T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HADOOP MAPREDUCE ALGORİTMASININ ANALİZİ İLE PERFORMANSA ETKİ EDEN PARAMETRELERİN TESPİTİ VE HADOOP ÜZERİNDE BAŞARIM ARTIMI

YÜKSEK LİSANS TEZİ

Hüseyin ŞARKIŞLA

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	**:**	**BİLİŞİM TEKNOLOJİLERİ**
Tez Danışmanı	**:**	**Yrd. Doç. Dr. Hayrettin EVİRGEN**

Haziran 2015

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HADOOP MAPREDUCE ALGORİTMASININ ANALİZİ İLE PERFORMANSA ETKİ EDEN PARAMETRELERİN TESPİTİ VE HADOOP ÜZERİNDE BAŞARIM ARTIMI

YÜKSEK LİSANS TEZİ

Hüseyin ŞARKIŞLA

Enstitü Anabilim Dalı	:	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	:	**BİLİŞİM TEKNOLOJİLERİ**

Bu tez 25/ 06/2015 tarihinde aşağıdaki jüri tarafından oybirliği/~~oyçokluğu~~ ile kabul edilmiştir.

Yrd. Doç. Dr. Hayrettin Evirgen	Doç.Dr.Numan Çelebi	Yrd.Doç.Dr. Fuat Şimşir
.....	.....	.....
Jüri Başkanı	**Üye**	**Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Hüseyin ŞARKIŞLA

15.05.2015

ÖNSÖZ

Bugün kullandığımız web teknolojileri sayesinde bir yandan internette dolaşırken bir yandan da kendi içeriğimiz kullanıma sunulmaktadır. Resim veya video yükleniyor, milyonlarca e-posta gönderiliyor ve internette gezinirken bile arkamızda bir takım veriler bırakıyoruz. Sadece üretilen veri boyutu değil bu verinin büyüme hızı da artıyor.

Veri büyürken bu verinin içerisinden anlamlı olanları ayıklamak firmalar ve kurumlar için son derece önemli hale gelmiştir. Firmalar ve kurumlar kullanıcı eğilimlerini takip etmek ve kullanıcı ile ilgili bazı analizleri yapmak ancak bu büyük verinin dağıtık bir yapıda hızlı bir şekilde işlenmesi ile mümkün olduğuna kanaat getirmişlerdir. Örneğin Google'ın Web üzerinde dolaşırken karşımıza ilgimizi çekecek reklamları getirebilmesi ya da Amazon'da alışveriş yaparken beğenilerimize uygun ürünlerin karşımıza çıkması dağıtık veri işleme sayesinde olmaktadır. Dağıtık veri işleme araçlarından Hadoop günümüzün en önemli teknolojilerinden diyebiliriz. Birçok firma bu sistemi kullanarak başarıya ulaşmıştır. Bu şirketler arasında Facebook, Yahoo, Google, Amazon gibi teknoloji devleri başı çekmektedir. Türkiye'de ise önemli telekomünikasyon şirketleri, danışmanlık şirketleri Hadoop ile ilgilenmektedir.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

Ana düğüm	: Hadoop kümesi ana bilgisayarı (Master / Name Node)
Cpu	: Merkezi İşlemci
Dfs.block.size	: Girdi verisi blok boyutu
Düğüm	: Kümedeki bilgisayar veya işi yapacak nesne (Node)
Hadoop	: Apache hadoop kütüphanesi
Hdfs	: Hadoop dağıtık dosya sistemi
Jvm	: Java sanal makinesi (java virtual machine)
Gb	: Giga bayt
Görev denetleyici	: Hadoop kümesi görev denetleyici bilgisayar (Task Tracker)
İşçi düğüm	: Hadoop kümesi çalışan bilgisayarı (Slave / Data Node)
İş denetleyici	: Hadoop kümesi iş denetleyici bilgisayarı (Job Tracker)
Io.sort.factor	: Birleştirilebilecek veri bloğu sayısı
Io.sort.mb	: Tampon bellek miktarı
Mapred.compress .map.output	: Veri sıkıştırma durumu
Mapred.reduce. parallel.copies	: Kopyalamada kullanılabilecek izlek sayısı
Map reduce	: Map reduce kütüphanesi
Mapred.tasktrack er.map/reduce.tas ks.maximum	: Bir görev denetleyici için maksimum görev sayısı
Mapred.map/redu ce.tasks.speculati ve.execution	: Bir görevin meşgul göreve katılma durumu
Mb	: Mega bayt

Master	: Yöneten ana düğüm
Ssh	: Güvenli kabuk (Secure shell)
Raid	: Ucuz disklerin artıklıklı dizisi
Rpm	: Dakikadaki okuma sayısı

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Hadoop, map reduce, hdfs, map reduce merformans parametreleri

Map reduce kütüphanesi Google tarafından bilişim dünyasına kazandırılan dağıtık mimari üzerinde çok büyük verilerin kolay bir şekilde analiz edilebilmesini sağlayan programlama modelidir. Bu doküman hadoop map reduce algoritması iş akışını inceler ve map reduce işlemlerinin ve yapılandırma parametrelerinin farklı aşamalardaki farklı kullanımını ve yapılandırma parametrelerinin varsayılan değerleri, artıları eksileri ve tavsiye edilen "Konfigürasyon Parametre Modeli" 'ni açıklar.

Uygulamaya özgü "Konfigürasyon Parametre Modeli"ni oluşturmak için uygulama ortamı düğümler arasında koordinasyonu sağlayan bir bilgisayar ve verilerin saklandığı dört adet bilgisayar olmak üzere toplam beş bilgisayardan oluşmuş, her bir bilgisayar 1 gb/s ile haberleşen anahtar ile birbirine bağlanmış ve hadoop küme yapısı oluşturulmuştur. Deneyde yapılan testler ile parametreler için en uygun değer değerler tespit edilmiştir. Amacımız az donanım maliyeti ile ölçekleme yaparak hadoop map reduce sistemi için en uygun değer yapılandırma parametrelerini bulup tavsiye edilen "Konfigürasyon Parametre Modeli" ' ni açığa çıkarmaktır.

FINDING CONFIGURATION PARAMETERS AFFECTING PERFORMANCE AND USING OPTIMIZED PARAMETERS INCREASING THROUGHPUT ON HADOOP CLUSTER

SUMMARY

Keywords: Hadoop, map reduce, hdfs, map reduce performance parameters

Map reduce framework is a programming model brought to information world by Google that enables very large data analyzed in easy way on distributed architecture. This study analyses hadoop map reduce algorithm in a way that it describes different phases of map reduce operations, usage of configuration parameters in the map reduce job. It explains the configuration parameters, their default values, advantages, disadvantages, and creates a "Configuration Parameter Model" with suggested values in different conditions for this cluster.

In order to create Configuration Parameter Model, hadoop map reduce cluster is created on environment for experiment which has five computers and has got one main computer which enables coordinating with master node and four computers which are slave nodes. The experiments are made on parameters which is trouble for cluster, optimum parameters values detected made by running tests. Our goal is to expose suggested "Configuration Parameter Model" by finding optimum configuration parameters using cluster and by decreasing hardware cost minimum.

BÖLÜM 1. GİRİŞ

2004 yılında Google tarafından geliştirilen map reduce paralel hesaplama kütüphanesi büyük veri işleme sorunları için etkili bir çözüm haline gelmiştir [11]. İki fonksiyonu olan, map ve reduce basit programlama arabirimleri sayesinde, map reduce bilişim dünyasında önemli ölçüde birçok büyük veri uygulamalarının tasarım ve uygulamasını kolaylaştırmıştır. Ayrıca, yük dengeleme, ölçeklenebilirlik ve hata toleransı dahil olmak üzere yaygın olarak kabul edilen diğer avantajlar da sunmaktadır. Hadoop ise endüstride ve akademik araştırılmalarda yaygın olarak kullanılan map reduce kütüphanesinin java programlama dili ile yazılmış açık kaynak uygulamasıdır [1].

Birçok çalışma, farklı düzeylerde veya farklı açılardan hadoop map reduce kütüphanesi performansını artırmak için yapılmıştır. Bunlar birkaç kategoriye ayrılmaktadır. Bunlardan ilki iş veya görevlerin daha akıllıca yürütülmesi için bunların sırasını optimize etmek amacıyla, zamanlama algoritmaları tasarımına odaklanmış [12][13]. İkinci grupta özel donanım veya yazılım yardımı ile map reduce verimliliğinin nasıl artırılması gerektiğine yönelik yapılan araştırılmalar yer almakta [14], üçüncü grupta ise belirli bir tipe yönelik özel map reduce uygulamaları için performans iyileştirmeleri yapılmıştır [15]. Bazı araştırmalar ise yürütme performansını artırmak için optimize edilmiş hadoop map reduce yapılandırma ayarları veya parametrelerini keşfetmeye odaklanmıştır.

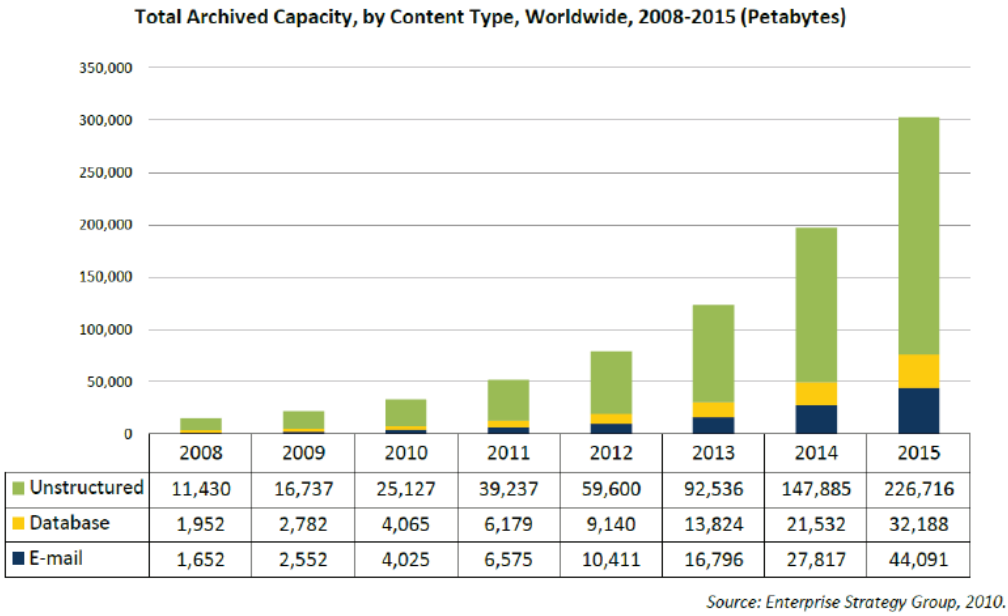
Birçok araştırmacı hadoop zamanlama algoritmalarını optimize etmek için çalışmalar yapmışlardır. 2009 yılında, Zaharia heterojen hadoop kümeleri üzerinde performansını artırmak için LATE (Longest Approximate Time to End) adında özel bir görev zamanlama algoritması önermiştir [12]. Tüm hadoop kümelerin genel performansını artırmaya yönelik, Hsin-Han dinamik yüklemeden kaynaklanan

sorunu çözmek için Load-Aware zamanlayıcı adlı yeni bir algoritma önermiştir [13]. Aynı amaç için Radheshyam tarafından hadoop kümesi üzerinde çalışan farklı iş tiplerini yürütebilen bir zamanlayıcı tasarlanmıştır. Başka bir çalışma da ise map reduce performansını artırmak için Localitiy-Aware Görev Zamanlayıcısı (Larts) adında başka bir yaklaşım önerilmiştir [16]. Bu çalışmaların ortak yönleri farklı çalışma durumu için akıllı ve verimli iş ve görev planlaması yaparak hadoop map reduce performansını artırmaya yöneliktir. Bu çalışmalara benzer hadoop map reduce algoritması performans iyileştirmesi için birçok çalışma mevcuttur. Ancak çalışmaların birçoğu mevcut kaynakları kullanarak yapılandırılan hadoop kümesinden ve hadoop map reduce kütüphanesinin yapılandırma ayarlarının sisteme özgü hangi değerlerde en uygun değerde olacağı üzerine konularını içermemektedir. Bizim çalışmamamızda ise maliyetleri çok yüksek olan donanımlar almak yerine mevcut olan artık kullanılmayan sıradan bilgisayarlar ile hadoop kümesi oluşturmaktır. Oluşturulan bu kümenin map reduce algoritma analizini, iş akışını yapıp sistemi en uygun hale getirmek için performansa etki eden hadoop yapılandırma ayar parametrelerini tespit ettikten sonra bu yapıyı genelleştirerek "Optimum Konfigürasyon Parametre Modeli" oluşturmaktır. Uygulama ortamı için sisteme güvenlik duvarı logları yüklenip önce varsayılan parametre ayarları ile test betikleri çalıştırılıp problemler ile karşılaşılmış ve bu yapıdaki hadoop kümesi için probleme neden olan parametreler ve değerleri üzerinde deneyler yapılmıştır. Bu yapıyı oluştururken her bir düğümde yapılandırılmış olan Ganglia [17] aracı ile karşılaşılan problemler tespit edilip sistemin kurulmasında probleme neden olabilecek kaynaklar açıklanmıştır. Problemler aşıldıktan sonra logları Linux ortamında istediğimiz şekilde dönüşüm işlerine tabi tutulmuş ve Apache Pig sorguları ile test edilmiştir. Burada dosyalar içerisinde yer alan "IP", "TARİH", "URL1", "URL2", "BROWSER" alanlarını anlamlandırabilmek ve Apache Pig tarafında sütunların ve sütün bilgilerinin doğru bir şekilde oluşması için Sed Unix Komut Düzenleyicisi ile metin dönüşüm işlemleri yapılmıştır.

1.1. Büyük Veri

Dijital olarak depolanan veri hacmi son yıllarda artış göstermektedir. Firmalar ve kurumlar için bu büyük veri, doğru analiz metotları ile yorumlanarak stratejik kararlarını doğru biçimde almalarına, risklerini daha iyi yönetmelerine ve inovasyon yapmalarını sağlamaktadır.

Çoğu kez kurum ve firmalar kaydedilen veri yönetimi için veri tabanları yönetim sistemleri kullanırlar. Ancak veri hacmi çok büyük ise sıradan veri tabanı yönetim sistemleri sorguları işlemede ve cevap vermede yetersiz kalır. Örneğin dünyaca bilinen perakendecilerden olan Wall-Mart veri tabanında saatte 2.5 peta bayt veri işlenmektedir. Depolanan verinin büyüklüğünü daha iyi görmek için "Enterprise Strategy Group" tarafından bulunan sonuçlar aşağıdaki grafikte yer almaktadır.



Şekil 1.1. İçeriğe göre 2008-2015 yılları arasındaki arşivlenen veri grafiği

1.2. Veri Analizi

Veri analizi, daha doğru karar almak amacıyla faydalı çıkarımlar yapmak için ham verinin analiz edilmesi işlemidir. Birçok veri müşteriler veya çalışanlar için onlar hakkında önemli bilgilere sahip olduğundan, analiz edildiği takdirde gelecekteki iş planlaması veya müşteri memnuniyeti hakkında daha doğru kararlar önerebilir.

Örneğin belirli bir bütçeye sahip bir kütüphane kitap satın alacağı zaman hangi kitapların daha çok ödünç alındığı, bu kitapların kaç bölüm tarafından kullanıldığı gibi bilgilere ihtiyacı vardır. Yani ne kadar çok bilgiye sahipsek o kadar da iyi kararlar ve sonuçlar elde ederiz. Bu ve buna benzer büyük veri analiz uygulamalarını hızlı ve esnek bir şekilde yapabilmek için dağıtık hesaplama teknolojisi geliştirildi.

1.3. Dağıtık Hesaplama

Dağıtık hesaplama bir uygulamanın çoklu sistemler üzerinde yürütülmesi anlamına gelir. Bu hesaplama çok fazla işlem gerektiren programların, birden fazla göreve bölünüp, görevlerin bilgisayarlar arasında dağıtılmasıyla gerçekleşir. Dağıtılmış bilgisayarlar ortak bir ağda birbirlerine ağ bağlantısı kullanarak iletişim kurarlar.

Dağıtık hesaplama, görevleri paralel ve birbirinden bağımsız varsayarak uygulamanın gerçekleştirilmesini sağlar; bu sayede hiçbir görev bir diğerini beklemez. Bu paralel çalışma, aynı anda birden fazla işlem ve eş zamanlı okuma gibi özelliklerinden dolayı büyük veri analizi sorununu çözmüştür.

Dağıtık Hesaplama teknolojisinin genel faydaları şunlardır:

- Ölçeklenebilirlik
- Hız
- Yedekleme

Ölçeklenebilirlik temelde birden fazla bilgisayarı kümeye ekleyebilme anlamına gelir. Eğer onlarca birlikte çalışan bilgisayara sahip bir kümemiz varsa çok yük olmadan ve problemsiz bir şekilde başka bir bilgisayarda bu kümeye ilave edilebilmelidir.

Ayrıca, hız da bir avantajdır. Çünkü birçok hesaplamayı ve okumayı paralel yapabildiğimizden herhangi tek bir makineye kıyasla daha yüksek oranda işlem hızına sahip olduğumuz anlamına gelmektedir.

Son avantaj olan yedeklemede ise kümede tek bilgisayar devre dışı kaldığı zaman, diğerleri işleme devam eder, tüm veri kalan bilgisayarlara kopyalanır ve devre dışı kalan bilgisayardan dolayı yarım kalan bir işlem varsa, başka bir bilgisayar tarafından yeniden başlatılır.

Dağıtık Hesaplamanın bazı dezavantajları da vardır:

- Güvenlik
- Problem anında hatayı ayıklamanın zor olması

Dağıtık kümedeki bilgisayarların tek bir bilgisayar göre daha az güvenli olduğu bir gerçektir. Çünkü bilgisayarlar arasındaki ağ iletişimi, her bilgisayardaki kimlik denetimi işlemi ve her kopyalanan verinin şifrelenmesi gibi güvenlik açıklarına sebep olacak olayların yönetilmesi gerekmektedir.

Hata ayıklama işlemi dağıtık kümede her bilgisayarda olabileceğinden, yöneticinin her bir makineye bağlanıp logları incelemesi bu işlemi zorlaştırmaktadır.

BÖLÜM 2. HADOOP

Hadoop kümesi veri depolama ve dağıtık bilgisayar ortamında yapılandırılmamış büyük miktarlarda verileri analiz etmek için özel olarak tasarlanmış hesaplama kümesinin özel bir türüdür. Bu kümeler düşük maliyetli sıradan bilgisayarlarda hadoop açık kaynak kodlu dağıtık işleme yazılımı üzerinde çalışır. Kümedeki iş görev dağılımını yapan, yönetimi ve güvenliği sağlayan ana bilgisayarlar olarak bir bilgisayar ana düğüm ve bir bilgisayar da iş izleyici makinesidir. Diğer bilgisayarlar veri düğüm ve görev izleyici olarak belirlenir ve kendilerine verilen iş ve iş parçacıklarını yerine getirmekle görevlidirler.

Hadoop kümeleri ölçeklenebilir veri analizi yapan uygulamaların hızını artırmak için kullanılır. Veri hacminin büyümesi durumunda sisteme harici disk eklemek gibi harici küme düğümleri eklenerek verimlilik ve performans artırılabilir. Ayrıca hadoop kümesinin disk bozulmalarına ve veri kayıplarına karşı kendi güvenlik önlemleri vardır. Her veri bloğu, diğer küme düğümleri üzerine kopyalanır ve bu kopyalama sayısı parametre olarak değiştirilebildiğinden güvenlik seviyesi kullanıcıya bağlıdır. Ancak varsayılan olarak her bir bloğu 3 farklı yerde tutar.

2.1. Hadoop Map Reduce Genel Yapısı

Hadoop map reduce, iş denetleyici ve görev denetleyici süreçlerinden oluşur. İş denetleyici yazılan map reduce programının küme üzerinde dağıtılarak çalıştırılmasından sorumludur. Ayrıca dağıtılan iş parçacıklarının çalışması sırasında oluşabilecek herhangi bir problemde o iş parçacığının sonlandırılması ya da yeniden başlatılması da iş denetleyicinin sorumluluğundadır.

Görev denetleyici, işçi düğümlerin bulunduğu bilgisayarlarda çalışır ve iş denetleyiciden tamamlanmak üzere iş parçacığı talep eder. İş denetleyici, ana düğümün yardımıyla işçi düğümün yerel diskindeki veriye göre en uygun map işini görev denetleyiciye verir. Bu şekilde verilen iş parçacıkları tamamlanır ve sonuç çıktısı yine hadoop dosya sistemi üzerinde bir dosya olarak yazılarak program sonlanır.

2.2. Hadoop Sistemine Giriş

Hadoop, sıradan sunuculardan oluşan küme üzerinde büyük verileri işlemek amaçlı uygulamaları çalıştıran ve hdfs (hadoop dağıtık dosya sistemi) ile hadoop map reduce özelliklerini bir araya getiren, java ile geliştirilmiş açık kaynaklı bir kütüphanedir. Daha yalın bir dille anlatmak gerekirse, hadoop, hdfs ve map reduce bileşenlerinden oluşan bir yazılımdır [18].

2.2.1. Hadoop kümesi ve hdfs bileşenleri

Hadoop kümesi 5 ana bileşenden oluşur. Bunlar:

- Ana düğüm
- Veri düğüm
- İkincil ana düğüm
- İş izleyici
- Görev izleyici

2.2.2. Hadoop kümesi ve hadoop mapreduce kütüphanesi

Hadoop dağıtık hesaplama ve dağıtık depolama işlemleri için master/slave mimarisini kullanır. Bu dağıtık depolama sistemi “hadoop dosya sistemi” veya “hdfs” olarak adlandırılır.

Ana düğüm ana süreç olarak blokların sunucular üzerindeki dağılımından, yaratılmasından, silinmesinden, bir blokta sorun meydana geldiğinde yeniden oluşturulmasından ve her türlü dosya erişiminden sorumludur. Kısaca metadata olarak adlandırılan hdfs üzerindeki tüm dosyalar hakkındaki bilgiler ana düğüm tarafından saklanır ve yönetilir. Her kümede yalnızca bir adet ana düğüm bulunur.

Veri düğümün işlevi blokları saklamak olan işçi süreçtir. Her veri düğüm kendi yerel diskindeki veriden sorumludur. Ayrıca diğer veri düğümlerdeki verilerin yedeklerini de barındırır. Veri düğümler küme içerisinde birden fazla olabilir.

İstediğimiz verileri filtrelemek için kullanılan map fonksiyonu ve bu verilerden sonuç elde etmenizi sağlayan reduce fonksiyonlarından oluşan program yazıldıktan sonra hadoop üzerinde çalıştırılır. Hadoop map ve reduce bölümlerinden oluşan iş parçacıklarını küme üzerinde dağıtarak aynı anda işlenmesini ve bu işler sonucunda oluşan verilerin tekrar bir araya getirilmesinden sorumludur. Hadoop sisteminin gücü işlenen dosyaların her zaman ilgili düğümün yerel diskinden okunması ile ağ trafiğini meşgul etmemesinden ve birden fazla işi aynı anda işleyerek doğrusal olarak ölçeklenmesinden gelmektedir [18].

Map reduce, iş izleyici ve görev izleyici süreçlerinden oluşur. İş izleyici yazılan map reduce programının küme üzerinde dağıtılarak çalıştırılmasından sorumludur. Ayrıca dağıtılan iş parçacıklarının çalışması sırasında oluşabilecek herhangi bir problemde o iş parçacığının sonlandırılması ya da yeniden başlatılması da iş izleyicinin sorumluluğundadır. Görev izleyici, veri düğümlerinin bulunduğu sunucularda çalışır ve iş izleyicilerde tamamlanmak üzere iş parçacığı talep eder. İş izleyici, ana düğümün yardımıyla veri düğümün yerel diskindeki veriye göre en uygun map işini görev izleyiciye verir. Bu şekilde verilen iş parçacıkları tamamlanır ve sonuç çıktısı yine hdfs üzerinde bir dosya olarak yazılıp program sonlanır.

2.2.3. Hadoop dağıtık dosya sistemi yapısı

Hdfs map reduce gibi kütüphaneler altında çalışan büyük ölçekli dağıtık veri işleme için tasarlanmış bir dosya sistemidir. Dağıtık yani birden çok bilgisayar üzerinde çalışan bir dosya sistemidir ve uygulama verilerinin kaydedilmesi için kullanılır. Hdfs sayesinde sıradan sunucuların diskleri bir araya gelerek büyük, tek bir sanal disk oluştururlar. Bu sayede çok büyük boyutta birçok dosya bu dosya sisteminde saklanabilir. Bu dosyalar bloklar halinde (varsayılan 64MB) birden fazla ve farklı sunucu üzerine (varsayılan üç kopya) dağıtılarak raid benzeri bir yapıyla yedeklenir. Bu sayede veri kaybı önlenmiş olur. Ayrıca hdfs çok büyük boyutlu dosyalar üzerinde okuma işlemi imkânı sağlar, ancak rastlantısal erişim özelliği bulunmaz. Hdfs, ana düğüm ve veri düğüm süreçlerinden oluşmaktadır.

2.2.4. Map reduce algoritması ve çalışma prensibi

Map reduce büyük veri setleri ile yapılacak işlemlerin birden fazla iş birimine dağıtılmasını sağlayan yöntemdir. Bu setler üzerindeki işlemler serisi çeşitli birimlere dağıtılır ve sonra çıktıları birleştirilip sonuç üretilir. İki katmandan oluşmaktadır.

2.2.5. Map reduce fonksiyonları

Map katmanı, isim-değer çiftlerini girdi olarak alır, yapılacak işlemi gerçekleştirir. Ürettiği sonuç listesini ise girdideki isim ile birlikte çıktı olarak verir.

Reduce katmanı map katmanındaki sonuç listelerini toplar ve tek sonuca indirgeme işlemini yapar. Birden fazla iş birimi map işiyle uğraşırken, bir sonraki katman sonuçları toplayıp tek sonuç haline getirir.

2.2.6. Map reduce iş akışı

Map reduce iş akışı iki ana aşamaya ayrılır. Bunlar map işlemleri ve reduce işlemleridir.

2.2.7. Map işlemleri

Map süreçleri, hdfs büyük girdi verisini dfs.block.size tarafından kontrol edilen parametre değerine küçük veri bloklarına böler. Varsayılan olarak bu blok değeri 64 mb'tır. Bu veri blokları map görevlerinin girdi verileridir. Her bir map kısmına giden blok sayısı mapred.max.split.size ve mapred.min.split.size parametrelerinin değerlerine bağlıdır. Eğer minimum değer blok boyutundan küçükse ve maksimum değerde blok değerinden büyükse her map kısmına bir blok gönderilir. Blok verisi girdi formatındaki fonksiyonundaki belirtilen anahtar, değer (key, value) çiftine ayrılır. Map fonksiyonu girdideki her (anahtar,değer) çifti için çalışır ve map fonksiyonundan oluşan çıktı döngüsel olarak bellek tampon kısmına yazılır. Buradaki tampon 100 mb varsayılan değerdir ve io.sort.mb özelliği tarafından kontrol edilir.

Diske dökme, tampon boyutu io.sort.spill.percent özelliği ile kontrol edilen eşik değerine ulaştığında (varsayılan 0.8) , arka planda izlekler (threads) içeriği tampon bellekten temizleyip diske yazmaya başlar. Diske dökme işlemi başladıktan sonra map kısmında tampon belleğe yazma işlemi devam eder. Bu dökmeler Round-Robin Metodu ile mapred.local.dir özelliği ile işte belirtilen yere yazarlar ve her tampon bellek eşik değerine ulaştığında yeni bir dökme dosyası oluşturulur.

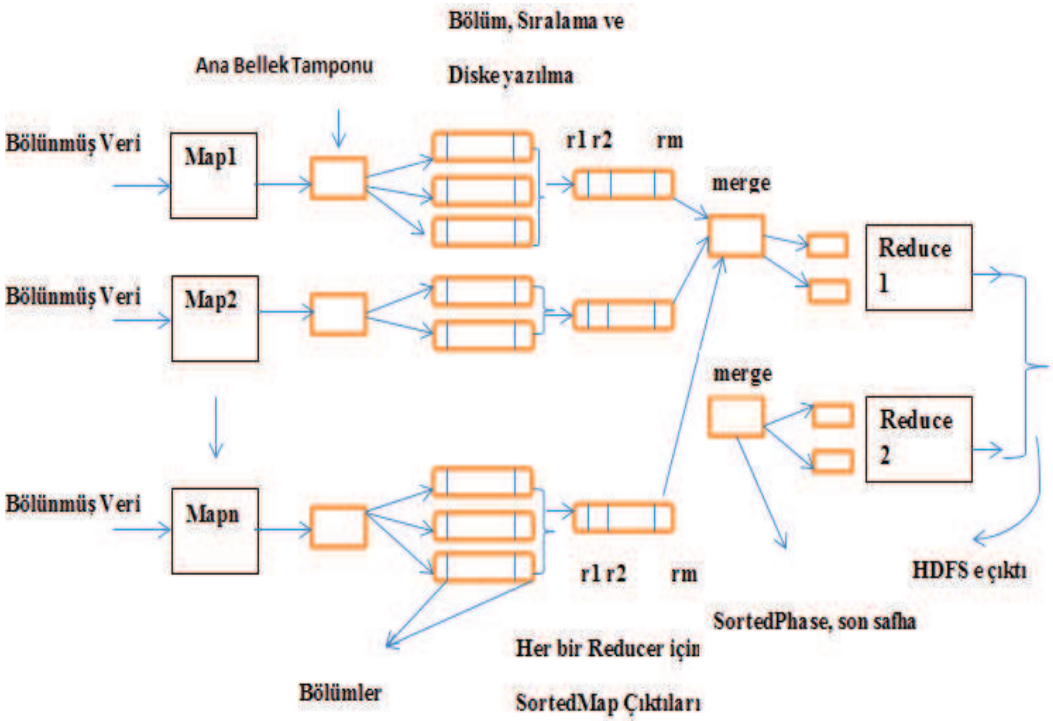
Bölümleme, diske yazmadan önce gönderilecekleri Reducer kısmına göre arka planda veriyi bölümlere ayırır.

Sıralama, bellekteki sıralama anahtar değerine göre işleme alınır. Sıralanmış çıktı verisi eğer birleştirme (combine) fonksiyonu tanımlanmış ise ona gönderilir.

Birleştirme, map görevi bitmeden önce dökme dosyaları tek bir bölümlenmiş ve sıralanmış çıktı dosyası şeklini alır. Dökme dosya sayısı üçten fazla ise birleştirici devreye girer ve birleştirip tek bir dosyaya dönüştürür. İo.sort.factor özelliği bir seferde maksimum hangi sayıda birleştirme yapılacağını belirtir ve varsayılan birleşme değeri 10 değeridir.

Sıkıştırma, map çıktısını daha hızlı diske yazma, daha az yer kaplama ve reducer kısmına giden veri miktarının daha az olmasını sağlamak için diske yazmadan önce sıkıştırma yapabilir. Varsayılan olarak bu değer sıkıştırılmadan yazmaktır. Sıkıştırmayı aktif hale getirebilmek için mapred.compress.map.output değerini "true" yapmak gerekmektedir.

Aşağıdaki diyagram map reduce işini ve iş içindeki veri akışını farklı safhalardan açıklar.



Şekil 2.1. Map reduce algoritmasının küme yapısı üzerinde çalışma prensibi.

2.2.8. Reduce işlemleri

Kopyalama map görev işlemini tamamladıktan hemen sonra reducer kısmında karşılık gelen her map görevi çıktı verisini kopyalanmaya başlar. Reduce görevi map çıktı verilerini paralel olarak işleyen beş izleğe sahiptir ve bu parametre değeri mapred.reduce.parallel.copies özelliği ile değiştirilebilir. Map çıktı verisi reduce görev denetleyici tampon bellek kısmına kopyalanır. Tampon bellek eşik değeri mapred.job.shuffle.merge.percent ve mapred.inmem.merge.threshold özellikleri ile belirlenir. Tampon bellek bu eşik değerine ulaştığında, veri birleştirilir ve oluşturulan dökmeler diske yazılır. Kopyalar diskte toplanırken arka plandaki izlek sıralanmış ve birleştirilmiş dosyaları tekrar birleştirme yapar.

Sıralama, aslında bu safha birleşme safhasıdır, çünkü sıralama işlemi map kısmında yapılır. Bu aşama tüm map işlemleri yapıldıktan ve çıktıları kopyalandıktan sonra başlar. Map çıktıları sıralanmış şekli ile birleştirme yapılır. İo.sort.factor varsayılan olarak on değerinde olduğu için bu sayıdan büyük map sayısı için tur şeklinde çalışır. Kırk adet map çıktımız varsa dört tur sonunda bu işlemi yapacaktır.

Reduce safhasında sıralanmış çıktıdaki her anahtar için reduce fonksiyonu çalışır. Bu aşamanın çıktısı hadoop dağıtık dosya sistemine direkt olarak yazılır.

BÖLÜM 3. HADOOP PARAMETRELER

3.1. Hadoop Konfigürasyon Parametreleri

Hadoop sistemindeki parametreler varsayılan değerleri ve hangi dosyada tutulduğu bilgileri aşağıda verilmiştir.

Tablo 3.1. Hadoop kümesi varsayılan parametre değerleri

Parametre	Hadoop Dosya Adı	Varsayılan
mapreduce.task.io.sort.mb	mapred-site.xml	100
mapreduce.map.sort.spill.percent	mapred-site.xml	0.80
mapreduce.task.io.sort.factor	mapred-site.xml	100
mapreduce.map.combine.minspills	mapred-site.xml	3
mapreduce.job.reduces	mapred-site.xml	1
mapreduce.cluster.local.dir	mapred-site.xml	${hadoop.tmp.dir}/mapred/local
mapreduce.reduce.merge.memtomem.enabled	mapred-site.xml	False
mapreduce.framework.name	mapred-site.xml	yarn/local
mapreduce.reduce.shuffle.parallelcopies	mapred-site.xml	5
mapreduce.reduce.memory.totalbytes	mapred-site.xml	Runtime.maxMemory()
mapreduce.reduce.shuffle.memory.limit.percent	mapred-site.xml	0.25
mapreduce.job.ubertask.enable	mapred-site.xml	False
mapreduce.job.ubertask.maxmaps	mapred-site.xml	9
mapreduce.job.ubertask.maxreduces	mapred-site.xml	1
mapreduce.job.ubertask.maxbytes	mapred-site.xml	dfs.block.size
mapreduce.map. failures.maxpercent	mapred-site.xml	0
mapreduce.reduce. failures.maxpercent	mapred-site.xml	0
mapreduce.map.memory.mb	mapred-site.xml	1024
mapreduce.reduce.memory.mb	mapred-site.xml	1024
mapreduce.reduce.shuffle.merge.percent	mapred-site.xml	0.90

3.2. Performansı Etkileyen Parametreler

Dfs.block.size, girdi verinin bölüneceği veri blokları boyutudur.

Mapred.compress.map.output, çıktı map bölümlerinin sıkıştırılıp sıkıştırılmayacağı parametre değeridir.

Mapred.map/reduce.tasks.speculative.execution, bir görev, yazılım konfigürasyonundan veya donanımdan kaynaklı nedenlerden dolayı yavaş çalıştığında iş denetleyici yedek olarak diğer eş değer görevi çalıştırır. Bu "speculative execution" olarak bilinir. Hangisi önce bitirirse diğeri öldürülür.

Mapred.tasktracker.map/reduce.tasks.maximum, bir görev denetleyici için paralel çalışacak maksimum map/reduce sayısıdır.

Io.sort.mb, map görevi tarafından çıktının sıralama işlemini yaparken kullandığı tampon bellek boyutudur.

io.sort.factor, map ve reduce safhasında sıralama işleminde bir kere de birleştirilecek (merge) veri bloğu (stream) sayısıdır.

Mapred.job.reuse.jvm.num.tasks, bir görev denetleyici üzerindeki her JVM için çalışacak maksimum görev sayısıdır.-1 değeri limit olmadığını ve aynı JVM 'nin bir iş için tüm görevler tarafından kullanabileceğini gösterir.

Mapred.reduce.parallel.copies, map çıktılarını reducer kısmına kopyalamak için kullanılan izlek (thread) sayısıdır.

BÖLÜM 4. HADOOP KONFİGÜRASYON MODELİ

4.1. Konfigürasyon Modeli

Dfs.block.size değeri varsayılan olarak 64 mb'tır.

Birinci durumumuz testteki verilerimize göre modelimiz aşağıdaki gibi olmaktadır.

$$64 < \text{dfs.block.size} < \frac{(10000\ \text{mb} * 100\ \text{mb})}{2000\ \text{mb}}$$

$$\text{dfs.block.size} > 500\ \text{mb}$$

Birinci duruma göre modelimiz aşağıdaki gibi olmaktadır.

$$64 < \text{dfs.block.size} < \frac{(\text{Girdi Veri Boyutu} * \text{Tampon Bellek Boyutu})}{\text{Düğüm Ana Belek Boyutu}}$$

Mapred.compress.map.ouput değeri varsayılan olarak "false" durumundadır. Eğer sistem kısıtlı bir disk alanına sahipse disk alanını tasarruflu kullanıp hızlı yazma ve reducer tarafından bilgiye hızlı ulaşması sağlanır. Yalnız cpu özellikleri burada önem taşır. Tek çekirdeğe sahip düğümlerde bu özelliğin açılması önerilmez.

Mapred.map/reduce.tasks.speculative.execution, meşgul map küme yapısında işi biten bir görevin bu meşgul yapıda diğer görevlere katılarak çalışma zamanın azalttığından bu değerin açık olması toplam başarımı artırır.

Mapred.tasktracker.map/reduce.tasks.maximum bir görev denetleyici için maksimum map/reduce sayısıdır. Varsayılan değeri ikidir.

İkinci durumumuzda eğer düğüm RAM=2 gb ve 1 çekirdek cpu birimine sahipse; bir görev için maksimum gerekli bellek 500 mb ve görev denetleyici, işçi düğüm ve diğer işlemler için 500+ 500+ 500= 1,5 gb olmaktadır.

Maksimum çalışabilecek görev sayısı =

$$= \frac{2000\ \text{gb} - (0{,}5\ \text{gb} + 0{,}5\ \text{gb} + 0{,}5\ \text{gb}))}{0{,}5\ \text{gb}}$$

İkinci duruma göre modelimiz aşağıdaki gibi olmaktadır.

$$\text{Maksimum Görev Sayısı} =$$

$$= \frac{(\text{Toplam Bellek Boyutu} - (\text{GörevDenetleyici} + \text{İşçi düğüm} + \text{İş denetleyici Bellek İhtiyaçları}))}{\text{Bir Görev İçin Gerekli Bellek Miktarı}}$$

Üçüncü durumuzda bir önceki durumda test edilen değerlere oranla; eğer düğüm RAM=8 gb ve 8 çekirdek cpu birimine sahipse; bir görev için maksimum gerekli ana bellek 500mb ve görev denetleyici, işçi düğüm ve diğer işlemler için 1+1+1= 3 gb 'tır.

Maksimum çalışabilecek görev sayısı =

$$= \frac{8\ \text{gb} - 3\ \text{gb}}{0{,}5\ \text{gb}} = 10$$

Sonuç olarak çalışacak map/reduce sayısı bellek kullanımına göre ve görevin hesaplama karmaşıklığına bağlıdır. Io.sort.mb değişkeni sıralama işlemi için tampon bellek boyutu ve varsayılan olarak 100 mb 'tır.

Üçüncü durumumuza göre modelimiz aşağıdaki gibi olmaktadır.

$$\text{io.sort.mb} < \frac{(\text{Düğüm Sayısı} * \text{Ana Bellek Boyutu}) * (\text{dfs.block.size})}{\text{Girdi Verisi Boyutu}}$$

Dördüncü durumda testteki verilerimize göre modelimiz aşağıdaki gibi olmaktadır.

$$5 * 2\text{gb} < \frac{10\ \text{gb}}{64\ \text{mb} * \text{io.sort.mb}}$$

$$\text{io.sort.mb} < 64\ \text{mb}$$

Buradaki durumda bu değer en fazla 64 mb olması önerilir. İo.sort.factor varsayılan olarak bu değer 10 ‘dur. Mapred.job.reuse.jvm.num.tasks varsayılan olarak bir değerini alır.

Dördüncü duruma göre modelimiz aşağıdaki gibi olmaktadır.

$$\text{Toplam Düğüm Sayısı} < \frac{(\text{Girdi Veri Boyutu})}{\text{dfs.block.size} * \text{io.sort.mb}}$$

Beşinci durumda map sayımız aşağıdaki verilere göre oldukça fazla ve olduğundan mapred.job.reuse.jvm.num.tasks değeri -1 alınması önerilir.

$$\frac{10\ \text{gb}}{64\ \text{mb}} > 5$$

Beşinci duruma göre modelimiz aşağıdaki gibi olmaktadır.

$$\frac{\text{Girdi Veri Miktarı}}{\text{Blok Boyutu}} > \text{Düğüm Sayısı}$$

Altıncı durumda mapred.reduce.parallel.copies varsayılan olarak beştir.

Altıncı duruma göre cpu özellikleri yeterli ise bu değer modeli aşağıdaki gibi olmaktadır.

$$5 < \text{mapred.reduce.parallel.copies} < \frac{(\text{Girdi Veri Boyutu})}{\text{dfs.block.size}}$$

Son olarak yedinci model için geçici alan miktarı olan mapred.local.dir değeri en az toplam diskin ¼ katı olmalıdır. Aşağıdaki formülümüz bu oranda olduğunu kanıtlamaktadır.

$$\text{Minimum Toplam Disk Boyutu} > 3 * \text{Analizi Yapılacak Toplam Veri Girdisi} + \text{Görev Sayısı} * \text{Tampon Bellek Miktarı}$$

BÖLÜM 5. MODEL ADIMLARI

5.1. Hadoop Küme Ortamının Hazırlanması

Hadoop küme ortamımızda hadoop1.2.1.jar kütüphanesi ve sistem jdk1.6.33 java uygulaması üzerinde çalışmakta ve tüm testler için bütün girdi ve çıktılar hdfs sisteminde saklanmıştır. Düğümümüz 32-bit ubuntu işletim sistemi üzerinde çalıştırılmıştır. Her bir düğümdeki bilgisayar Intel Pentium 2.8 ghz işlemciye 2 gb bellek ve 150 gb 7200 rpm disk özelliklerine sahiptir.

5.1.1. Düğümler için ubuntu makinelerinin hazırlanması

Her bir bilgisayardaki 150 gb disk kapasitesine sahip depolama alanına test ortamımızda her biri yaklaşık 102 mb boyutunda Ocak 2013 ayına ait güvenlik duvarı logları oluşan 96 adet 10 gb boyutunda test verimizi manuel olarak kopyaladık.

5.1.2. Java kurulumu ve yapılandırması

Hadoop kümesinin çalışabilmesi için her bir düğümde java 1.6 versiyonu veya üzeri bir versiyona sahip Java uygulamasının kurulu olması gerekmektedir. Java 1.6 uygulamasını sistemimizdeki her bir düğümdeki ubuntu işletim sistemi üzerinde kurmak için aşağıdaki komutları sırasıyla uyguladık.

Java kurulumu için gerekli kodlar aşağıdaki gibidir.

- sudo apt-get update
- sudo apt-get install sun-java6-jdk
- sudo update-java-alternatives -s java-6-sun

5.1.3. Kullanıcı tanımlama ve ssh yapılandırması

Hadoop kurumulunu güvenlik, izin yönetimi, yedekleme kolaylığı açısından ve ubuntu işletim sistemi üzerinde çalışan diğer uygulamalardan ayırmak için sadece hadoop için bir grup ve bu gruba ait bir kullanıcı tanımını aşağıdaki komutla gerçekleştirdik.

Kullanıcı ve grup tanımı aşağıdaki gibidir.

- sudo addgroup hadoop
- sudo adduser --ingroup hadoop hduser

Kullanıcı ve grup tanımlaması yapıldıktan sonra hadoop sisteminin düğümlerine güvenli bir bağlantı oluşturabilmesi için hduser kullanıcısı için ssh bağlantı erişimini tanımladık.

Ssh bağlantı erişimi aşağıdaki kodları sırasıyla çalıştırdık.

- su – hduser
- ssh-keygen -t rsa -P ""

İkinci satır boş bir şifre ile RSA anahtar çifti oluşturacak. Aslında gerçek sistemlerde boş bir şifre önerilmez. Ancak düğümler arasındaki geçişi kolaylaştırmak için bizim sistemimizde şifre girilmemiştir. Bu işlemlerden sonra yaratılan bu anahtar ile yerel makineye ssh ile bağlanmak için cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys kodunu kullandık.

Son adım olarak da hduser kullanıcısı ile yerel makineye ssh bağlantısını test etmek için ssh localhost komutunu kullandık.

5.1.4. IPV6 yapılandırması

IPV6 ağ bağlantılarını kullanmadığımız için bu bağlantıları pasif durumuna getirdik. Ülkemizde IPV6 altyapısı henüz hazır durumda olmadığı için IPV6 ile ilgili ayarları pasif duruma getirmemiz gerekti. Bu ayar için /etc/sysctl.conf dosyasını açarak dosya sonuna aşağıdaki komutları ekledik.

- net.ipv6.conf.all.disable_ipv6 = 1
- net.ipv6.conf.default.disable_ipv6 = 1
- net.ipv6.conf.lo.disable_ipv6 = 1

5.2. Log Analizi İçin Tek Düğümlü Hadoop Kümesi

Hadoop kümesi için hadoop 1.2.1.jar uygulamasını indirdikten sonra bilgisayarlarda oluşturduğumuz grup ve kullanıcıya gerekli yetkiyi verdikten sonra aşağıdaki komutları uyguladık.

- $ cd /usr/local
- $ sudo tar xzf hadoop-1.0.3.tar.gz
- $ sudo mv hadoop-1.2.1 hadoop
- $ sudo chown -R hduser:hadoop hadoop

5.2.1. Ayar dosyalarının yapılandırılması

Hadoop kümesinde parametrelerin ve genel yapılandırmaların yapıldığı başlıca ayar dosyaları hadoop-env.sh, core-site.xml, mapred-site.xml ve hdfs-site.xml' dir. Bu dosyaların yapılandırmaları Tablo 5.1. gibi olması gerekmektedir.

Tablo 5.1. Tek düğümlü küme parametre yapılandırması

Yapılandırma Dosyası	Özellik
core-site.xml	<property> <name>hadoop.tmp.dir</name> <value>/app/hadoop/tmp</value> </property>
	<property> <name>fs.default.name</name> <value>hdfs://localhost:54310</value> </property>
mapred-site.xml	<property> <name>mapred.job.tracker</name> <value>localhost:54311</value> </property>
hdfs-site.xml	<property> <name>dfs.replication</name> <value>1</value> </property>

5.2.2. Hdfs dosya sistemini biçimlendirme

Hadoop kümesinin çalışabilmesi yapmamız gerek ilk iş ana düğümü biçimlendirmek oldu. Bu işlem için hduser@ubuntu1:~$ /usr/local/hadoop/bin/hadoop namenode –format komutunu kullandık

5.2.3. Tek düğümlü hadoop kümesini başlatma

Hadoop kümesini başlatmak için hadoop uygulamasını kurulmuş olduğu dizine gelip ,hduser@ubuntu:~$ /usr/local/hadoop/bin/ ./start-all.sh, komutunu çalıştırdık.

5.2.4. Tek düğümlü hadoop kümesini durdurma

Hadoop kümesini durdurmak için hadoop uygulamasını kurulmuş olduğu dizine gelip hduser@ubuntu:~$ /usr/local/hadoop/bin/ ./stop-all.sh komutunu çalıştırdık.

5.2.5. Yerel makinedeki veriyi hadoop dosya sistemine aktarma

Yerel bilgisayardaki log dosyalarını hadoop dosya sistemine aktarabilmek için aşağıdaki komutu çalıştırdık.

- hduser@ubuntu:~$ hadoop fs -mkdir input
- hduser@ubuntu:~$ hadoop fs –put /user/hduser/İndirilenler/4ok /user/hduser/input

5.2.6. Hadoop web ara yüzleri

Web ara yüzleri şekiller bölümünde Şekil 5.1, Şekil 5.2, Şekil 5.3 görüntülerinden ulaşılabilir.

5.2.7. Ana düğüm web ara yüzü

Web ara yüzleri şekiller bölümünde Şekil 5.1, Şekil 5.2, Şekil 5.3 görüntülerinden ulaşılabilir.

5.2.8. İş izleyici ve görev izleyici web ara yüzü

Web ara yüzleri şekiller bölümünde Şekil 5.1, Şekil 5.2, Şekil 5.3 görüntülerinden ulaşılabilir.

5.3. Log Analizi İçin Çok Düğümlü Hadoop Kümesi Kurulumu

Çok düğümlü hadoop kümesinin kurulumu için tek düğüm sistemde ubuntu bilgisayarı için yaptığımız işlemleri diğer dört bilgisayar için de yaptık. Her biri tek düğüm sistemde çalışacak gibi test edildi. Çok düğümlü sisteme entegre etmek için her bir bilgisayarın “host” dosyası güncellendi ve aşağıdaki gibi varsayılan parametre ayarları yapıldı.

5.3.1. Ayar dosyalarının yapılandırılması

Tablo 5.2. Çok düğümlü küme parametre yapılandırması

Yapılandırma Dosyası	Özellik
core-site.xml	<property> <name>hadoop.tmp.dir</name> <value>/app/hadoop/tmp</value> </property>
	<property> <name>fs.default.name</name> <value>hdfs://ubuntu1:54310</value> </property>
mapred-site.xml	<property> <name>mapred.job.tracker</name> <value>ubuntu1:54311</value> </property>
hdfs-site.xml	<property> <name>dfs.replication</name> <value>3</value> </property>

BÖLÜM 6. MODELİN UYGULANMASI

6.1. Tek Düğümlü Kümede Uygulama Deneyi

6.1.1. Amaç

Bu deneyin amacı tek düğümlü küme yapısı üzerinde varsayılan parametre ayarları ve bizim optimize ettiğimiz parametre ayarları ile pig betiğini çalıştırıp kullanılan kaynak miktarları, işlem hızları ve arasındaki farkları göstermektir.

6.1.2. Parametre yapılandırması

Tek düğümlü küme üzerinde yapılan deneyde 2.8 ghz saat hızı i işlemciye sahip sıradan bir bilgisayar kullandık. hadoop 1.2.1 versiyonu ve java 1.6 uygulamalarını Linux ubuntu üzerine kurulumunu ve yapılandırmasını yaptık. Hadoop sisteminin düğümleri ile düzgün bir şekilde haberleşmesini sağlayan ssh yapılandırmasını yaptıktan sonra bizim belirlediğimiz parametre değerlerini aşağıdaki tabloda görüldüğü gibi girdik.

Tablo 6.1. Tek düğümlü kümenin eklediğimiz parametrelerle yapılandırması

Yapılandırma Dosyası	Özellik
core-site.xml	<property> <name>hadoop.tmp.dir</name> <value>/app/hadoop/tmp</value> </property>
	<property> <name>fs.default.name</name> <value>hdfs://localhost:54310</value> </property>
mapred-site.xml	<property> <name>mapred.job.tracker</name> <value>localhost:54311</value> </property> <property> <name>mapred.child.java.opts</name> <value>-Xmx1024M</value> </property> <property> <name> mapred.tasktracker.map </name> <value>1</value> </property>
hdfs-site.xml	<property> <name>dfs.replication</name> <value>1</value> </property> <property> <name>dfs.block.size</name> <value>134217728</value> </property>

6.1.3. Sonuç

Varsayılan değerlerle sistem çalıştırıldığında işlemci kullanımı ve işlem adımları aşağıdaki gibi olmaktadır.

2015-05-09 15:50:10,704 [main] INFO . - 0% complete

2015-05-09 15:50:11,338 [main] INFO . - HadoopJobId: job_201505091531_0002
2015-05-09 15:50:11,338 [main] INFO . - Processing aliases A,data
2015-05-09 15:50:11,338 [main] INFO . - detailed locations: M: A[5,4],A[-1,-1],data[7,7] C: R:
2015-05-09 15:50:11,338 [main] INFO . - http://ubuntu1:50030/jobdetails.jsp?jobid=job_201505091531_0002
2015-05-09 15:50:34,434 [main] INFO . - 4% complete
2015-05-09 15:50:34,434 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:50:53,501 [main] INFO . - 9% complete
2015-05-09 15:50:53,501 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:51:13,568 [main] INFO . - 13% complete
2015-05-09 15:51:13,568 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:51:31,128 [main] INFO . - 18% complete
2015-05-09 15:51:31,129 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:51:51,701 [main] INFO . - 22% complete
2015-05-09 15:51:51,702 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:52:10,265 [main] INFO . - 27% complete
2015-05-09 15:52:10,266 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:52:23,817 [main] INFO . - 31% complete
2015-05-09 15:52:23,817 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:52:37,367 [main] INFO . - 36% complete
2015-05-09 15:52:37,367 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:52:51,419 [main] INFO . - 40% complete

2015-05-09 15:52:51,419 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:53:05,967 [main] INFO . - 45% complete
2015-05-09 15:53:05,967 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:53:21,015 [main] INFO . - 50% complete
2015-05-09 15:53:21,015 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:53:23,021 [main] INFO . - Running jobs are [job_201505091531_0002]
2015-05-09 15:53:26,059 [main] INFO . - 100% complete

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.2.1 0.13.0 hduser 2015-05-09 15:50:02 2015-05-09 15:53:26 FILTER
Success!
İşlem Zamanı = StartAt- FinishedAt=15:53:26 - 15:50:02 = 204 saniye

		Map	Reduce	Toplam
FileSystemCounters	FİLE_BYTES_WRITTEN	4.640.186	0	4.640.186
	HDFS_BYTES_READ	2.309.019.162	0	2.309.019.162
	HDFS_BYTES_WRITTEN	1.647.702	0	1.647.702
Map-Reduce Framework	Map input records	0	0	6.824.240
	Physical memory (bytes) snapshot	0	0	1.649.659.904
	Spilled Records	0	0	0
	Total committed heap usage (bytes)	0	0	715.128.832
	CPU time spent (ms)	0	0	199.990
	Virtual memory (bytes) snapshot	0	0	29.645.090.816
	SPLIT_RAW_BYTES	16.278	0	16.278
	Map output records	0	0	5.812

Şekil 6.1. Tek düğümlü kümede varsayılan parametrelerle sistem çalıştığında oluşan özet bilgiler

Optimize edilmiş değerlerle sistem çalıştırıldığında işlem hızı ve işlem adımları aşağıdaki gibi olmaktadır.

2015-05-09 15:36:18,402 [main] INFO . - http://ubuntu1:50030/jobdetails.jsp?jobid=job_201505091531_0001

2015-05-09 15:36:36,595 [main] INFO . - 4% complete
2015-05-09 15:36:36,596 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:36:50,161 [main] INFO . - 8% complete
2015-05-09 15:36:50,161 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:37:01,722 [main] INFO . - 12% complete
2015-05-09 15:37:01,722 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:37:14,794 [main] INFO . - 16% complete
2015-05-09 15:37:14,794 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:37:28,369 [main] INFO . - 20% complete
2015-05-09 15:37:28,369 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:37:43,443 [main] INFO . - 25% complete
2015-05-09 15:37:43,443 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:37:55,504 [main] INFO . - 29% complete
2015-05-09 15:37:55,504 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:38:09,074 [main] INFO . - 33% complete
2015-05-09 15:38:09,074 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:38:21,131 [main] INFO . - 37% complete
2015-05-09 15:38:21,132 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:38:33,192 [main] INFO . - 41% complete
2015-05-09 15:38:33,192 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:38:44,242 [main] INFO . - 45% complete

2015-05-09 15:38:44,242 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:38:50,769 [main] INFO . - 50% complete
2015-05-09 15:38:50,769 [main] INFO . - Running jobs are [job_201505091531_0001]
2015-05-09 15:38:53,332 [main] INFO . - 100% complete
2015-05-09 15:38:53,346 [main] INFO SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.2.1 0.13.0 hduser 2015-05-09 15:36:07 2015-05-09 15:38:53 FILTER
Success!
İşlem Zamanı = StartAt- FinishedAt=15:38:53 - 15:36:07 = 166 saniye

		Map	Reduce	Toplam
FileSystemCounters	FILE_BYTES_WRITTEN	2.529.638	0	2.529.638
	HDFS_BYTES_READ	2.308.921.580	0	2.308.921.580
	HDFS_BYTES_WRITTEN	1.647.702	0	1.647.702
Map-Reduce Framework	Map input records	0	0	6.824.240
	Physical memory (bytes) snapshot	0	0	905.228.288
	Spilled Records	0	0	0
	Total committed heap usage (bytes)	0	0	390.070.272
	CPU time spent (ms)	0	0	183.180
	Virtual memory (bytes) snapshot	0	0	16.171.274.240
	SPLIT_RAW_BYTES	8.808	0	8.808
	Map output records	0	0	5.812

Şekil 6.2. Tek düğümlü kümede bizim parametrelerimizle sistem çalıştığında oluşan özet bilgiler

6.2. Çok Düğümlü Kümede Uygulama Deneyi

6.2.1. Amaç

Bu deneyin amacı çok düğümlü küme yapısı üzerinde varsayılan parametre ayarları ve bizim optimize ettiğimiz parametre ayarları ile pig betiğini çalıştırıp kullanılan işlemci ve bellek miktarları ve arasındaki farkları göstermektir.

6.2.2. Parametre yapılandırması

Çok düğümlü küme üzerinde yapılan deneyde 2.8 ghz saat hızı işlemciye sahip sıradan 5 adet bilgisayar kullandık. Her biri için hadoop 1.2.1 versiyonu ve java 1.6 uygulamalarını Linux ubuntu üzerine kurulumunu ve yapılandırmasını yaptık. Hadoop'un düğümleri ile düzgün bir şekilde haberleşmesini sağlayan ssh yapılandırmasını yaptıktan sonra bizim belirlediğimiz parametre değerlerini aşağıdaki tabloda görüldüğü gibi girdik.

- Düğümlerden biri
    1. Hem ana düğüm hem de veri düğüm
    2. İş denetleyici
    3. Görev denetleyici
- Diğer 4 düğüm
    1. Veri düğümler
    2. Görev denetleyiciler

Çok düğümlü küme yapısı kurulumunu aşağıdaki adımları sırasıyla uygulayarak gerçekleştirdik.

- Her bir bilgisayara ayrı ayrı tek düğümlü küme yapısı kurulumu yapıldı.
- Ana düğümün veri düğümlerle haberleşmesini sağlamak için ana düğüm makinesinin ssh anahtarını veri düğüm makinelerine kopyaladık.
- "masters" dosyası içine "ubuntu1" bilgisayar ismini ekledik.
- "slaves" dosyası içine "ubuntu2", "ubuntu3", "ubuntu4", "ubuntu5" bilgisayar isimlerini ekledik.
- Bilgisayarların IP adresleri atandı ve bilgisayarların "hosts" dosyasına işlendi.
- "ubuntu1" bilgisayar IP adresi 192.168.0.1 olarak değiştirdik.
- "ubuntu2", "ubuntu3", "ubuntu4", "ubuntu5" bilgisayarları için sırasıyla 192.168.0.2, 192.168.0.3, 192.168.0.4, 192.168.0.5 adreslerini girdik.
- Tüm düğümlerdeki Hdfs url adresini "hdfs://ubuntu1:54310" olarak değiştirdik.

- Son olarak da “start-dfs.sh” ve “start-mapred.sh” betiklerini çalıştırıp kümemizi aktif hale getirdik.

6.2.3. Sonuç

Varsayılan değerlerle işlem hızı ve işlem adımları aşağıdaki gib gerçekleşmektedir.

2015-05-09 16:55:07,278 [main] INFO .mapReduceLayer.MapReduceLauncher - http://ubuntu1:50030/jobdetails.jsp?jobid=job_201505091650_0001
2015-05-09 16:55:21,991 [main] INFO .mapReduceLayer.MapReduceLauncher - 4% complete
2015-05-09 16:55:21,991 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:23,499 [main] INFO .mapReduceLayer.MapReduceLauncher - 8% complete
2015-05-09 16:55:23,499 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:31,539 [main] INFO .mapReduceLayer.MapReduceLauncher - 12% complete
2015-05-09 16:55:31,540 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:33,551 [main] INFO .mapReduceLayer.MapReduceLauncher - 16% complete
2015-05-09 16:55:33,551 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:40,581 [main] INFO .mapReduceLayer.MapReduceLauncher - 22% complete
2015-05-09 16:55:40,581 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:45,604 [main] INFO .mapReduceLayer.MapReduceLauncher - 28% complete

2015-05-09 16:55:45,604 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:47,615 [main] INFO .mapReduceLayer.MapReduceLauncher - 32% complete
2015-05-09 16:55:47,615 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:53,641 [main] INFO .mapReduceLayer.MapReduceLauncher - 37% complete
2015-05-09 16:55:53,641 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:55:55,650 [main] INFO .mapReduceLayer.MapReduceLauncher - 42% complete
2015-05-09 16:55:55,651 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:56:00,169 [main] INFO .mapReduceLayer.MapReduceLauncher - 47% complete
2015-05-09 16:56:00,169 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:56:04,186 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0001]
2015-05-09 16:56:07,277 [main] INFO .mapReduceLayer.MapReduceLauncher - 100% complete
2015-05-09 16:56:07,286 [main] INFO .SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.2.1 0.13.0 hduser 2015-05-09 16:54:55 2015-05-09 16:56:07 FILTER

Success!
İşlem Zamanı = StartAt- FinishedAt = 16:56:07 - 16:54:55 = 72 saniye

		Map	Reduce	Toplam
FileSystemCounters	FİLE_BYTES_WRITTEN	4.637.678	0	4.637.678
	HDFS_BYTES_READ	2.309.019.162	0	2.309.019.162
	HDFS_BYTES_WRITTEN	1.647.702	0	1.647.702
Map-Reduce Framework	Map input records	0	0	6.824.240
	Physical memory (bytes) snapshot	0	0	1.746.923.520
	Spilled Records	0	0	0
	Total committed heap usage (bytes)	0	0	715.128.832
	CPU time spent (ms)	0	0	192.770
	Virtual memory (bytes) snapshot	0	0	30.474.682.368
	SPLIT_RAW_BYTES	16.278	0	16.278
	Map output records	0	0	5.812

Şekil 6.3. Çok düğümlü kümede varsayılan parametrelerle sistem çalıştığında oluşan özet bilgiler

Optimize edilmiş değerlerle sistem çalıştırıldığında işlem hızı ve işlem adımları aşağıdaki gibi olmaktadır.

2015-05-09 17:07:32,771 [main] INFO .mapReduceLayer.MapReduceLauncher - http://ubuntu1:50030/jobdetails.jsp?jobid=job_201505091650_0003
2015-05-09 17:07:45,856 [main] INFO .mapReduceLayer.MapReduceLauncher - 5% complete
2015-05-09 17:07:45,856 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:07:46,358 [main] INFO .mapReduceLayer.MapReduceLauncher - 10% complete
2015-05-09 17:07:46,358 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:07:49,373 [main] INFO .mapReduceLayer.MapReduceLauncher - 16% complete
2015-05-09 17:07:49,373 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:07:56,402 [main] INFO .mapReduceLayer.MapReduceLauncher - 21% complete
2015-05-09 17:07:56,402 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]

2015-05-09 17:07:57,406 [main] INFO .mapReduceLayer.MapReduceLauncher - 25% complete
2015-05-09 17:07:57,407 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:00,432 [main] INFO .mapReduceLayer.MapReduceLauncher - 30% complete
2015-05-09 17:08:00,432 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:06,960 [main] INFO .mapReduceLayer.MapReduceLauncher - 36% complete
2015-05-09 17:08:06,960 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:08,468 [main] INFO .mapReduceLayer.MapReduceLauncher - 40% complete
2015-05-09 17:08:08,468 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:10,980 [main] INFO .mapReduceLayer.MapReduceLauncher - 45% complete
2015-05-09 17:08:10,980 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:12,988 [main] INFO .mapReduceLayer.MapReduceLauncher - 49% complete
2015-05-09 17:08:12,988 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:15,999 [main] INFO .mapReduceLayer.MapReduceLauncher - Running jobs are [job_201505091650_0003]
2015-05-09 17:08:17,571 [main] INFO .mapReduceLayer.MapReduceLauncher - 100% complete
2015-05-09 17:08:17,574 [main] INFO .SimplePigStats - Script Statistics:

HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.2.1 0.13.0 hduser 2015-05-09 17:07:22 2015-05-09 17:08:17 FILTER

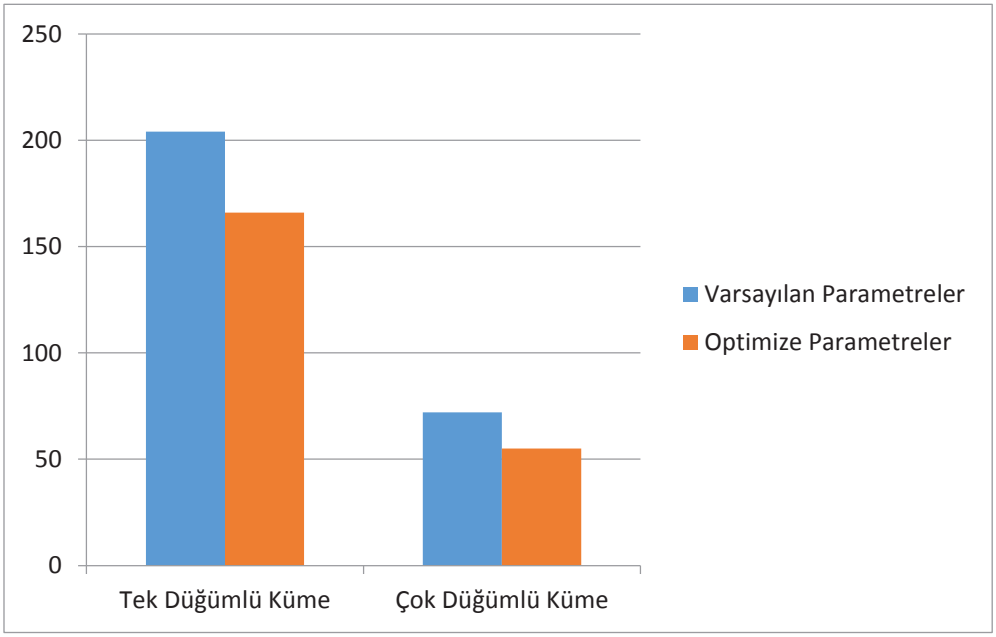
Success!

İşlem Zamanı = StartAt- FinishedAt = 17:08:17 - 17:07:22 = 55 saniye

		Map	Reduce	Toplam
FileSystemCounters	FİLE_BYTES_WRITTEN	2.529.542	0	2.529.542
	HDFS_BYTES_READ	2.308.921.580	0	2.308.921.580
	HDFS_BYTES_WRITTEN	1.647.702	0	1.647.702
Map-Reduce Framework	Map input records	0	0	6.824.240
	Physical memory (bytes) snapshot	0	0	950.165.504
	Spilled Records	0	0	0
	Total committed heap usage (bytes)	0	0	390.070.272
	CPU time spent (ms)	0	0	175.210
	Virtual memory (bytes) snapshot	0	0	16.585.211.904
	SPLIT_RAW_BYTES	8.808	0	8.808
	Map output records	0	0	5.812

Şekil 6.4. Çok düğümlü kümede bizim parametrelerimizle sistem çalıştığında oluşan özet bilgiler

Tek ve çok düğümlü sistemin varsayılan parametrelerle ve bizim tespit ettiğimiz parametrelerle çalıştıklarında elde edile grafik Şekil 6.5. gibi olmaktadır.



Şekil 6.5. Deneye göre düğüm sayısına, varsayılan ve optimize parametre değerlerine için geçen işlem süreleri

6.3. Deney Ortamı Bileşenleri ve Deney İçin Hazırlanan Sorgular

6.3.1. Deney ortamı bileşenleri

Beş düğümden oluşmuş hadoop sistemimizde düğüm ile ifade ettiğimiz her bir bilgisayarlara aşağıdaki bahsedilen uygulamalar kurulmuştur.

- Hadoop-1.2.1.jar
- Jdk1.6.33
- Ubuntu12.04
- Intel Pentium 2.8 Ghz işlemci
- 2 gb ana bellek
- 150 gb 7200 rpm disk

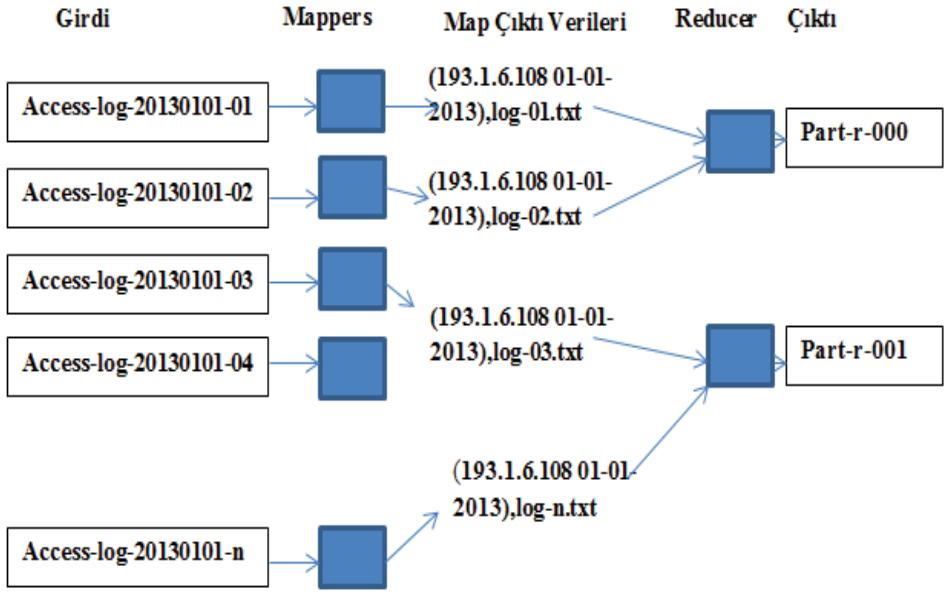
Test ortamımızda her biri yaklaşık 102 mb boyutunda Ocak 2013 ayına ait güvenlik duvarı logları oluşan 96 adet 10gb boyutunda bir test verimiz bulunmaktadır. Bu veri içerisinde alanlar '$' işareti ile ayrılmış sırasıyla "IP","TARİH","URL1","URL2","BROWSER" alanları yer almaktadır. Verinin işlenebilmesi için önce hdfs dosya sistemine kopyalanmıştır. Veriyi işleyecek olan test fonksiyonumuz ise Apache Pig Latin dilini kullanarak komut ekranında parametre olarak girilen "IP" ve "TARİH" değerlerine göre; belirtilen IP adresine sahip bilgisayarın belirtilen tarihlerde hangi web sitelerine girmiş olduğunu tespit etmektir.

6.3.2. Deney için hazırlanan sorgular

İstediğimiz IP = "193.1.6.180" ve TARİH= "01-01-2013"değerleri bu şekilde olduğunda betiğimiz aşağıdaki gibi olmaktadır.

```
A = LOAD '/input/3ok/' using PigStorage('$') AS
(IP:chararray, Date:chararray,url1:chararray,url2:chararray,browser:chararray);
data = FILTER A BY IP == '193.1.6.108' AND (Date matches '.*31/Dec/2012.*');
STORE data INTO "user/hduser/output"
```

Girdi verilerimiz 96 adet metin dosyasından oluştuğundan dolayı hadoop map reduce sistemi ana düğümü 96 adet map ve her biri için görev denetleyici oluşturmaktadır. Her bir görev denetleyicisinin amacı bulunduğu map bölümünde ilgili filtreleme işlemini gerçekleştirmektir. Her görev denetleyicinin yaptığı filtreleme işleminin sonucu tampon belleklere yazılır. Reducer toplama işlemini yaparken okuma işlemini buradan yapar. Sonuçlar metin dosyası şeklinde tekrar yerel diske yazılır. Şekil 6.6 bu filtreleme işlemini map reduce iş akışında daha iyi açıklanmaktadır.



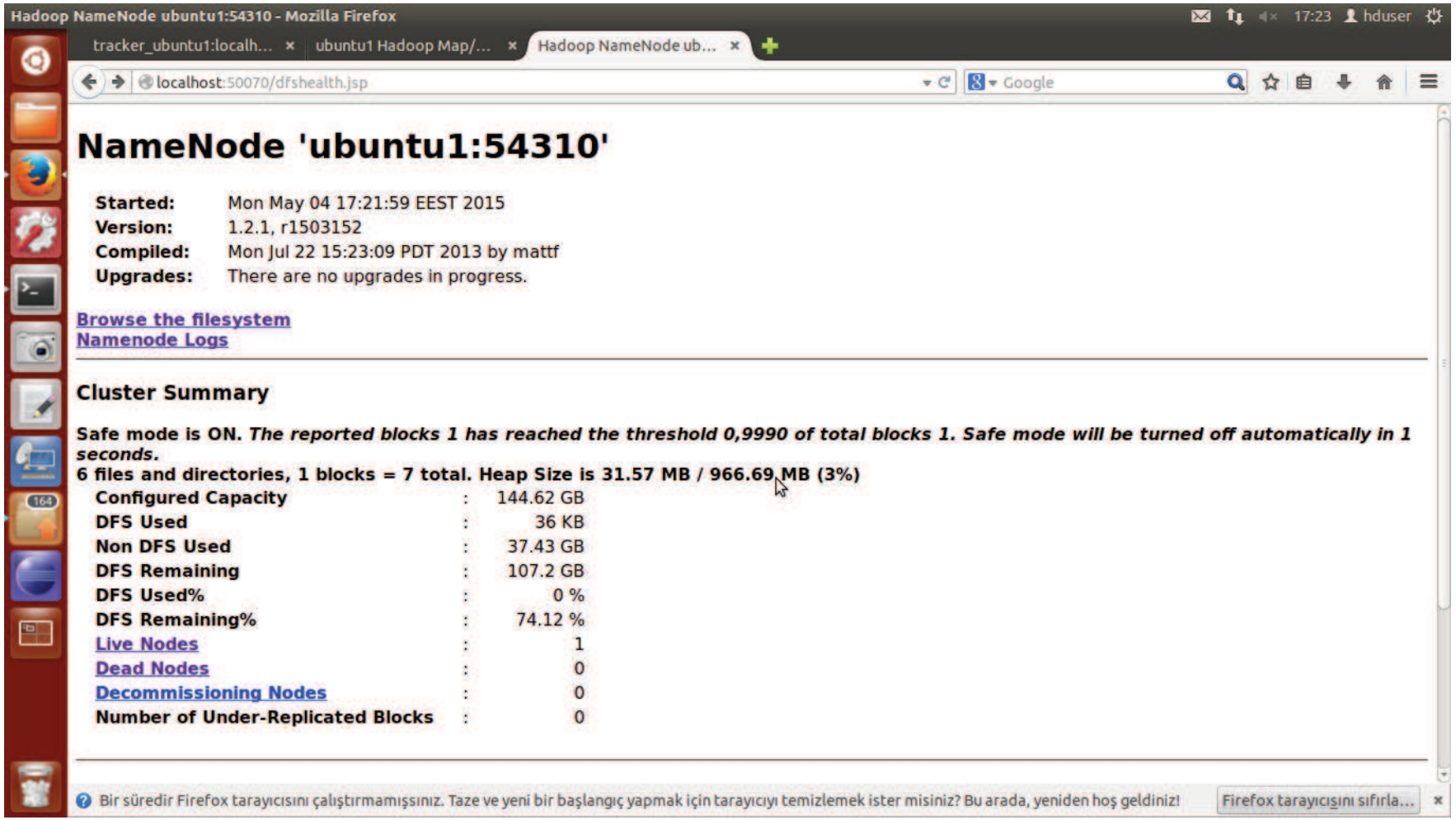
Şekil 6.6. Deneyde logların filtreleme işlemleri olan map ve reduce fonksiyonları üzerindeki dağılımı

BÖLÜM 7. SONUÇLAR

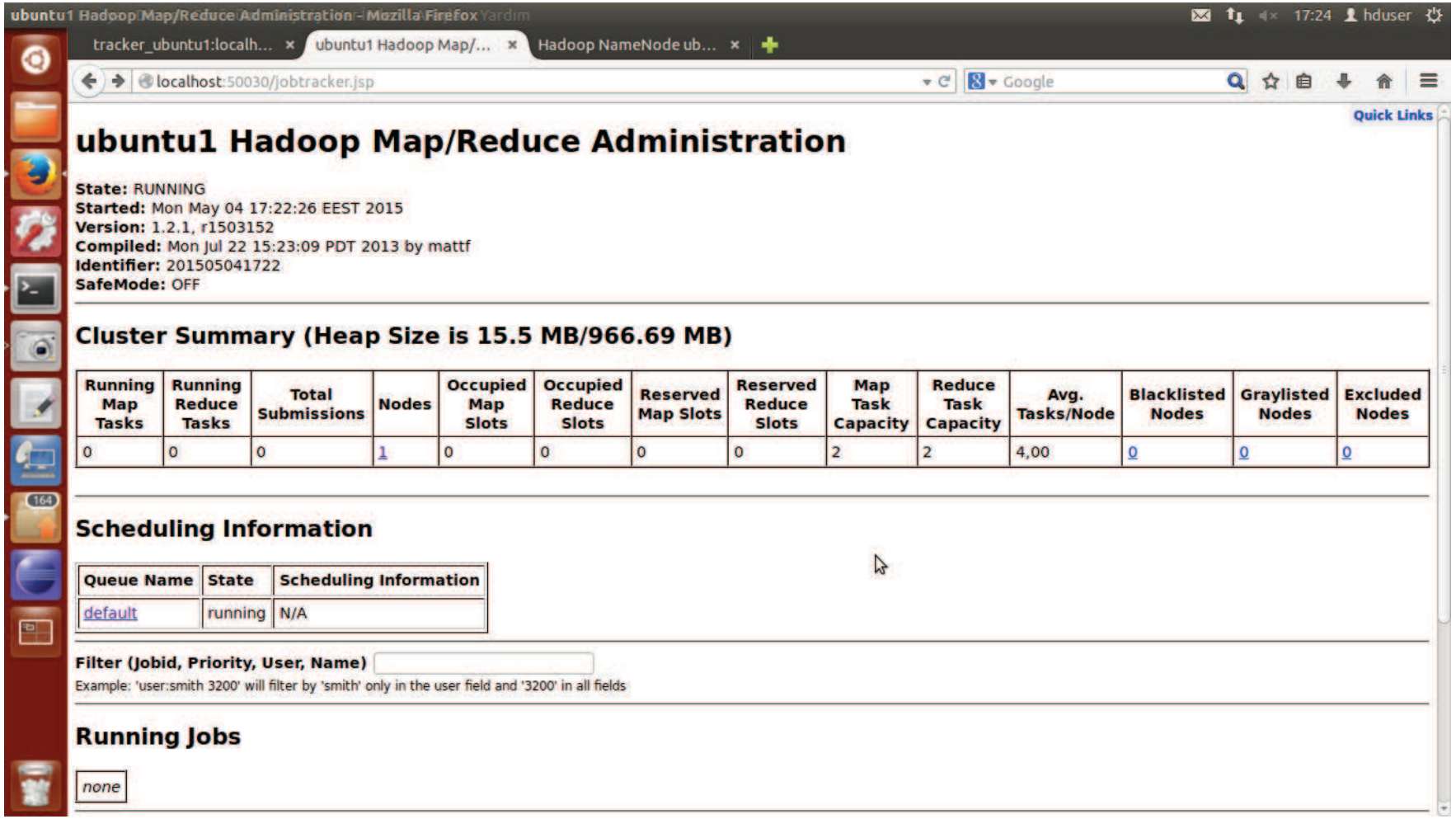
Hadoop map reduce algoritmasını kullanan bir dağıtık sistemde; donanım ekleyip maliyeti artırmadan, girdi verilerinize, yapacağınız analizin karmaşıklığına ve hadoop küme yapınızın özelliklerine göre bazı yapılandırma parametrelerini en uygun değerine getirerek hadoop sisteminin performansı artabileceği tespit edilmiştir.

Bu çalışmada en uygun değerleri, yaptığımız testler sonucunda elde edip parametre değerlerini dağıtık sistemimize manuel olarak işledik. Gelecekteki çalışmamız için dağıtık sistem katmanı üzerine bir uygulama ile en uygun değerlerin sistem özelliklerine -yapılandırılan dağıtık disk kapasitesi, girdi verisi, yapılacak analiz türü, her bir düğümdeki bellek miktarları, cpu özellikleri, vb.- göre tespit edilip otomatik olarak parametre değerlerinin değiştirilmesini sağlamaya çalışıyoruz.

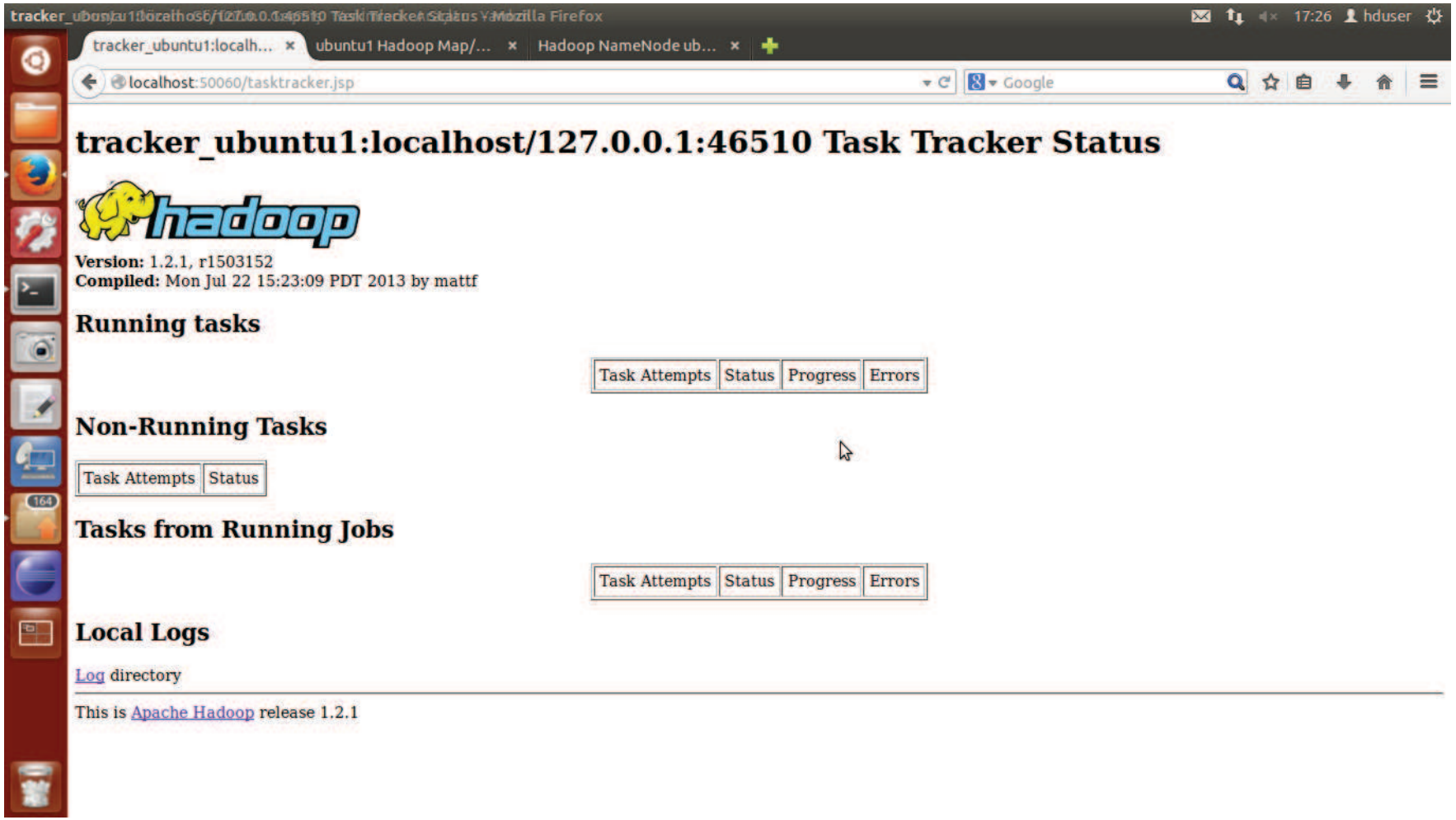
Ayrıca üzerinde çalıştığımız uygulamalardan bir diğeri de sadece bir sistemden gelen loglar yerine yerel veya internet ağında bulunan bir veya birden fazla makinenin (bilgisayar, sunucu, güç kaynağı, güvenlik duvarı, vs.) loglarına otomatik ulaşarak tüm sistemdeki cihazların bilgilerine ulaşıp bu cihazları hadoop dağıtık dosya sisteminde analizini hızlı ve verimli bir şekilde gerçekleştirmektir.



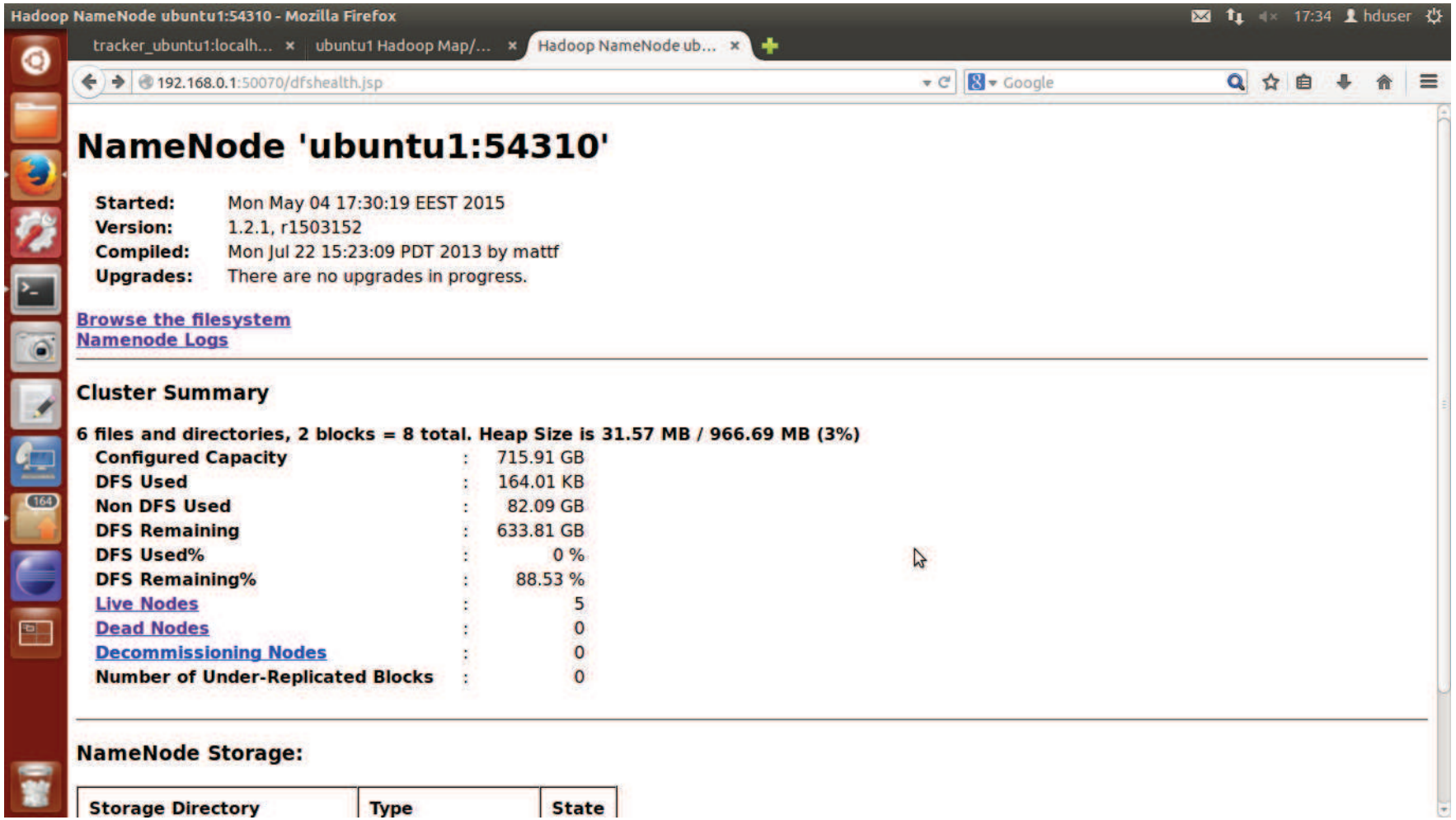
Şekil 5.1. Tek Düğümlü hadoop kümesi çalıştırıldıktan sonra düğümün genel sistem özelliklerini gösteren ekran görüntüsü



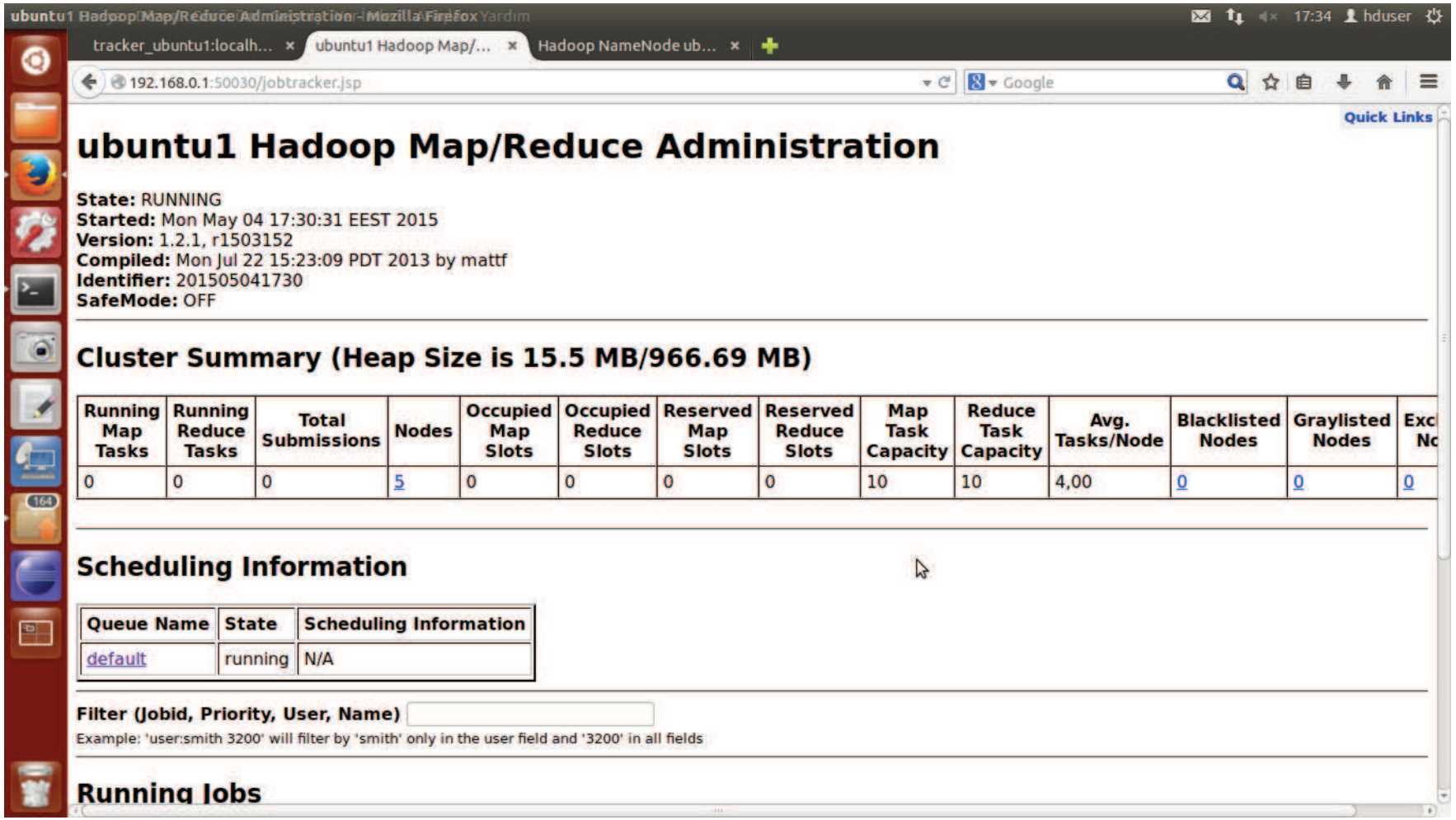
Şekil 5.2. Tek Düğümlü hadoop kümesi çalıştırıldıktan sonra düğümün genel sistem özelliklerini gösteren ekran görüntüsü



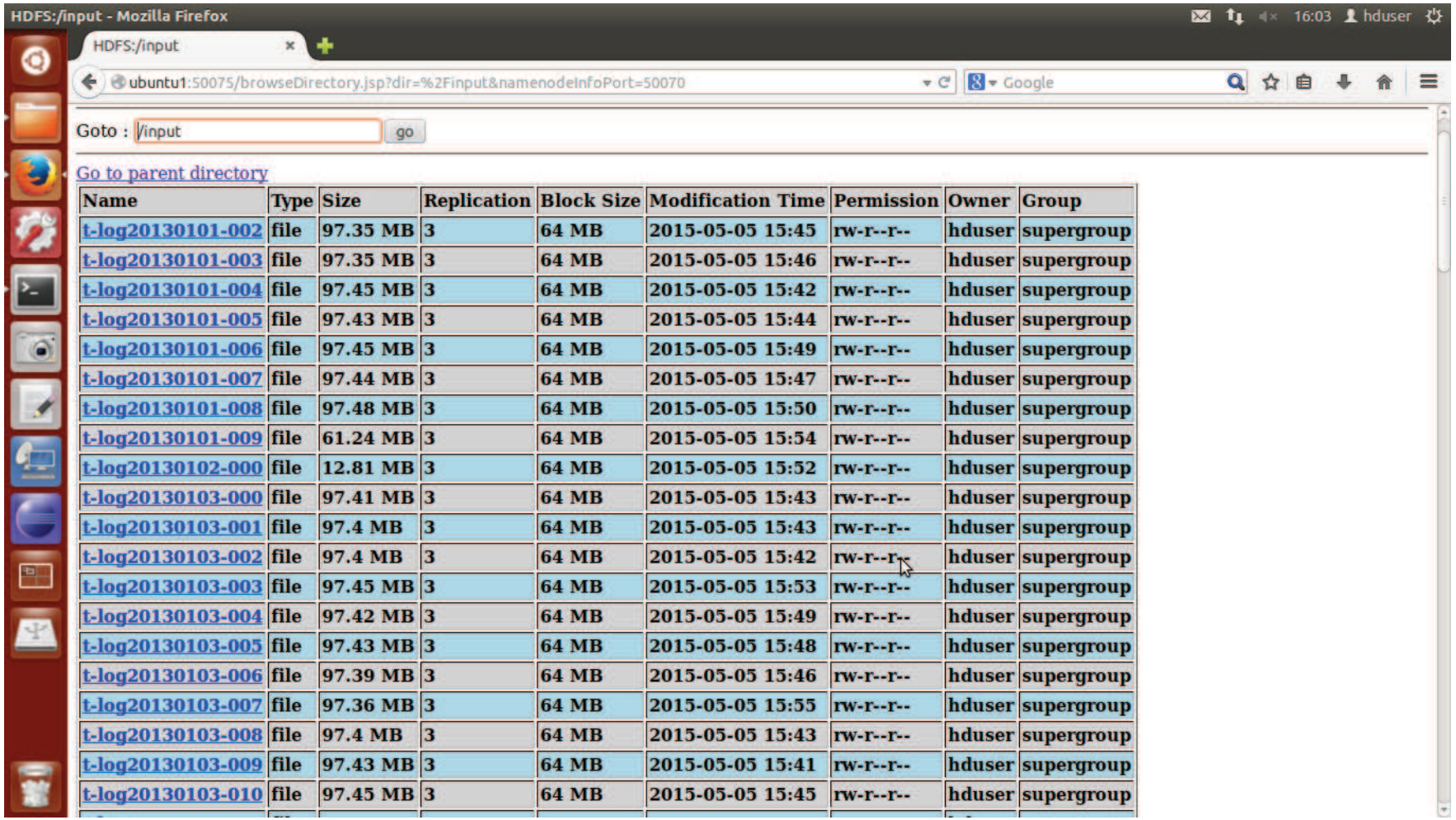
Şekil 5.3. Tek düğümlü hadoop kümesinde hadoop çalıştıktan sonra görev izleyicinin özet bilgilerini gösteren ekran görüntüsü



Şekil 5.4. Çok düğümlü hadoop kümesi çalıştırıldıktan sonra sistemin genel özet bilgilerini gösteren ekran görüntüsü

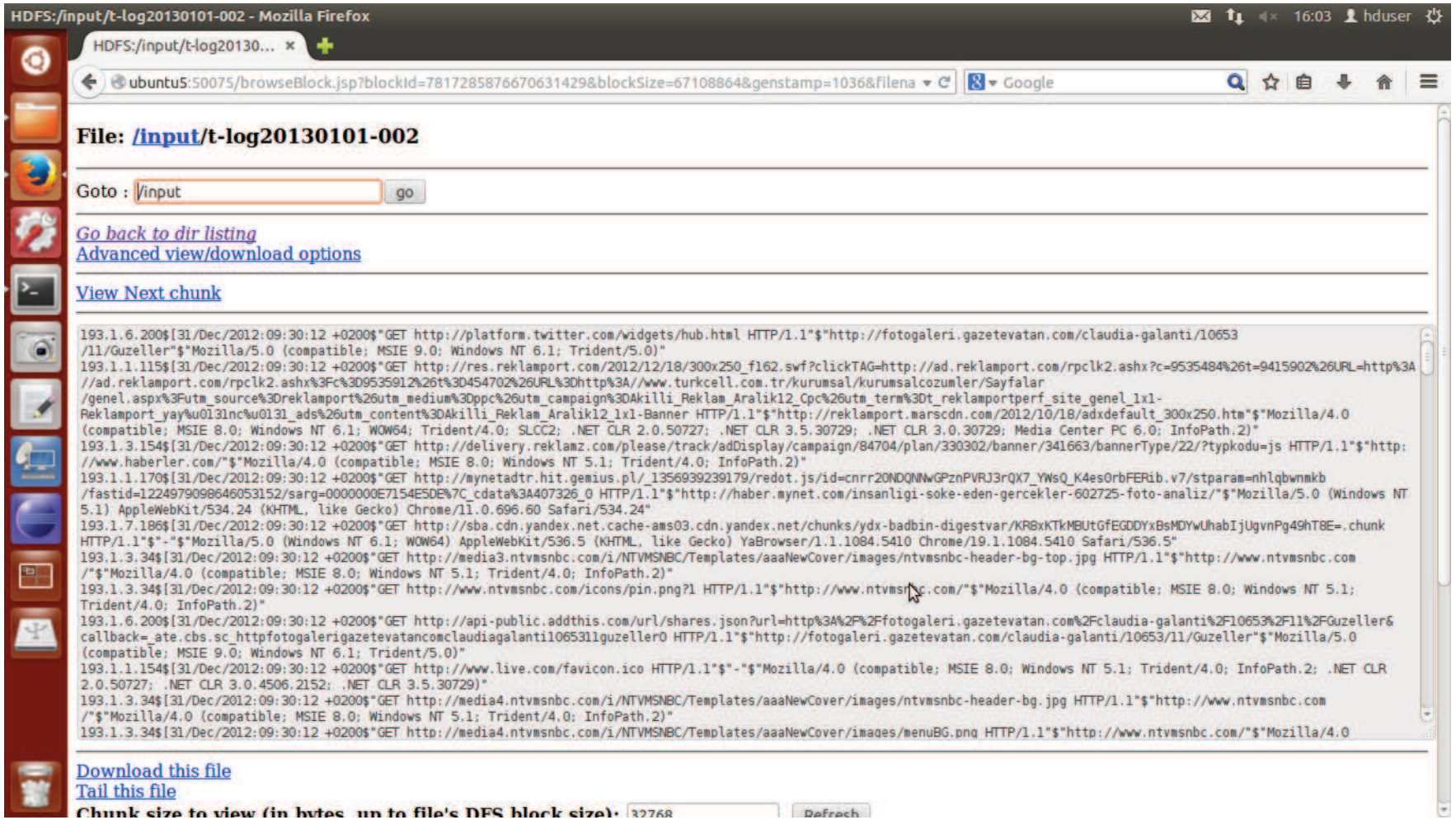


Şekil 5.5. Çok düğümlü hadoop kümesi çalıştırıldıktan sonra iş izleyicinin özet bilgilerini gösteren ekran görüntüsü



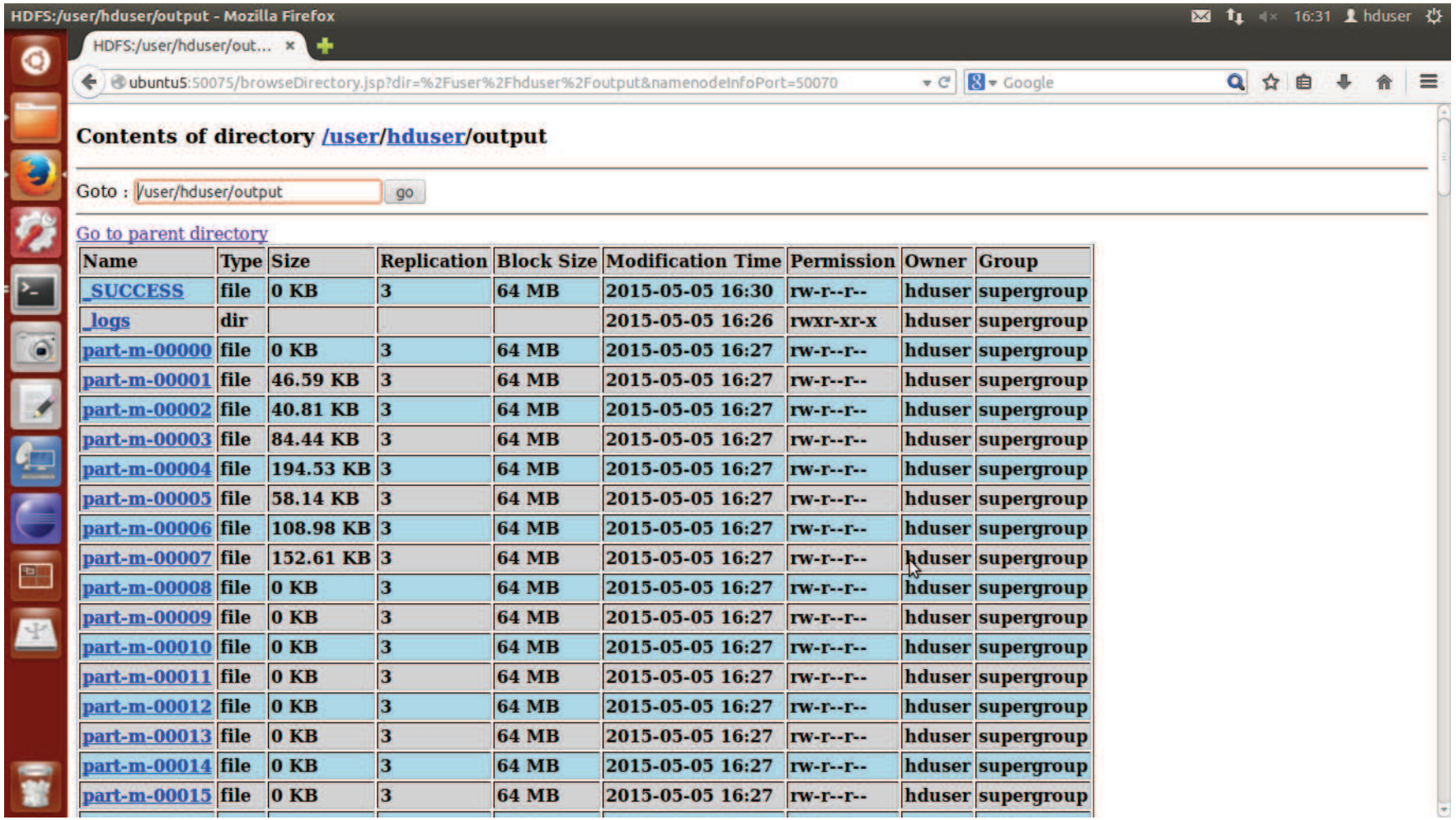
Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
t-log20130101-002	file	97.35 MB	3	64 MB	2015-05-05 15:45	rw-r--r--	hduser	supergroup
t-log20130101-003	file	97.35 MB	3	64 MB	2015-05-05 15:46	rw-r--r--	hduser	supergroup
t-log20130101-004	file	97.45 MB	3	64 MB	2015-05-05 15:42	rw-r--r--	hduser	supergroup
t-log20130101-005	file	97.43 MB	3	64 MB	2015-05-05 15:44	rw-r--r--	hduser	supergroup
t-log20130101-006	file	97.45 MB	3	64 MB	2015-05-05 15:49	rw-r--r--	hduser	supergroup
t-log20130101-007	file	97.44 MB	3	64 MB	2015-05-05 15:47	rw-r--r--	hduser	supergroup
t-log20130101-008	file	97.48 MB	3	64 MB	2015-05-05 15:50	rw-r--r--	hduser	supergroup
t-log20130101-009	file	61.24 MB	3	64 MB	2015-05-05 15:54	rw-r--r--	hduser	supergroup
t-log20130102-000	file	12.81 MB	3	64 MB	2015-05-05 15:52	rw-r--r--	hduser	supergroup
t-log20130103-000	file	97.41 MB	3	64 MB	2015-05-05 15:43	rw-r--r--	hduser	supergroup
t-log20130103-001	file	97.4 MB	3	64 MB	2015-05-05 15:43	rw-r--r--	hduser	supergroup
t-log20130103-002	file	97.4 MB	3	64 MB	2015-05-05 15:42	rw-r--r--	hduser	supergroup
t-log20130103-003	file	97.45 MB	3	64 MB	2015-05-05 15:53	rw-r--r--	hduser	supergroup
t-log20130103-004	file	97.42 MB	3	64 MB	2015-05-05 15:49	rw-r--r--	hduser	supergroup
t-log20130103-005	file	97.43 MB	3	64 MB	2015-05-05 15:48	rw-r--r--	hduser	supergroup
t-log20130103-006	file	97.39 MB	3	64 MB	2015-05-05 15:46	rw-r--r--	hduser	supergroup
t-log20130103-007	file	97.36 MB	3	64 MB	2015-05-05 15:55	rw-r--r--	hduser	supergroup
t-log20130103-008	file	97.4 MB	3	64 MB	2015-05-05 15:43	rw-r--r--	hduser	supergroup
t-log20130103-009	file	97.43 MB	3	64 MB	2015-05-05 15:41	rw-r--r--	hduser	supergroup
t-log20130103-010	file	97.45 MB	3	64 MB	2015-05-05 15:45	rw-r--r--	hduser	supergroup

Şekil 5.6. Çok düğümlü hadoop kümesi çalıştırılıp hdfs dosya sistemine varsayılan blok boyutu ile log dosyalarını attıktan sonraki ekran görüntüsü



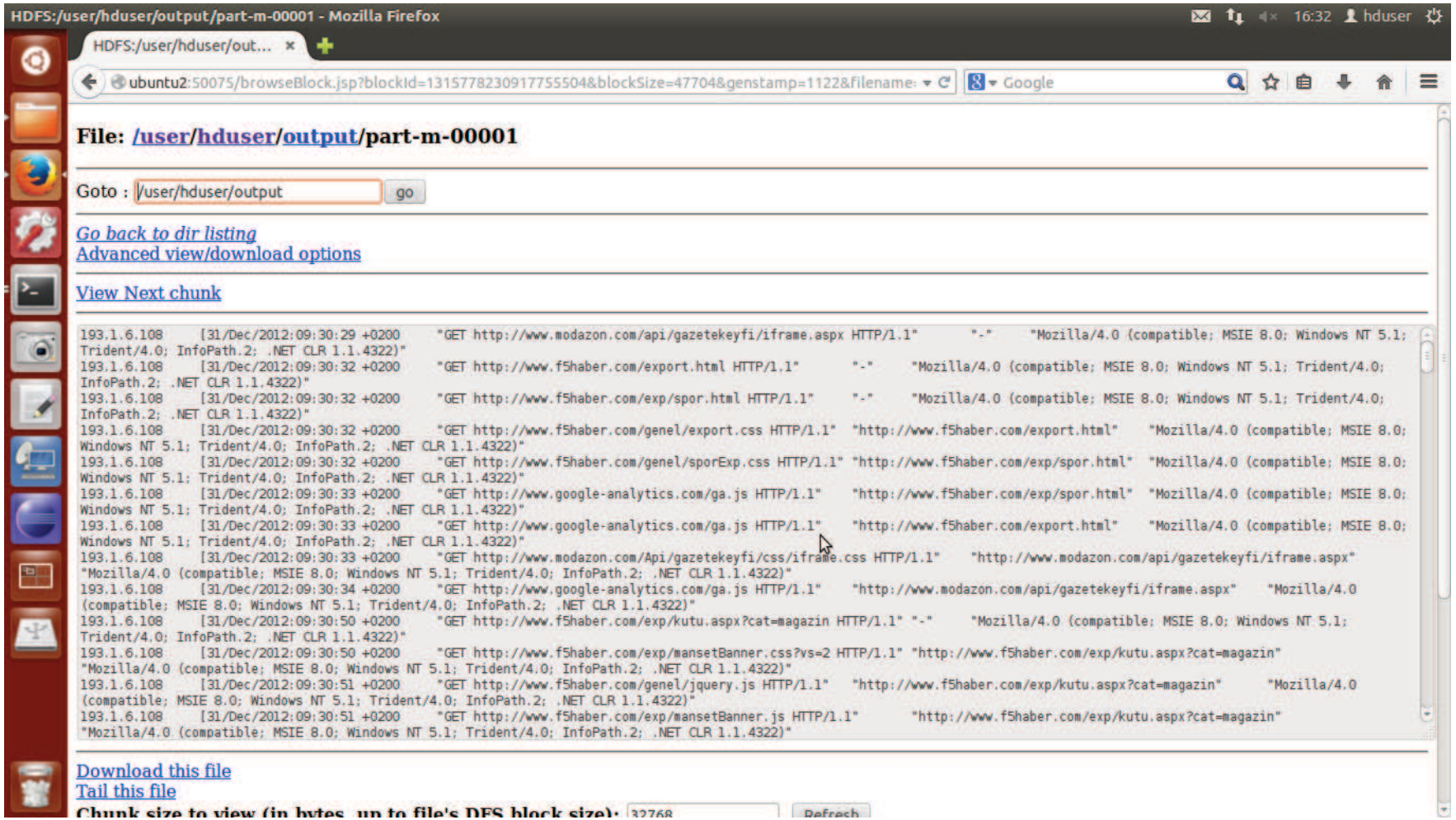
Şekil 5.7. Çok düğümlü hadoop kümesi çalıştırılıp hdfs dosya sistemine varsayılan blok boyutu ile log içeriğindeki alanların '$' işareti le ayrılmış ekran görüntüsü

Şekil 5.8. Çok düğümlü hadoop kümesi üzerinde pig betiği çalıştırıldıktan sonra alınan ekran görüntüsü



Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	3	64 MB	2015-05-05 16:30	rw-r--r--	hduser	supergroup
_logs	dir				2015-05-05 16:26	rwxr-xr-x	hduser	supergroup
part-m-00000	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00001	file	46.59 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00002	file	40.81 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00003	file	84.44 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00004	file	194.53 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00005	file	58.14 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00006	file	108.98 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00007	file	152.61 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00008	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00009	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00010	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00011	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00012	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00013	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00014	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup
part-m-00015	file	0 KB	3	64 MB	2015-05-05 16:27	rw-r--r--	hduser	supergroup

Şekil 5.9. Çok düğümlü hadoop kümesi üzerinde pig betiği çalıştırıldıktan sonra oluşan çıktı dosyalarını gösteren ekran görüntüsü



Şekil 5.10. Çok düğümlü hadoop kümesi üzerinde pig betiği çalıştırıldıktan sonra oluşan çıktı dosyalarının içeriğini gösteren ekran görüntüsü

KAYNAKLAR

[1] http://hadoop.apache.org, Erişim Tarihi: 10.05.2015.

[2] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In OSDI, 2004.

[3] I. Elghandour, A. Aboulnaga. Restore: Reusing results of mapreduce jobs. In VLDB, 2012.

[4] Impetus Hadoop Performance Tuning http://www.impetus.com, Erişim Tarihi : 10.05.2015.

[5] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava. Building a high-level dataflow system on top of MapReduce: the pig experience. In VLDB, 2009.

[6] H. Herodotou and S. Babu. Profiling, what-if analysis, and cost-based optimization of mapreduce programs. In VLDB,2011.

[7] H. Herodotou, F. Dong, and S. Babu. Mapreduce programming and cost-based optimization? crossing this chasm with starfish. In VLDB, 2011.

[8] F. N. Afrati and J. D. Ullman. Optimizing joins in a mapreduce environment. In EDBT, 2010.

[9] H. Herodotou, F. Dong, and S. Babu. Mapreduce programming and cost-based optimization? crossing this chasm with starfish. In VLDB, 2011.

[10] C. Lam Hadoop In Action. Apress, 1 edition, June 2011.

[11] J. Dean S. Ghemawat MapReduce:simplified data processing on large clusters Commun. ACM, 51 (1) (2008), pp. 107–113.

[12] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, Improving mapreduce performance in heterogeneous environments, in: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, OSDI, 2008, pp. 29–42.

[13] H.H. You, C.C. Yang, J.L Huang, A load-aware scheduler for MapReduce framework in heterogeneous cloud environments, in: Proceedings of the 2011 ACM Symposium on Applied Computing, 2011, pp. 127–132.

[14] S. Zhang, J. Han, Z. Liu, K. Wang, S. Feng, Accelerating MapReduce with distributed memory cache, in: 15th International Conference on Parallel and Distributed Systems, ICPADS, 2009, pp. 472–478.

[15] Y. Becerra Fontal, V. Beltran Querol, P, D. Carrera, et al. Speeding up distributed MapReduce applications using hardware accelerators, in: International Conference on Parallel Processing, ICPP, 2009, pp. 42–49.

[16] R. Nanduri, N. Maheshwari, A. Reddyraja, V. Varma, Job aware scheduling algorithm for MapReduce framework, in: 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom, 2011, pp. 724–729.

[17] M. L. Massie, B. N. Chun, D. E. Culler, The ganglia distributed monitoring system: Design, implementation, and experience, Parallel Computing 30 (2004) 817–840.

[18] http://devveri.com, Erişim Tarihi: 10.05.2015.

[19] J. Xie, et al. Improving MapReduce performance through data placement in heterogeneous Hadoop clusters, in: 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Ph.D. Forum, IPDPSW, 2010, pp. 1–9.

[20] Sharma, S., (n.d.), Advanced Hadoop Tunining and Optimizations, [online] http://www.slideshare.net/ImpetusInfo/ppt-on-advanced-hadoop-tuning-n-optimisation, Erişim Tarihi: 10.05.2015.

[21] C. He, Y. Lu, D. Swanson, Matchmaking: a new MapReduce scheduling technique, in: 3rd International Conference on Cloud Computing Technology and Science, CloudCom, 2011, pp 40–47.

[22] L. Massie, B. N. Chun, D. E. Culler, The ganglia distributed monitoring system: Design, implementation, and experience, Parallel Computing 30 (2004) 817–840.

[23] Y. Becerra Fontal, V. Beltran Querol, P, D. Carrera, et al. Speeding up distributed MapReduce applications using hardware accelerators, in: International Conference on Parallel Processing, ICPP, 2009, pp. 42–49.

ÖZGEÇMİŞ

Hüseyin Şarkışla, 30.04.1986 da Sivas'ta doğdu. İlk, orta ve lise eğitimini Sivas Merkez'de tamamladı. 2004 yılında Sivas Lisesi'nden mezun oldu. 2005 yılında Anadolu Üniversitesi Bilgisayar Mühendisliği Bölümü'nde eğitimine başladı. 2007 yılında dil eğitimi almak amacı ile WAT programı ile Amerika'nın Virginia eyaletinde 4 ay kaldı. 2008 yılında bir dönem boyunca Polonya Varşova Teknik Üniversitesi'nde eğitim gördü. 2010 Anadolu Üniversitesi'nden mezun oldu. Yaklaşık 1 yıl kadar Eskişehir'de FreeLancer olarak PHP programla dili ile Web Uygulamaları geliştirdi. 2011 yılında Türkiye Vagon Sanayi Anonim Şirketi'nde çalışmaya başladı. Şu anda Türkiye Vagon Sanayi Anonim Şirketi'nde Bilgisayar Mühendisi ve Yazılım Uzmanı olarak görev yapmaktadır.