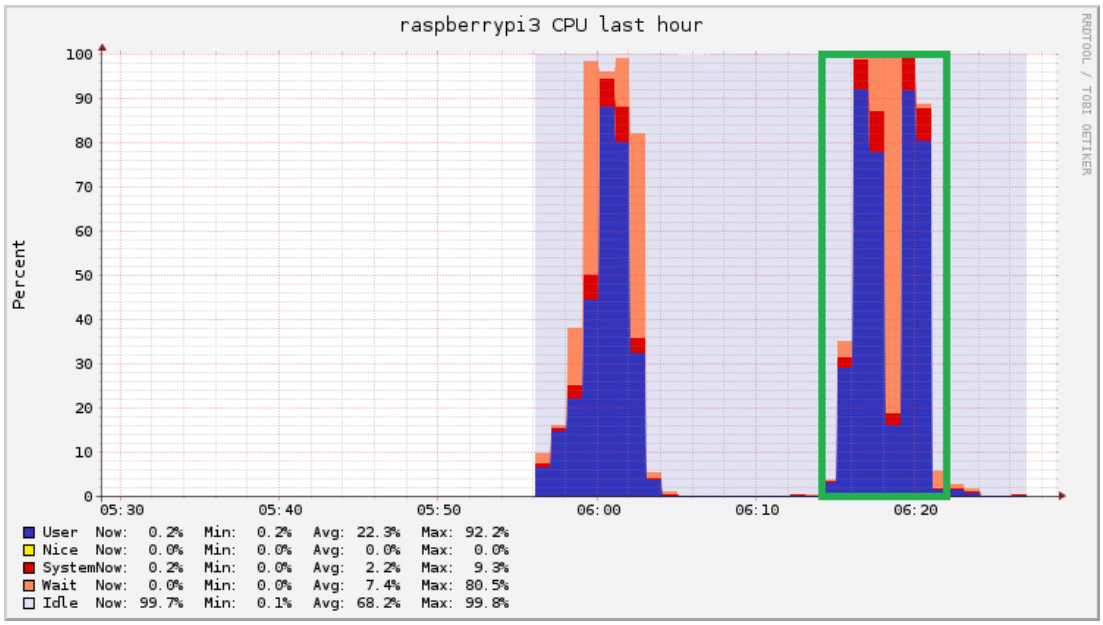
Şekil 74.Uygulamaya ait bilgiler.

1.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



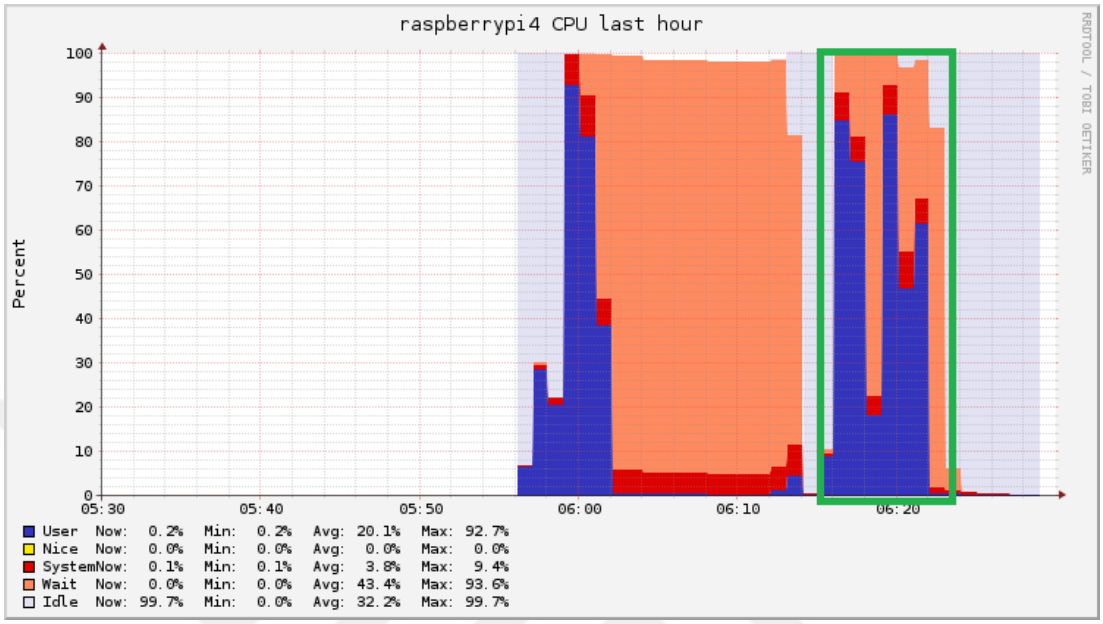
Şekil 75.Test sırasında sistemin CPU kullanımı bilgisi.

2.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



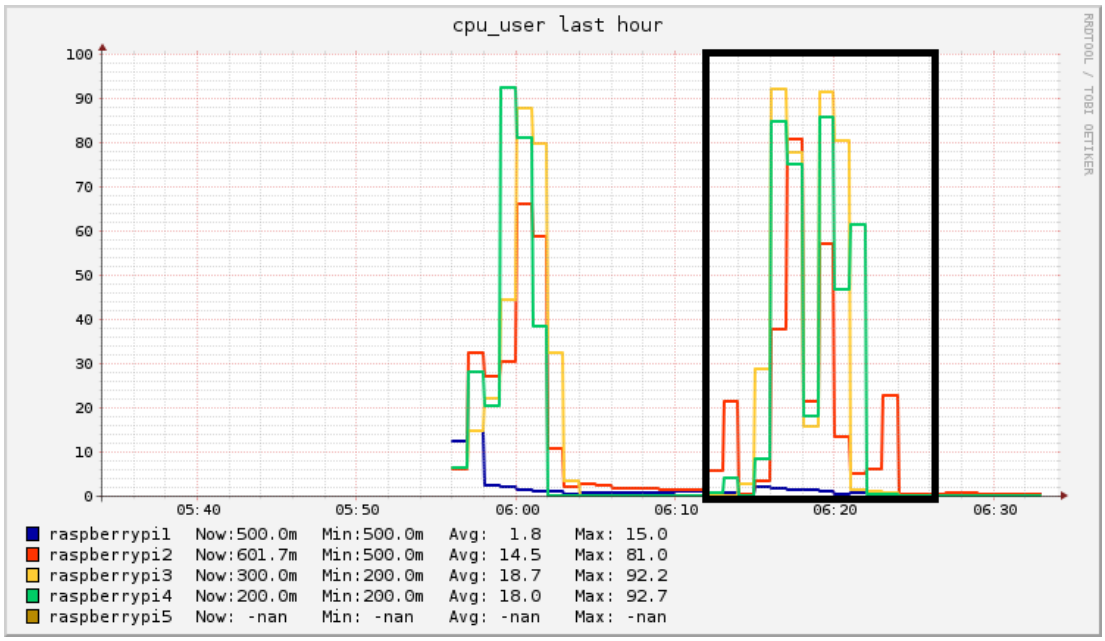
Şekil 76.Test sırasında sistemin CPU kullanımı bilgisi.

3.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



Şekil 77.Test sırasında sistemin CPU kullanımı bilgisi.

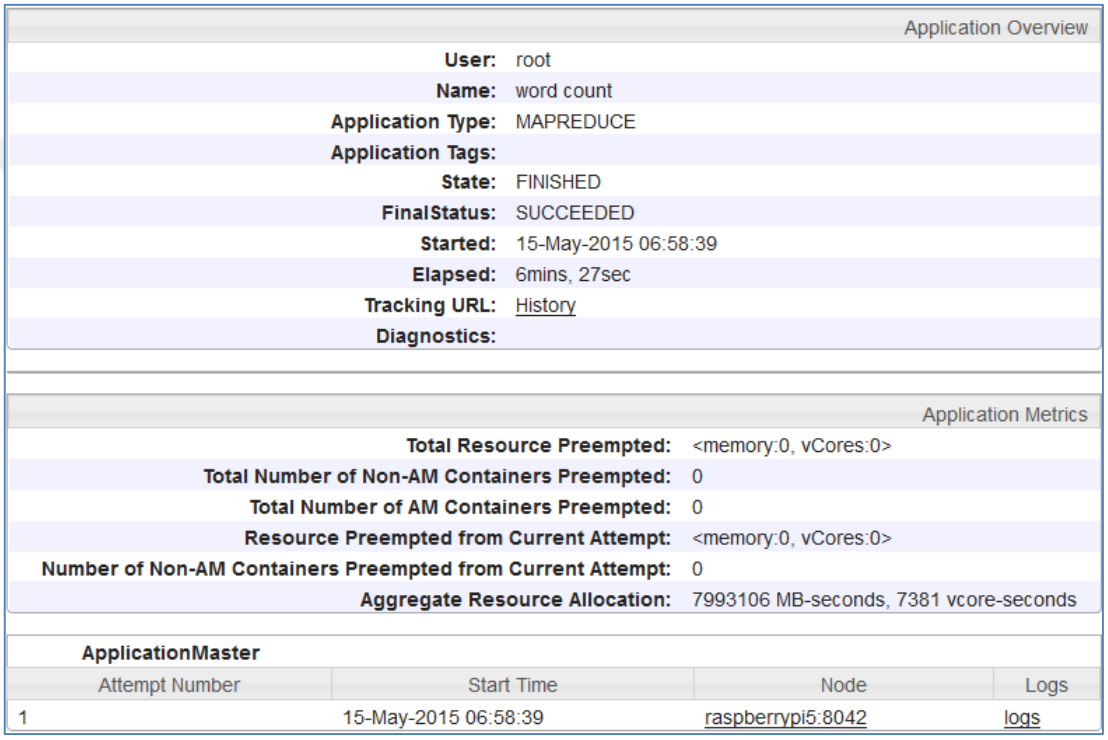
Uygulama çalıştığı süre boyunca işlemcileri %100 verimde kullandığı görülmektedir.



Şekil 78.Üç sunucu için CPU kullanımı karşılaştırması.

5.6 Test işleminin 4 Düğüm Açık Şekilde Yapılması

Wordcount uygulaması dört düğüm üzerinde çalıştırıldığında toplam 6 dk. 27 sn.' lik bir sürede tamamlanmaktadır. Uygulama çalışma süresinde sistem üzerindeki kaynaklara ait kullanım bilgileri Ganglia ile toplanmaktadır. Kaynak kullanımına ait raporlar aşağıda gösterilmektedir.



Application Overview

User:	root
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	15-May-2015 06:58:39
Elapsed:	6mins, 27sec
Tracking URL:	History
Diagnostics:	

Application Metrics

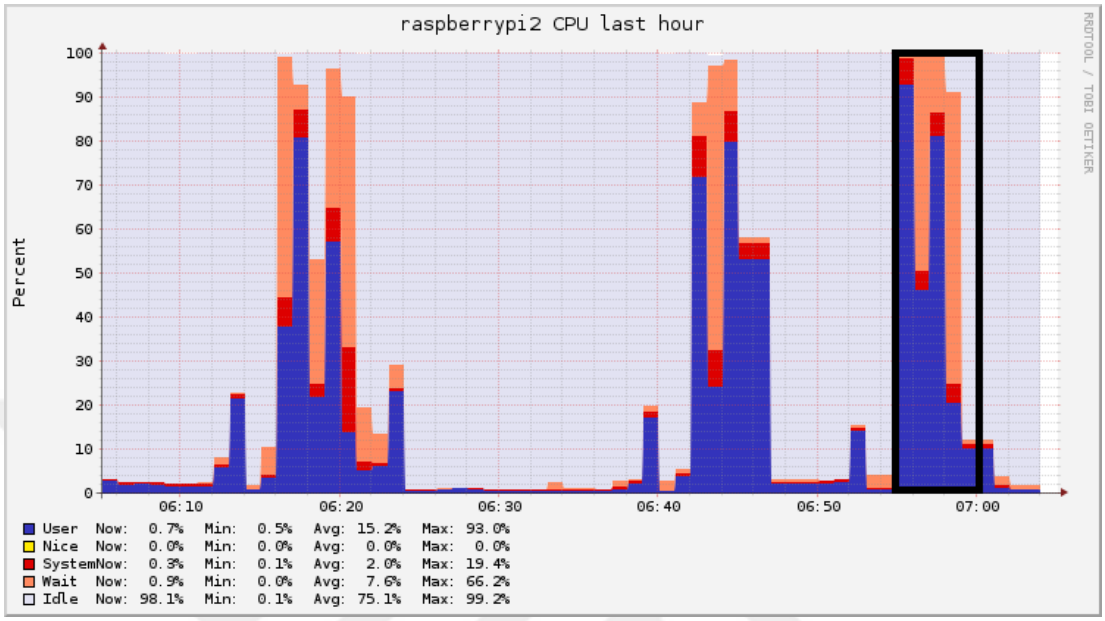
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	7993106 MB-seconds, 7381 vcore-seconds

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	15-May-2015 06:58:39	raspberrypi5:8042	logs

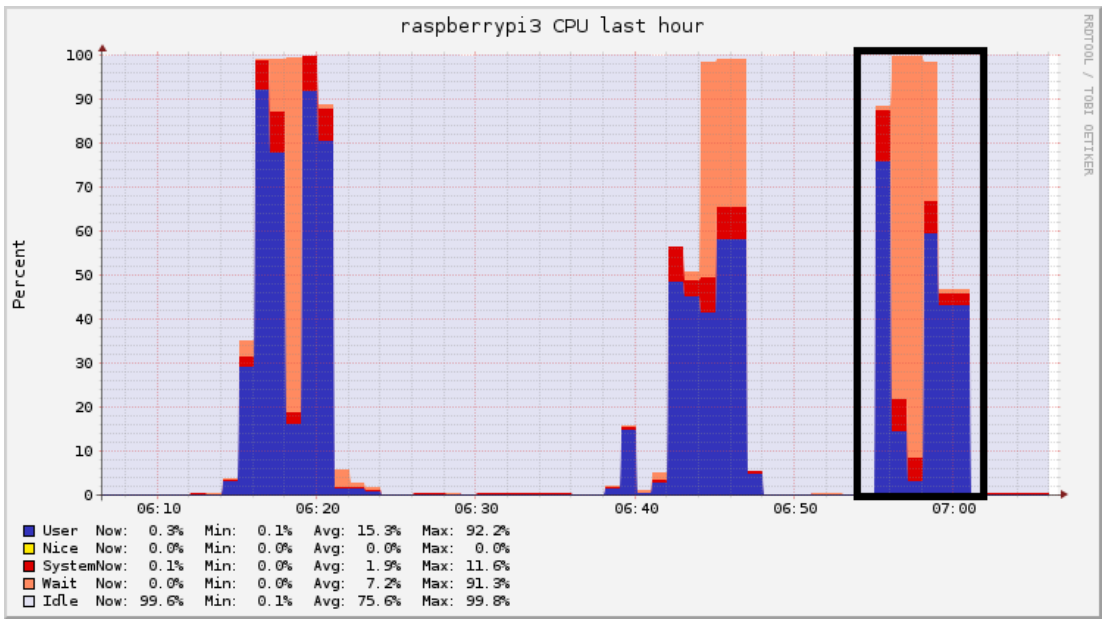
Şekil 79.Uygulamaya ait bilgiler.

1.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



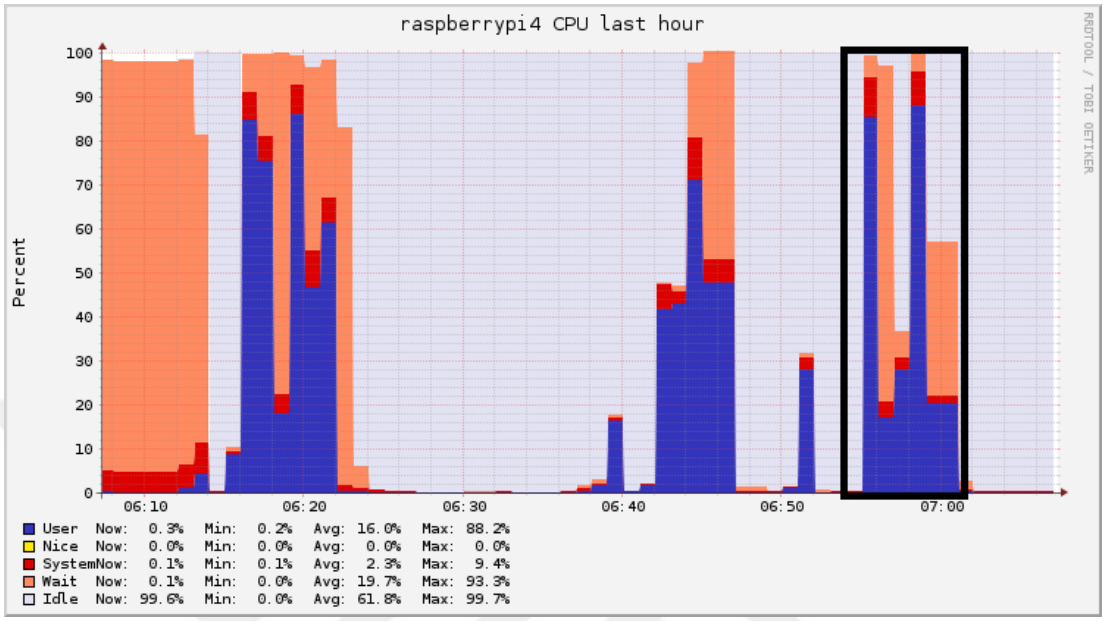
Şekil 80.Test sırasında sistemin CPU kullanımı bilgisi.

2.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



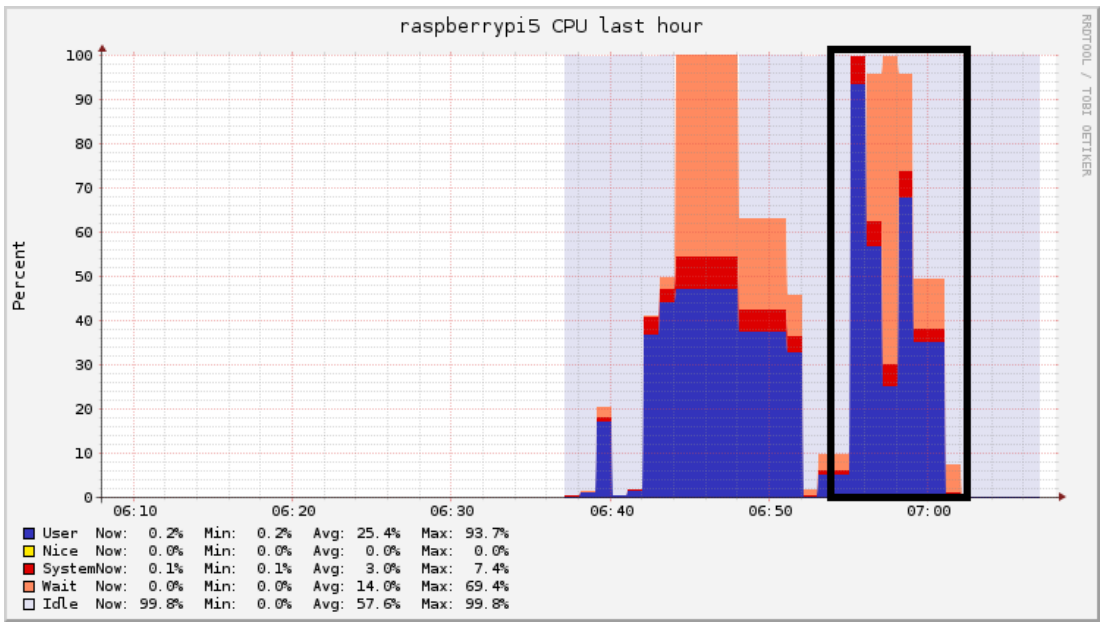
Şekil 81.Test sırasında sistemin CPU kullanımı bilgisi.

3.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



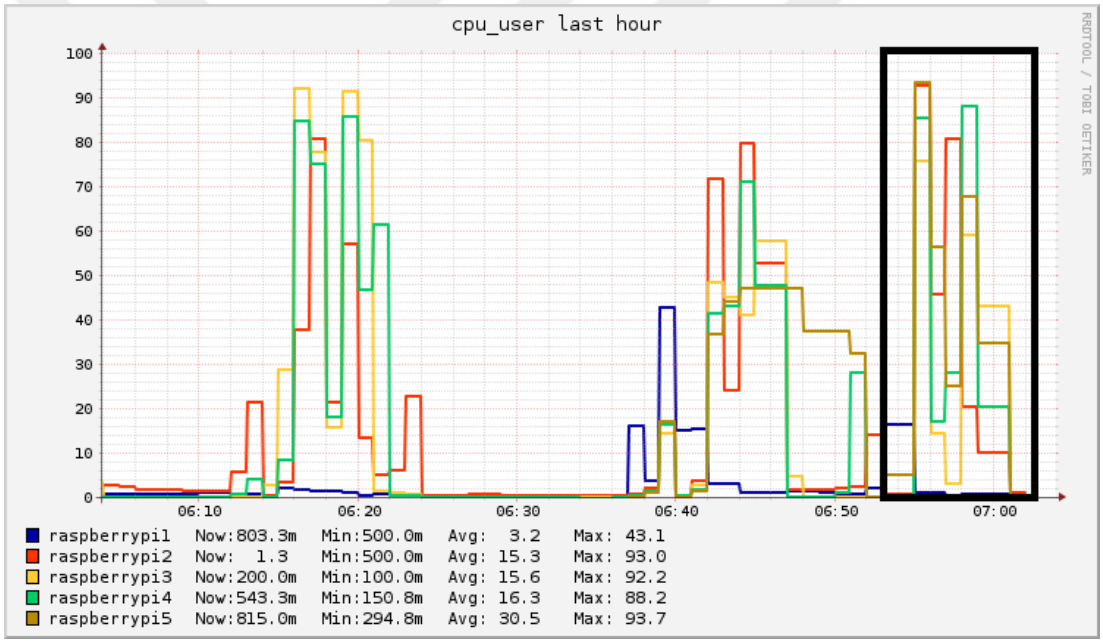
Şekil 82.Test sırasında sistemin CPU kullanımı bilgisi.

4.Düğüm İçin Uygulama Çalışma Süresinde İşlemci Kullanımı;



Şekil 83.Test sırasında sistemin CPU kullanımı bilgisi.

Şekil 84’ de dört tane sunucunun işlemci performans karşılaştırması görülmektedir. “raspberrypi1” isimli sistem ana (master) sistem olduğu için kaynak kullanımı diğer düğümlere göre düşüktür. Diğer node’ larda ise kaynak kullanımları yüksek seviyede görünmektedir. Yapılan tüm test işlemlerinde ARM mimarili sistem istikrarlı bir performans sergilemiş ve kaynak kullanımı yüksek seviyelere ulaşmaktadır. INTEL mimarili karşılaştırma sisteminde ise testler sırasında bazı aşamalarda yüksek kaynak kullanımı sırasında sistemlerin kilitlendiği ve yeniden başlatma işlemi yapılmadan sistemler çalışır duruma gelmediği gözlenmiştir bu sebeple INTEL mimarili sistemler için bu test çalışmaları aşamasında istikrarlı olmayan bir durum görünmektedir.



Şekil 84.Dört sunucu için CPU kullanımı karşılaştırması.

5.7 Sistemin Fiziksel Bilgisayarlar İle Test Edilmesi

5.7.1 Fiziksel INTEL Mimarili Sistemin Kurulumu

INTEL mimarili sistemi ARM mimarili sistem ile aynı olacak şekilde öncelikle işletim sistemi kurulumları yapılmaktadır. İşletim sistemi kurulumlarından sonra sistemler ARM mimarili sistemde olduğu gibi bir "network switch" üzerinde çalışacak şekilde gerekli ayarlar yapılmaktadır. ARM mimarili sistem üzerine kurulan Hadoop yazılımı dizini kopyalanıp INTEL mimarili sistem üzerine taşınmaktadır. Ağ ayarları işletim sistemlerinin "hostname" bilgileri birebir aynı olması sebebi ile uygulama mevcut şekilde çalıştırılmaktadır. INTEL mimarili beş sunucunun da kurulumu tamamlandıktan sonra ARM mimarili sistemde yapılan uygulama testi INTEL mimarili sistem üzerinde de çalıştırılmaktadır.

5.7.2 Mimari Karşılaştırma

Bu tez çalışmasında test edilen ARM mimarili yapı üzerindeki yazılım tüm yazılım ve sistem ayarları aynı kalacak şekilde INTEL mimarili geleneksel sistemler üzerinde de test edilmektedir. Bu test için oluşturulan mimariye ait karşılaştırmalı tablo 10' da ki gibidir.

	ARM Mimari	**INTEL Mimari**
İşlemci	ARM Cortex-A7 CPU (900MHz) Quad Core	Intel® Core™2 Duo Processor T7250 (2M Cache, 2.00 GHz, 800 MHz FSB)
Memory	1GB SD RAM	4 GB DDR RAM
Disk	8 GB Class 10 MicroSD	500 GB SATA
Fiyat	120 TL	1.000 TL
4 Düğüm İle Test Tamamlanma Süresi	6:27 dk.	4:57 dk.

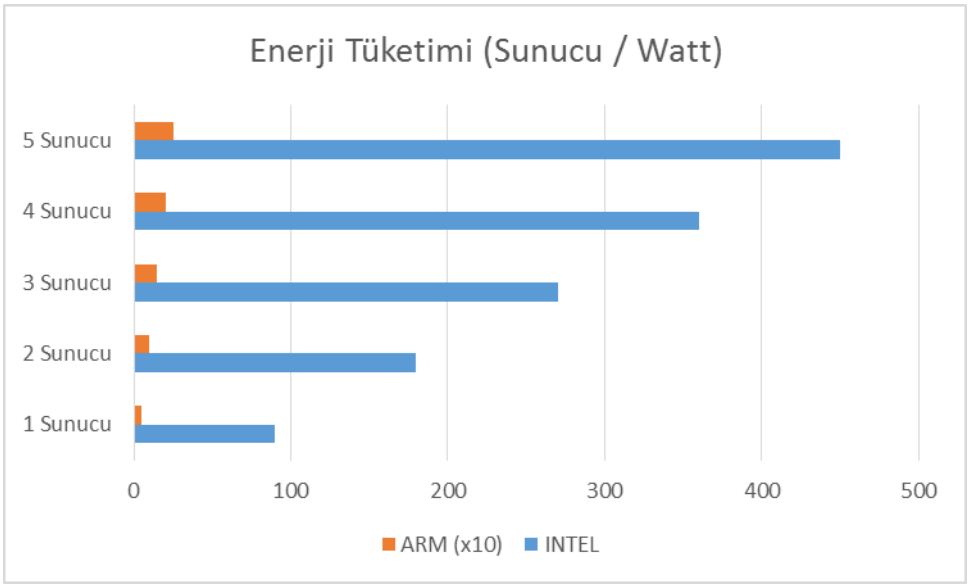
Tablo 10.ARM ile INTEL Mimari Karşılaştırma.

Tablo 10' da ki karşılaştırmada ARM mimarili test sistemi ile INTEL mimarili sistem arasındaki fiyat farkı yaklaşık 8,5 kat şeklindedir buna karşılık uygulamadaki zaman

kazanımı ise 1,5 dk. şeklindedir. Bu karşılaştırmaya göre ARM mimarili sistemin fiyat performans yönünden INTEL mimarili sisteme göre avantajlı olduğu görünmektedir.

5.7.3 Enerji Tüketimi Karşılaştırması

ARM mimarili sistemde Raspberrypi 2 cihazları kullanılmaktadır bu cihazların her biri 0,5 watt olarak çok düşük miktarda enerji tüketmektedir. Toplamda 2,5 watt enerji tüketimi bulunmaktadır. INTEL mimarili sistemde her bir sunucu 90 watt toplamda 450 watt enerji tüketmektedir. Enerji tüketimi karşılaştırması şekil 85' de gösterilmektedir. Ayrıca INTEL mimarili sistemlerde ısınma sorunları olması sebebi ile sistemlerin soğutulması gerekmektedir bu işlemler içinde ayrıca enerji tüketimi ihtiyacı ortaya çıkmaktadır. ARM mimarili sistemlerde ise ısınma sorunu tespit edilmemiştir. Yapılan testlerde uzun süreli çalışmalar yapılmış, sistemler uzun süreler boyunca açık ve yük altında çalıştırılmasına rağmen ısınma sorunu tespit edilmemiştir. Ayrıca sistem hiç kapatılmadan 48 saat boyunca yük altında çalıştırılmıştır INTEL mimarili sistemde karşılaşılan ısınma sorunu ARM mimarili sistemlerde tespit edilmemiştir.

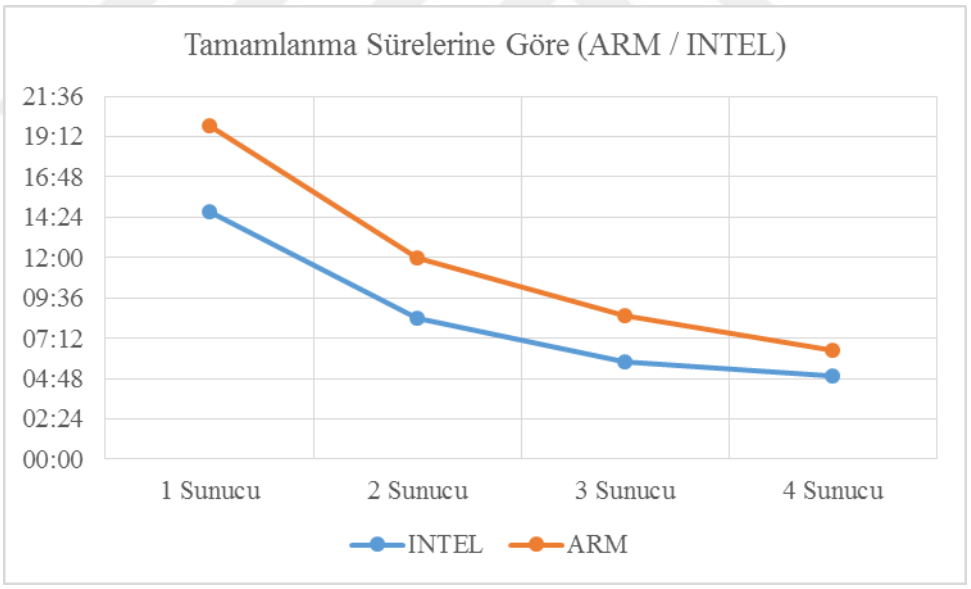


Şekil 85:Enerji tüketimi karşılaştırması.

5.7.4 Test Sonuçları

INTEL mimarili sistem ile yapılan test işleminde 4 sunucu ile uygulamanın tamamlanma süresi 4:57 dk. sürmektedir. Bu test işlemleri birden fazla kez denenmiştir işlemcilerin ısınması sebebi ile oluşan kilitlenmeler sebebi ile bazı testlerin süreleri daha fazla ve farklı çıkmaktadır bu sebeple sistem istikrarlı olarak çalışmamaktadır.

ARM ve INTEL mimarili sistemler üzerinde yapılan test işlemlerine ait karşılaştırma bilgileri şekil 86' da gösterilmektedir. Şekil 86' da ki grafiği incelediğimizde ARM mimarili sistemlerdeki işlemci sayısı arttıkça INTEL mimarili sistemdeki performans sürelerine ulaşılmaktadır. Bu durum ARM mimarili sistemlerin paralellik özelliklerini daha verimli kullanabildiğini göstermektedir.



Şekil 86.Uygulamanın sistemler üzerindeki tamamlanma süreleri karşılaştırması.

Application Overview	
User:	root
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
State:	FINISHED
FinalStatus:	SUCCEEDED
Started:	22-Jun-2015 02:16:52
Elapsed:	4mins, 57sec
Tracking URL:	History
Diagnostics:	

Application Metrics	
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	1885968 MB-seconds, 1511 vcore-seconds

Şekil 87:INTEL mimari ile yapılan 4 düğümlü işlem sonucu.

Test işlemleri sırasında performans ölçümleri Ganglia ile yapılmaktadır. INTEL mimarisi sistemin çalışması sırasında Ganglia üzerinden sunucuların %100 kullanımda olduğunu de görünmektedir.

Şekil 88: INTEL mimarili sisteme ait kaynak kullanımları.

INTEL mimarili sistemin çalışması sırasında yüksek işlemci kullanımı sırasında işlemcilerin ısınması sebebi ile işlemci soğutma fanları devamlı olarak çalıştığı görülmüştür ayrıca uzun süreli yüksek işlemci kullanımında sistemin kilitlendiği tespit edilmiştir. ARM mimarili sistemde ise ısınma sorunu olmadığı için uzun süreli yüksek işlemci kullanımlarında herhangi bir performans sorunu ile karşılaşılmamıştır.

5.8 Performans Hesaplamaları

Sunucu sayısı arttıkça işlemlerin daha hızlı tamamlandığı görülmektedir. Bunun matematiksel hesaplanması için denklem (1)' [27] deki "Amdahl's" kanununa göre "Speedup" hızlanma oranı formülü kullanılmaktadır.

$$Hızlanma\ Oranı = \frac{Seri\ Çalışma\ Süresi}{Paralel\ Çalışma\ Süresi} \quad (1)$$

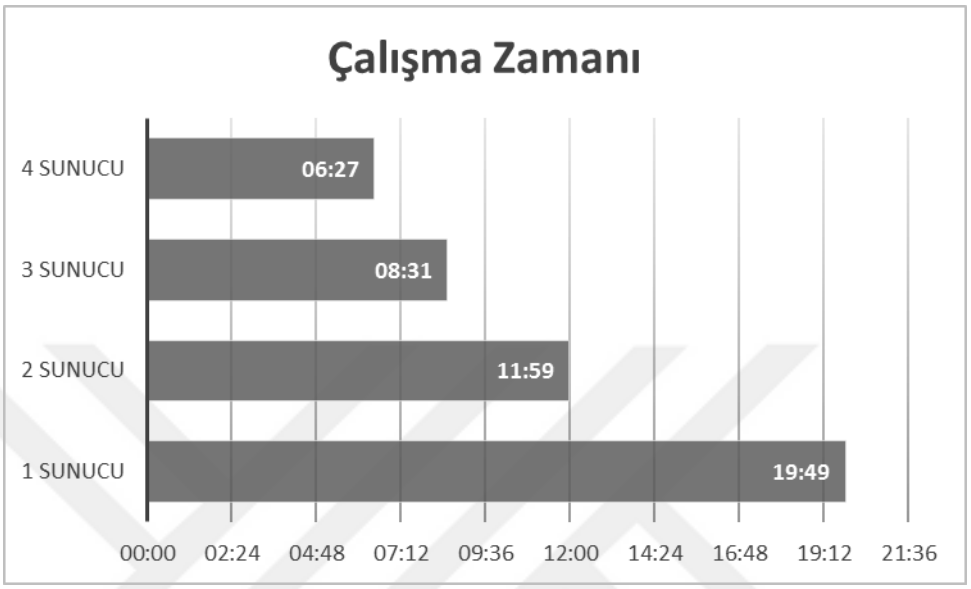
Tablo 13' de denklem (1)' ile hesaplanmış hızlanma ve verimlilik sonuçları gösterilmektedir.

CPU	Hızlanma (SpeedUp)	Verimlilik (Efficiency)
1	1	100,00%
2	1,65	82,50%
3	2,33	77,67%
4	3,07	76,75%

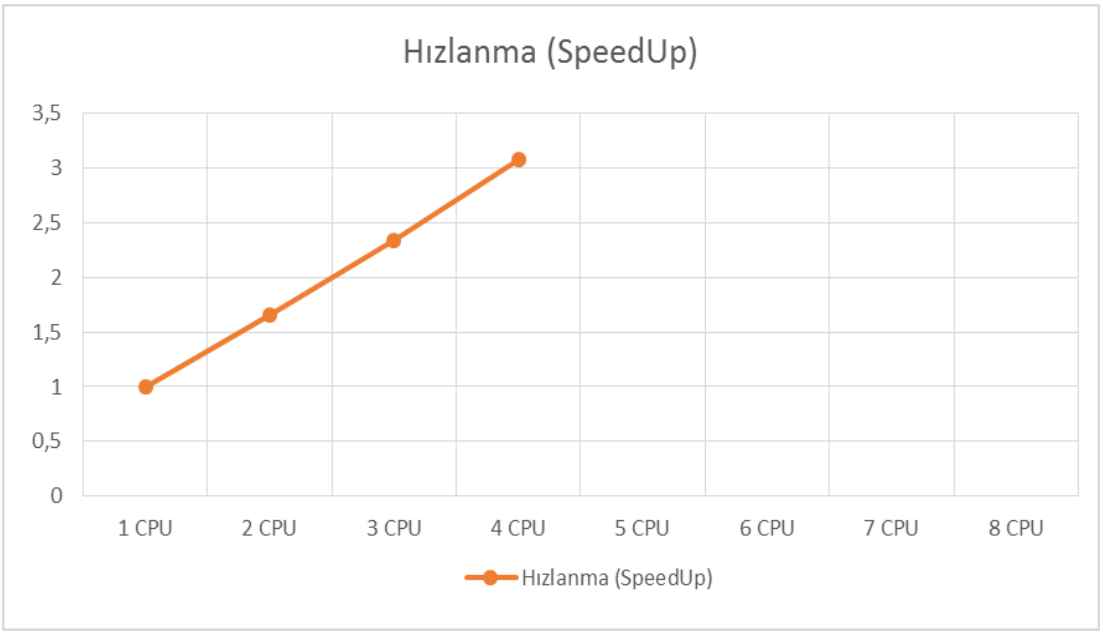
Tablo 11.Sunucu sayısına göre hızlanma değerleri.

Sunucu sayısına göre performans grafiği şekil 89'de gösterilmektedir. Sunucu sayısı arttıkça işlemlerin daha hızlı tamamlandığı görülmektedir. Tek sunucu ile iki sunucu arasında işlemlerin tamamlanma oranına göre %31,8 oranında bir performans artışı görünmektedir. Sunucu sayısı arttıkça performans iyileşiyor fakat performans

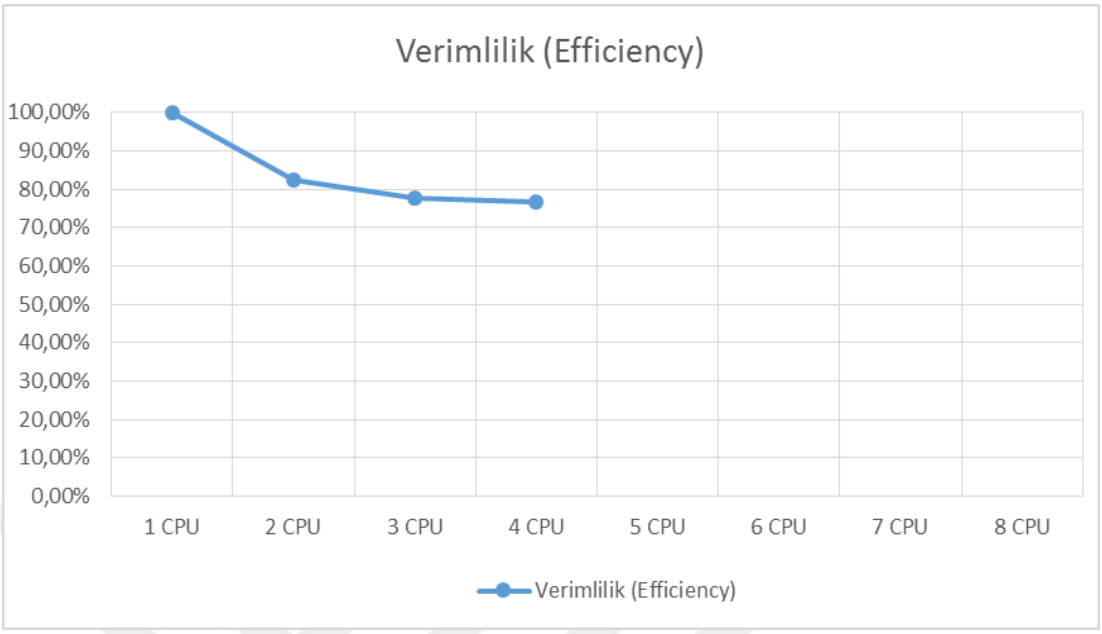
oranında artış oranı yanı olmamaktadır. Performans oranı sunucu sayısı arttıkça kademe kademe düşmektedir.



Şekil 89.Sunucu sayısına göre performans grafiği.



Şekil 90.Hızlanma performans grafiği.



Şekil 91.Verimlilik (Efficiency) grafiği.

Testlerin tekrar tekrar çalıştırılarak yanı performans sonuçlarının oluştuğu tespit edilmiştir. Örneğin dört sununucu ile çıkan 6:27' lik tamamlanma zamanı yapılan tüm testlerde yaklaşık değerlerde çıkmaktadır. Bu durum sistemin stabil olarak çalıştığını ve sürekli olarak aynı performans değerlerini verdiğini göstermektedir.

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında Apache Hadoop ve bileşenleri olan Apache Pig, Apache Hive ve günümüzde çok sık kullanılmaya başlanılan MapReduce uygulamaları denenmiş ve çalıştırılmıştır. Bu çalışmada Apache Hadoop ve bileşenlerinin ARM işlemcili mikro bilgisayarlar üzerinde de çalışabileceği gösterilmiştir. Çalışma sırasında gerçek zamanlı testler oda sıcaklığında çalışmalar uzun süre sistemlerin yük altında çalıştırılması işlemleri gerçekleştirilmiştir. Bu sayede mikro bilgisayarların ticari olarak büyük bilgisayar sistemleri gibi sistem odalarına dâhil olabileceği büyük verilerin işlenmesi gibi büyük ölçekli işlemlerde kurumsal olarak kullanılabileceği gösterilmiştir. Çalışma kapsamında oluşturulan mikro süper bilgisayarın kapasitesi 5 adet sunucu ile sınırlı tutulmasına rağmen sistemin genişletilmesinin çok basit ve hızlı bir şekilde olabileceği açıklanmıştır. İstenildiği takdirde 1000+ (sınırsız sayıda olabilir) mikro bilgisayarın bu sistem ile aynı anda kullanılabileceği anlatılmaktadır.

Bu çalışma sırasında Hadoop sisteminin yeni teknoloji olması ve devamlı bir gelişim içinde olması sebebi ile ayrıca ülkemizde bu konularda çok fazla çalışma olmasında dolayı daha fazla detaylı bilgiye ulaşmada sorunlar yaşanmıştır. Bu sistem üzerinde daha fazla çalışmalar ve yeni sürümler ile birlikte bu çalışma sırasında elde edilen performans değerlerinin çok daha fazlasına ulaşılabileceği görülmektedir.

Mikro bilgisayarların ülkemizde çok az sayıda kullanılıyor olması ve bu çalışma içerisindeki anlatımı kapsamında çalışmanın ülkemizde bu alanda yapılmış az sayıda çalışmalardan biri olması nedeniyle başka birçok çalışma için bir çıkış noktası ve başvuru kaynağı olabileceği düşünülmektedir.

Tez çalışmasının ana katkıları aşağıdaki gibi maddeler halinde açıklanmaktadır;

a) Hadoop MapReduce programlama modelinin mimarisi açıklanmıştır.

b) Hadoop dağıtık dosya sisteminin yapısı, bileşenleri ve MapReduce yazılımı ile olan ilişkileri açıklanmıştır.

c) Apache Pig ve Apache Hive uygulamaları hakkındaki bilgiler açıklanmış ve bu yazılımlar mikro bilgisayarlar üzerinde çalıştırılmıştır.

d) Hadoop sisteminin ARM mimarili mikro bilgisayarlar üzerinde çalışabileceği açıklanmaktadır.

e) Kapalı devre sistemi ile oluşturulan sistem mimarisinin sorunsuz ve normla bilgisayar sistemleri gibi çalışabileceği test edilmiştir.

f) Mikro bilgisayarların kurumsal olarak veri işleme vb. günümüz bilgisayar sistemlerinin gerçekleştirdiği görevleri yapabileceği açıklanmıştır.

Tasarlanan bu sistemin yeni sürümler ile ilerletilmesi ve daha performanslı kullanılması önerilmektedir. Bu tez çalışmasına başlandığında ve tez çalışması içerisinde kullanılan Hadoop sürümü 2.6.0 şeklindedir. Bu çalışma tamamlanmadan yeni bir sürüm olan 2.7.0 sürümü de yayınlanmıştır. Bu hızlı ilerleyiş içerisinde tasarlanan bu sisteminde daha hızlı şekilde kullanım alanları içerisinde girmesi beklenmektedir.

KAYNAKLAR

1. Wenhui Lin and Jun Liu. Performance Analysis of MapReduce Program in Heterogeneous Cloud Computing. Beijing University of Posts and Telecommunications Technology Research Institute, Aisino Corporation, Beijing 100195, China, 2-8, 2013
2. Ali Anwar, Krish K. R., and Ali R. Butt. On the Use of Microservers in Supporting Hadoop Applications. In Proceedings of the IEEE International Conference on Cluster Computing (Cluster), Madrid, Spain, pages 66-74, September 2014
3. Franks, Shaun and Yerby, Johnathan, Creating a Low-cost Supercomputer with Raspberry Pi 8-9, SAIS 2014
4. Dumitrel Loghin, Bogdan Marius Tudor, Hao Zhang, Beng Chin Ooi, Yong Meng Teo, A Performance Study of Big Data on Small Nodes, Department of Computer Science National University of Singapore, 2-9, 2015
5. http://en.wikipedia.org/wiki/Raspberry_Pi (21.05.2015)
6. https://www.raspberrypi.org/forums/viewtopic.php?f=89&t=102318 (25.05.2015)
7. https://anahtar.sanayi.gov.tr/tr/news/buyuk-veri-ve-pazarlama/1878 (03.05.2015)
8. https://anahtar.sanayi.gov.tr/tr/news/buyuk-veri-ve-pazarlama/1878 (21.05.2015)
9. https://anahtar.sanayi.gov.tr/tr/news/buyuk-veri-ve-pazarlama/1878 (21.05.2015)
10. Dali Ismail,Steven Harris, "Performance Comparison of Big Data Analysis using Hadoop in Physical and Virtual Servers", Washington University, Department of Computer Science & Engineering, Page 3, 2013
11. http://www.enacore.com.tr/tr/arsiv/19-yapisal-olmayan-verinin-analizi-ve-big-data (21.05.2015)
12. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (21.05.2015)
13. Jeffrey A. Delmerico, Nathanial A. Byrnes, Andrew E. Bruno, Matthew D. Jones, Steven M. Gallo and Vipin Chaudhary, "Performance of Clusters, Hadoop and Active Disks on Microarray Correlation Computations", Page 9, Department of Computer Science and Engineering, University at Buffalo, Dec 2009

14. Jeffrey Jestes, “MapReduce Job Processing”, The School of Computing at the University of Utah, 6-9, Nisan 2012
15. https://www.ibm.com/developerworks/library/wa-introhdfs (03.05.2015)
16. Dr. Atakan ERDEM, İstanbul Aydın Üniversitesi, BYL644, 2014 (21.05.2015)
17. Tom White, “Hadoop: The Definitive Guide”, O’Reilly Media, Inc.June, ISBN:978-1-449-38973-4, Page 63, 2009
18. http://en.wikipedia.org/wiki/Remote_procedure_call (10.05.2015)
19. İlginç Demir, Hadoop Tabanlı Büyük Ölçekli Görüntü İşleme Altyapısı, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği, 21-22, 2012
20. Tom White, “Hadoop: The Definitive Guide”, O’Reilly Media, Inc.June, ISBN:978-1-449-38973-4, Page 66, 2009
21. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Data+Organization (10.05.2015)
22. Tom White, “Hadoop: The Definitive Guide”, O’Reilly Media Inc., June, ISBN:978-1-449-38973-4, Page 66, 2009
23. http://www.widriksson.com/raspberry-pi-hadoop-cluster (11.05.2015)
24. Alan Gates, “Programming Pig”, O’Reilly Media Inc., ISBN:978-1-449-302641, Page 1, 2011
25. Edward Capriolo, Dean Wampler, and Jason Rutherglen, “Programming Hive”, O’Reilly Media Inc., ISBN:978-1- 31933-5, Page 6, October 2012
26. https://cwiki.apache.org/confluence/display/Hive/Home (10.05.2015)
27. Hennessy, John L.; David A., Patterson (2012). Computer Architecture: A Quantitive Approach. Waltham, MA: Morgan Kaufmann. pp. 46–47. ISBN 978-0-12-383872-8

ÖZGEÇMİŞ

Recep Ali Yılmaz, 1981 yılı Rize doğumludur. İlköğrenimini ve ortaokulu Rize' de tamamladı. Rize Mimar Sinan Teknik Lisesi Bilgisayar Bölümünden mezun oldu. 2004 yılında girdiği Karadeniz Teknik Üniversitesi Bilgisayar Teknolojisi ve Programlama bölümünden 2006 yılında mezun oldu. 2008 yılında girdiği Anadolu Üniversitesi İşletme Fakültesi' nden 2010 yılında mezun oldu. 2008 yılından itibaren Koç Holding bünyesinde Sistem ve Network Yönetimi Takım Lideri olarak çalışmaktadır. 2013 yılında başladığı Maltepe Üniversitesi Bilgisayar Mühendisliği Yüksek Lisans programına tez aşamasında devam etmektedir.