MANAS Journal of Engineering MJEN 3 (1) 2015 11-20

# Hadoop Çatısının Bulut Ortamında Gerçeklenmesi ve Terabyte Sort Deneyleri

Göksu Zekiye Özen
Kırgızistan-Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bişkek, Kırgızistan
goksu@goksu.info

Rayımbek Sultanov
Kırgızistan-Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bişkek, Kırgızistan
rayimbek.sultanov@manas.edu.kg

*Received:* *03.02.2015* *Reviewed:* *01.03.2015* *Accepted:* *02.03.2015*

**Özet** Hadoop çatısı, büyük veriyi işlemede, işlenecek verinin düğüm öbekleri üzerinde dağıtılması, işlenmesi ve tekrar birleştirilerek anlamlı hale getirilmesi için MapReduce programlama paradigmasını kullanır. MapReduce, geniş bilgisayar öbekleri üzerinde barındırılan büyük verinin işlenmesinde kullanılan tekniklerden biridir. Bu yöntemde işler daha küçük parçalara ayrılır ve düğümlere dağıtılarak işlenir. Öbekteki düğüm sayısı işlerin tamamlanma süresine etki etmektedir. Bu çalışmanın amacı bulut ortamında gerçeklenen bir Hadoop öbeği üzerinde Hadoop çatısının başarımının öbekteki düğüm sayısından nasıl etkilendiğini çeşitli kıyaslama araçları yardımıyla tespit etmektir. Bu amaçla 10 düğümlü bir öbek üzerinde Hadoop çatısı kurularak TeraByte Sort kıyaslama araçları yardımıyla farklı düğüm sayıları kullanılarak deneyler gerçekleştirilmiştir. Deney sonuçlarına göre düğüm sayısının artırılmasının Hadoop çatısının iş bitirme başarımını artırdığı ve iş için harcanan zamanı azalttığı görülmüştür.

*Anahtar sözcükler* *Büyük veri, MapReduce, Hadoop.*

# Hadoop Framework Implementation on Cloud and Experiments of Terabyte Sort

**Abstract** Hadoop framework employs MapReduce programming paradigm to process big data by distributing data across a cluster and aggregating. MapReduce is one of the methods used to process big data hosted on large clusters. In this method, jobs are processed by dividing into small pieces and distributing over nodes. The number of nodes in the cluster affect the execution time of jobs. Main idea of this paper is to determine how number of nodes affect the performance of Hadoop framework on a cloud environment with using benchmarking tools. For this purpose, various tests are carried out on a Hadoop cluster with 10 nodes hosted on a cloud environment by running Terabyte Sort benchmarking tools on it. According to test results, increasing number of nodes improves job execution performance of Hadoop framework and reduces job execution time.

*Keywords* *Big data, MapReduce, Hadoop.*

## 1. GİRİŞ

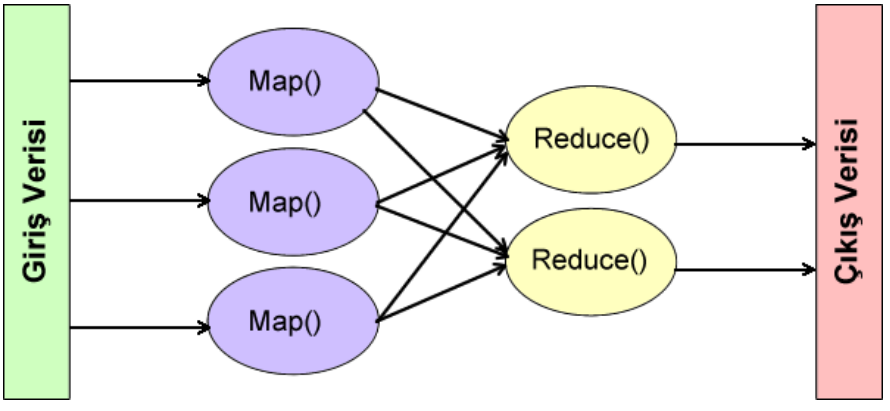
Farklı ortamlardan (internet siteleri, bloglar, akıllı telefonlar, tümleşik devreler v.b.) elde edilebilen yapısal olmayan veri, büyük veri olarak isimlendirilir. Büyük verinin anlamlı hale getirilebilmesi için geleneksel veri işleme ve saklama yöntemleri kullanılamamaktadır [1].

Büyük verinin ve büyük veri analiz araçlarının kullanım alanlarından bir tanesi karar destek sistemleridir [2]. Büyük veri işleme araçları olmadan geçmişe dönük veri analizi yapılabilirken, günümüzde dağıtık bilgisayar sistemlerinin hızı sayesinde geleceğe dönük anlık tahminler yapmak mümkün hale gelmiştir.

Günden güne artan bilgi hacmi, yoğunluğu ve çeşitliliği nedeniyle bazı organizasyonların günlük olarak işlemeleri gereken veri miktarı astronomik düzeye ulaşmıştır. Günümüzde bu verinin hızlı ve doğru bir biçimde işlenmesi, işlenen verilerden anlamlı sonuçlar üretilmesi ve depolanabilmesi önemli hale gelmiştir. Örneğin Facebook tarafından, günlük olarak işlenen veri miktarının yaklaşık 250 TeraByte olduğu rapor edilmiştir [3].

Büyük veri işlemede yaygın kullanılan Hadoop çatısı Apache Nutch projesinin içerisinde ölçeklenebilirlik sorununu gidermek için geliştirilmiştir. Apache Nutch, Google firması tarafından açık kaynaklı bir web arama motoru olarak Java dilinde yazılmıştır. Nutch projesi, paralel işleme ve hesaplama için MapReduce yöntemini ve dağıtık dosya sistemi GFS'yi (Google File System, Google Dosya Sistemi) kullanarak, Web adreslerini string olarak büyük tablolarda (big table) [4] tutarak, arama indekslerinden anlamlı sonuçlar çıkarmak için kullanılmıştır. Hadoop 2004 yılında bu projeden ayrılarak ayrı bir framework (çatı) olarak devam etmiştir [5].

Hadoop çatısı, master - slave mimarisine göre tasarlanmıştır. Hadoop işlenecek verinin düğüm öbekleri üzerinde dağıtılması, işlenmesi ve sonuçların birleştirilerek anlamlı hale getirilmesi için MapReduce programlama paradigmasını kullanır. Map aşamasında master düğüm verileri alıp küçük parçalara ayırarak slave düğümlere dağıtır ve reduce aşamasında ise tamamlanan işler birleştirilerek sonuç elde edilir. MapReduce yönteminde veri dağıtık ve paralel olarak işlenir [6]. Map ve Reduce işlemleri Şekil 1'de gösterilmektedir.

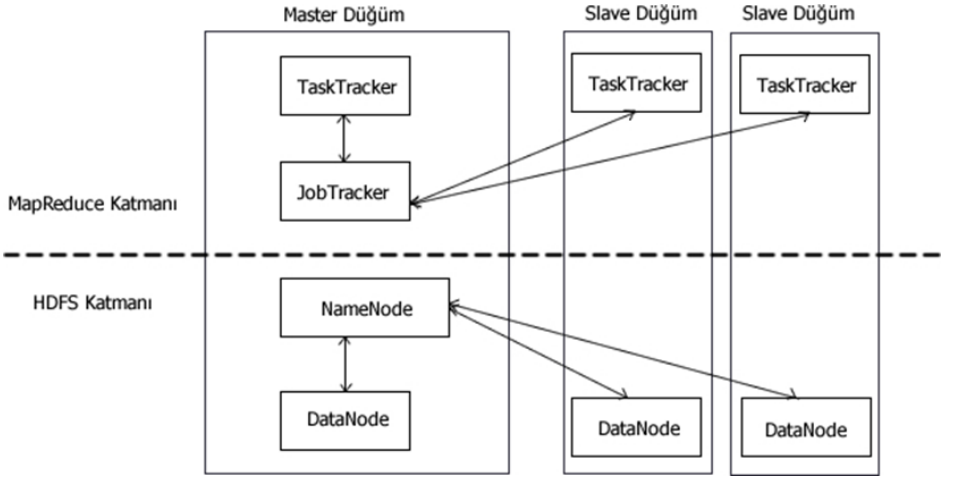



**Şekil 1**. MapReduce yöntemi.

Hadoop çatısı, Hadoop Dağıtık Dosya Sistemini (Hadoop Distributed File System, HDFS) kullanır. HDFS Java dilinde yazılmış açık kaynaklı, dağıtık sistemlerde çalışabilecek şekilde tasarlanmıştır, ucuz donanımlar üzerinde büyük veri setleriyle çalışabilir. HDFS; DataNode ve NameNode bileşenlerinden oluşur. Bir Hadoop öbeği üzerindeki ana düğüm hem NameNode hem de DataNode olarak görev yapabilmekte iken işçi düğümler sadece DataNode olarak görev yaparlar. NameNode

veri bloklarının oluşturulması, oluşturulan veri bloklarının diğer düğümlere dağıtılması, yönetilmesi ve bir veri bloğunda hata meydana geldiğinde bu veri bloğunun yeniden oluşturulması işlemlerinden, DataNode ise NameNode'den gelen talimatlarla veri bloklarının saklanmasından, silinmesinden ve yedeklenmesinden sorumludur. DataNode'ler periyodik olarak NameNode'ye heartbeat (kalp atışı) sinyalleri gönderirler. Bu sinyaller sayesinde NameNode, DataNode'nin çalışıp çalışmadığını kontrol eder. Herhangi bir hata meydana geldiğinde NameNode sinyal alamadığı DataNode'yi ölü olarak işaretler ve o DataNode'ye veri gönderimini, yedekleme ve silme talimatlarını durdurur. Ölü sayılan DataNode üzerindeki veriler NameNode tarafından tespit edilir ve öbekte sağlam çalışan diğer DataNode'ler üzerinde yedeklenerek işleme devam edilir. Hadoop çatısında NameNode düğümünün devre dışı kalması tüm sistemin başarısızlığına neden olur. NameNode başka bir düğümde otomatik olarak başlatılamaz veya yedeklenemez. Bu durumda sisteme elle müdahale edilmesi gerekir [7].

Hadoop çatısında (sürüm 1.x) MapReduce; JobTracker ve TaskTracker bileşenlerine sahiptir. JobTracker master düğüm üzerinde yer alan ve MapReduce görevlerini yerine getiren özel bir uygulamadır. JobTracker MapReduce işlerinin TaskTracker'lere dağıtılması, izlenmesi ve yönetilmesinden sorumludur. Tüm düğümler üzerinde çalışan TaskTracker, JobTracker'den gelen iş emirlerini kabul eder ve MapReduce işlerini çalıştırarak (Map ve Reduce aşamaları) tamamlanan işleri JobTracker'e bildirir [8]. Şekil 2'de Hadoop (sürüm 1.x) çatısının mimarisi gösterilmiştir.




**Şekil 2.** Hadoop (sürüm 1.x) Mimarisi.

Hadoop çatısı geniş bilgisayar öbeklerine kurulabilir ve ölçeklenebilir. Öbekteki düğüm sayısı arttıkça yönetimi zorlaşacağından dolayı kurulum, dağıtım, çizelgeleme ve etkin kaynak yönetimi için sanallaştırma çözümleri geliştirilmiştir. KVM (Kernel Based Virtual Machine, Çekirdek Tabanlı Sanal Makine) açık kaynaklı sanallaştırma çözümlerinden biridir [9].

Hadoop çatısı altında büyük veriyi işlemede ve test etmede kullanılan çeşitli MapReduce kıyaslama araçları sunulmaktadır [10]. Bu kıyaslama araçları ile veri setleri oluşturulabilmektedir veya hazır veri setleri kullanılabilmektedir. Bu veri setleri üzerinde arama, sözcük sayma, sıralama ve doğrulama gibi sık kullanılan büyük veri işlemleri gerçekleştirilebilmektedir.

Alan yazında Hadoop çatısının başarımı ile ilgili çeşitli çalışmalar yer almaktadır. Rizvandi ve arkadaşları tarafından yapılan çalışmada, yaygın MapReduce uygulamaları olan wordcount, grep ve terasort uygulamalarının farklı veri boyutu ve düğüm sayısı kullanılarak işin başarımını etkileyen faktörler araştırılmıştır [11]. Kim ve arkadaşlarının yaptığı çalışmada MRBench isimli bir kıyaslama

uygulaması tasarımı ve gerçeklemesi sunulmaktadır. MRBench uygulaması veri boyutu ve düğüm sayısının MapReduce sistemlerinin başarımına etkisini ölçmektedir [12]. Tan ve arkadaşlarının yaptıkları çalışmada Hadoop çatısının yapılandırma parametreleri ayarlanarak daha yüksek çalışma başarımı elde edebilmenin mümkün olduğu ortaya konmuş ve iş çalıştırma zamanı için bir analitik model olarak sunulmuştur [13]. O'Malley tarafından, Hadoop çatısı kullanılarak 910 düğümlü bir öbek üzerinde teragen kıyaslama aracı ile 1 TB (TeraByte) veri üretilmiş ve bu veri terasort kıyaslama aracı ile 209 sn içinde sıralanmıştır [14]. Maurya ve Mahajan tarafından sunulan çalışmada pi, wordcount, grep ve terasort gibi MapReduce kıyaslama araçlarının bir sunucuda 4 düğümlü bir Hadoop öbeği üzerinde çalıştırılarak düğüm sayısının çalışma süresine etkisi incelenmiştir. Her bir kıyaslama aracı için düğüm sayısı arttıkça, iş için harcanan zamanın azaldığı görülmüştür [15].

Bu makalede sunulan çalışma bulut ortamında gerçekleştirilmiştir. Ayrıca öbekteki düğüm sayısı 1'den 10'a kadar artırılmıştır. Hadoop çatısının varsayılan ayarları kullanılmış ve performans artışına sebep olacak herhangi bir parametre ayarı yapılmamıştır. TeraByte Sort kıyaslama araçlarıyla farklı sayıda düğüm üzerinde çeşitli deneyler gerçekleştirilmiş ve bulut üzerinde Hadoop çatısının başarımı araştırılmıştır.

Sunulan çalışmada kullanılan TeraByte Sort; teragen, terasort ve teravalidate kıyaslama araçlarından oluşmaktadır. Teragen kıyaslama aracı istenilen boyutta rastgele veri üreterek sonucu çıkış dizinindeki bir dosyaya yazar. Terasort giriş dizinindeki dosya ya da dosyalardaki verileri sıralayarak çıkış dizinindeki bir dosyaya yazar. Teravalidate kıyaslama aracı ise, terasort kıyaslama aracı ile sıralanan verilerin doğruluğunu kontrol eder [16].

## 2. MATERYAL ve METOT

Bu çalışmada teragen, terasort ve teravalidate kıyaslama araçları Hadoop çatısının temel ayarlarıyla, aynı deney düzenekleri üzerinde farklı düğüm sayısı üzerinde ayrı ayrı test edilerek iş için harcanan süreler kaydedilmiştir. Her bir deney 5 defa tekrarlanarak ortalama sonuçları alınmıştır.

### 2.1. Deney Ortamının Kurulması

Çalışmadaki bütün deneyler DigitalOcean [17] bulut sistemi üzerinde 10 adet sanal makina (düğüm) oluşturularak gerçekleştirilmiştir. Her bir düğümde kullanılan sanal makinalar Intel Hex-Core, 2.6 GHz (GigaHertz) tek çekirdekli işlemci, 1 GB (GigaByte) bellek ve 30 GB Katı Durum Sürücü (Solid State Drive, SSD) içermektedir. İşletim sistemi olarak Ubuntu 10.04 Uzun Vadeli Destek (Long Term Support, LTS) kullanılmıştır. Sanallaştırma işlemleri KVM sanallaştırma yazılımı üzerinde yapılmıştır. Her bir sanal makinaya ilk olarak Java Geliştirme Kiti'nin (Java Development Kit, JDK), jdk 1.6.0_45 sürümü kurulup ayarlamalar yapılmıştır. Ardından Hadoop 1.0.3 kurularak, Hadoop için gerekli ayarlamalar yapılmıştır. Bir düğüm ana düğüm olarak, geri kalan diğer 9 düğüm ise işçi düğüm olarak ayarlanmıştır. Düğümlere erişim Güvenli Kabuk (Secure Shell, SSH) sunucu yazılımıyla gerçekleştirilmiştir. Hadoop çatısında işlenen verinin türüne ve kıyaslama araçlarına göre işin başarımını artırmaya yönelik parametre ayarları yapılabilmektedir [18]. Tüm düğümlerde Hadoop konfigürasyon dizini altında yer alan core-site.xml, mapred-site.xml ve hdfs-site.xml dosyaları Hadoop çatısının çoklu düğüm üzerinde çalışması için gerekli temel ayarları içeren parametreler kullanılarak Tablo 1, Tablo 2 ve Tablo 3'te görüldüğü gibi yapılandırılmıştır.

**Tablo 1.** core-site.xml ayarları

| Parametre | Değeri | Açıklama |
|---|---|---|
| fs.default.name | hdfs://hostname/ | NameNode'nin kaynak tanımlayıcısı. |
| hadoop.tmp.dir | /tmp/hadoop-${kullanici.adi} | Verilerin tutulacağı geçici dizinin konumu. |

**Tablo 2.** mapred-site.xml ayarları

| Parametre | Değeri | Açıklama |
|---|---|---|
| mapred.job.tracker | host:port | MapReduce JobTracker uygulamasının çalıştığı düğüm adı ve port numarası. |
| mapred.local.dir | ${hadoop.tmp.dir}/mapred/local | MapReduce ara veri dosyalarının tutulduğu yerel dizin. |
| mapred.system.dir | ${hadoop.tmp.dir}/mapred/system | MapReduce kontrol dosyalarının tutulduğu dizin. |

**Tablo 3.** hdfs-site.xml ayarları

| Parametre | Değeri | Açıklama |
|---|---|---|
| dfs.replication | 3 | Varsayılan blok yedekleme sayısı. |
| dfs.data.dir | ${hadoop.tmp.dir}/dfs/data | DataNode'nin yerel dosya sistemi üzerinde yeri. |
| dfs.name.dir | ${hadoop.tmp.dir}/dfs/name | NameNode'nin yerel dosya sistemi üzerindeki yeri. |

## 2.2. Deneylerin Gerçekleştirilmesi

hadoop-1.0.3-examples.jar dosyası java dilinde yazılıp derlenmiş çalıştırılabilir bir dosyadır. Bu çalıştırılabilir dosyada çeşitli kıyaslama araçları yer almaktadır. Bunlar wordcount, TeraByte Sort, grep, PiEstimator v.b. kıyaslama araçlarıdır. Deneylerde, veri setini üretebilmesi ve üzerinde işlem yapılabilmesi için TeraByte Sort kıyaslama araçları tercih edilmiştir. Teragen kıyaslama aracıyla rastgele veri üretmek için master düğüm üzerinde aşağıdaki komut çalıştırılmıştır. İlk parametre her bir satır için 100 byte boyutunda rastgele veri üretilecek şekilde satır sayısını belirtir. Teragen kıyaslama aracı kullanılarak sırasıyla 1, 2, 5 ve 10 düğüm üzerinde 10 GB boyutundaki rastgele veri üretilerek HDFS dosya sistemindeki bir çıkış dizinine yazılmıştır.

*$ hadoop jar hadoop-examples-1.0.3.jar teragen <100 byte'lik satırların sayısı> <çıkış-dizini>*

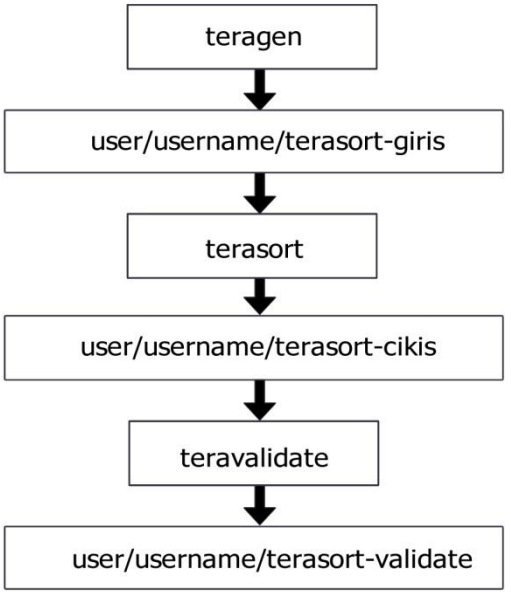
Ardından teragen çıkış dizini, terasort için giriş dizini olarak ayarlanarak 10 GB boyutundaki rastgele üretilmiş veri, yine 1, 2, 5 ve 10 düğüm üzerinde terasort kıyaslama aracı ile sıralanarak HDFS dosya sistemindeki bir çıkış dizinindeki dosyaya yazılmıştır. Terasort kıyaslama aracı için aşağıdaki komut, master düğüm üzerinde komut satırından çalıştırılmıştır.

*$ hadoop jar hadoop-examples-1.0.3.jar terasort <teragen çıkış-dizini (= giriş-dizini)> <çıkış-dizini>*

Terasort kıyaslama aracının ardından her defasında işin doğruluğunu test etmek için teravalidate kıyaslama aracı çalıştırılmıştır. Terasort çıkış dizini, teravalidate için giriş dizini olarak ayarlanarak verilerin doğruluğu sınanmıştır. Teravalidate kıyaslama aracı için aşağıdaki komut, master düğüm üzerinde komut satırından çalıştırılmıştır.

*$ hadoop jar hadoop-examples-1.0.3.jar teravalidate <terasort çıkış-dizini (= giriş-dizini)> <teravalidate çıkış-dizini>*

TeraByte Sort kıyaslama araçlarının çalıştırılma adımları Şekil 3'de görselleştirilmiştir.



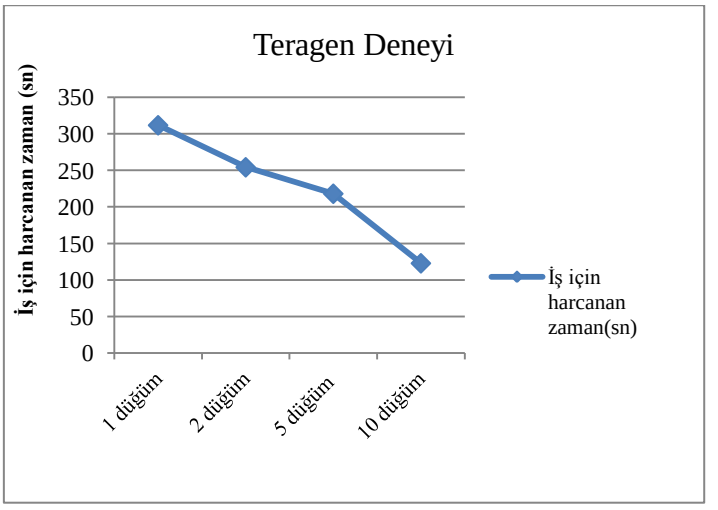

**Şekil 3.** TeraByte Sort kıyaslama araçları çalıştırılma adımları.

## 4. BULGULAR VE TARTIŞMA

Teragen kıyaslama aracı, farklı düğüm sayısı kullanılarak gerçekleştirilen deneylerde iş için harcanan zaman Tablo 4'de gösterilmektedir. İşin tamamlanma zamanı Şekil 4'de görselleştirilmiştir. Tablo 4 verileri incelendiğinde iş için harcanan zamanının 1 düğüm sayısı seçildiğinde 311,755 sn iken, %60 verimlilik artışı ile 10 düğüm sayısı seçildiğinde 123,035 sn olduğu görülmektedir. Şekil 4 verileri incelendiğinde düğüm sayısı artıkça iş için harcanan zamanın azaldığı ancak bu azalmanın kararlı bir azalma olmadığı görülmektedir.

**Tablo 4.** Farklı düğüm sayısı kullanılarak gerçekleştirilen teragen deneylerinde iş için harcanan zaman.

| Düğüm Sayısı | İş İçin Harcanan Zaman (sn) |
|---|---|
| 1 | 311,755 |
| 2 | 254,292 |
| 5 | 218,111 |
| 10 | 123,035 |

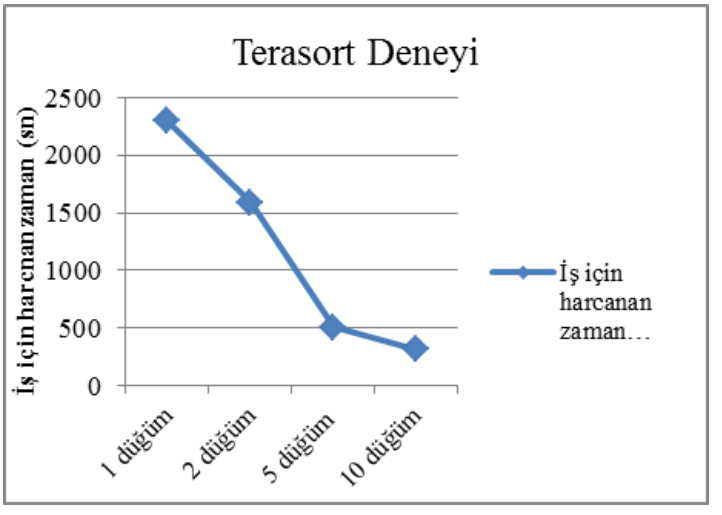



**Şekil 4.** Farklı düğüm sayısında gerçekleştirilen teragen deneyi iş için harcanan zaman.

Terasort kıyaslama aracı, farklı düğüm sayısı kullanılarak gerçekleştirilen deneylerde iş için harcanan zaman Tablo 5'de gösterilmektedir. İşin tamamlanma zamanı Şekil 5'de görselleştirilmiştir. Tablo 5 verileri incelendiğinde iş için harcanan zamanının 1 düğüm sayısı seçildiğinde 2299.372 sn iken, %86 verimlilik artışı ile 10 düğüm sayısı seçildiğinde 318.537 sn olduğu görülmüştür. Şekil 5 verileri incelendiğinde düğüm sayısı artıkça iş için harcanan zamanın azaldığı ancak bu azalmanın kararlı bir azalma olmadığı görülmektedir.

**Tablo 5.** Farklı düğüm sayısı kullanılarak gerçekleştirilen terasort kıyaslama aracı deneyi sonuçları.

| Düğüm Sayısı | İş İçin Harcanan Zaman (sn) |
|---|---|
| 1 | 2299.372 |
| 2 | 1590.587 |
| 5 | 504.177 |
| 10 | 318.537 |

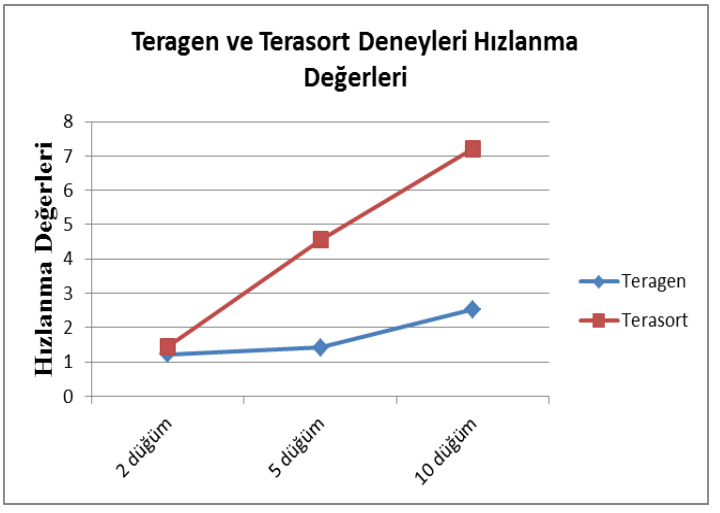



**Şekil 5.** Farklı düğüm sayısında gerçekleştirilen terasort deneyi iş için harcanan zaman.

Hızlanma değeri (speedup) bir algoritmanın verimliliğini ölçmede kullanılan yaygın bir ölçüttür [19]. Kullandığımız hızlanma değeri formülü Eşitlik 1'de verilmiştir.

$$Hızlanma\ Değeri = \frac{T_a}{T_n} \tag{1}$$

Eşitlik 1'de $T_a$ bir düğüm ya da tek işlemcide işin tamamlanma zamanını, $T_n$ ise n düğüm ya da n işlemcide işin tamamlanma zamanını göstermektedir. 10 düğümlü bir öbek üzerinde sırasıyla 1, 2, 5 ve 10 düğüm kullanılarak gerçekleştirilen teragen ve terasort deneylerinin sonuçlarından elde edilen hızlanma değerleri Şekil 6'da gösterilmiştir.




**Şekil 6.** Teragen ve terasort deneyleri hızlanma değerleri

Şekil 6 verileri incelendiğinde öbekteki düğüm sayısının artırılmasının terasort kıyaslama aracı hızlanma değerini doğrusal olarak arttırdığı, ancak teragen kıyaslama aracının hızlanma değerini doğrusal olarak arttırmadığı görülmüştür. Bu oran terasort deneyinde 1 ile 10 düğüm arasında yaklaşık 7 kat iken, teragen deneyinde yaklaşık 2,5 kat şeklindedir. Hadoop çatısının varsaylan ayarları ile Teragen aracının düğüm sayısı arttıkça iş parçacığı sayısını arttırmadığı ve işin bütün düğümler üzerinde yeterince paralelleştirilemediği görülmektedir.

## 5. SONUÇLAR

Bu çalışmada; Hadoop çatısı bulut ortamında, KVM sanallaştırma yazılımı kullanılarak 10 düğümlü bir öbekte kurulmuştur. Düğümlerin tamamı Hadoop çatısının varsayılan ayarları kullanılarak herhangi bir performans artışına yol açmayacak şekilde ayarlanmıştır. Bu düğümler üzerinde Hadoop çatısı altında gelen TeraByte Sort kıyaslama araçları yardımıyla farklı sayıda düğüm üzerinde veri setleri oluşturularak deneyler gerçekleştirilmiştir. Her bir deneyde iş için harcanan zaman kaydedilmiştir. Deneylerde düğüm sayısının işin başarımına nasıl etki ettiği incelenmiştir. Deneylerin sonuçlarına göre düğüm sayısının artmasının iş için harcanan zamanı azalttığı ve Hadoop çatısının iş bitirme başarımını artırdığı görülmüştür. Bu başarım artışı 1 ile 10 düğüm arasında; teragen deneyinde %60 iken terasort deneyinde %86 olarak bulunmuştur. Hadoop çatısının varsayılan ayarları ile düğüm sayısı arttıkça Terasort kıyaslama aracının hızlanma değerlerinin yaklaşık olarak doğrusal arttığı, Teragen kıyaslama aracının hızlanma

değerlerinin ise doğrusal olarak artmadığı görülmüştür. Hadoop çatısının başarımını artırmak için kullanılan kıyaslama aracına göre başka parametreler de bulunmaktadır. Teragen kıyaslama aracının başarımını artıracak başka parametre ayarları da yapılarak gelecek çalışmalarda bu uygulamanın etkinliği yeniden test edilebilir. Gelecek çalışmalarda düğüm sayısı değişimi ile birlikte çeşitli parametre değişimleri beraber kullanıldığında işlerin başarımını artırmanın mümkün olacağı değerlendirilmektedir.

## KAYNAKLAR


[1] Hurwitz J, Nugent A, Halper F, Kaufman M. Big Data For Dummies. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2013.

[2] Ishii M; Jungkyu H; Makino H. Design and performance evaluation for Hadoop clusters on virtualized environment. In: 2013 International Conference on Information Networking; 28-30 January 2013; Bangkok, Thailand. New York, NY, USA: IEEE. pp.244-249.

[3] Lublinsky B, Smith KT, Yakubovich A. Professional Hadoop Solutions. Indianapolis, Indiana, USA: John Wiley & Sons, Inc., 2013.

[4] Chang F, Dean J, Ghemawat S. Bigtable: A distributed storage system for structured data. In: 7th Usenix Symposium on Operating Systems Design and Implementation; 06-08 November 2006; Seattle, WA, USA. Berkeley, CA, USA: USENIX Assoc. pp.205-218.

[5] Holmes A. Hadoop in Practice. Shelter Island, NY: Manning Publications, 2012.

[6] Perera S, Gunarathne T. Hadoop MapReduce Cookbook. Birmingham, UK: Packt Publishing, 2013.

[7] HDFS Architecture Guide, http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, erişim tarihi 06.09.2014.

[8] Apache Hadoop, http://wiki.apache.org/hadoop, erişim tarihi 08.09.2014.

[9] KVM, Kernel based virtual machine, http://www.linux-kvm.org, erişim tarihi 10 09 2014.

[10] S. Guo, Hadoop Operations and Cluster Management Cookbook, Birmingham, UK, Packt Publishing, 2013.

[11] Rizvandi NB, Taheri J, Moraveji R, Zomaya AY. A study on using uncertain time series matching algorithms for MapReduce applications. Concurrency and Computation: Practice and Experience August 2013; 25: 1699–1718.

[12] Kim K, Jeon K, Han H, Kim SG. Mrbench : A benchmark for map-reduce framework. In: 14th International Conference on Parallel and Distributed Systems; 8-10 December 2008; Melbourne, Australia. Los Alamitos, CA, USA:IEEE.

[13] Tan YS, Tan J, Chng ES. Hadoop framework: Impact of data organization on performance. Wiley Online Library 2011; 43: 1241-1260.

[14] O'Malley O. Terabyte sort on apache hadoop. http:// sortbenchmark.org/Yahoo-Hadoop.pdf, May 2008.

[15] Maurya M, Mahajan S. Performance analysis of mapreduce programs on hadoop cluster. In: 2012 World Congress on Information and Communication Technologies; 30 October 2012-02 November 2012; Trivandrum, Indian. New York, USA: IEEE. pp. 505-510.

[16] Hadoop Examples Terasort, https://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html, erişim tarihi 15.09.2014.

[17] DigitalOcean Cloud Service, http://www.digitalocean.com, erişim tarihi 17.10.2014.

[18] Herodotou H, Lim H, Luo G, Borisov N, Dong L, Cetin FB, Babu S. Starfish: A self-tuning system for big data analytics. In: 5th Biennial Conference on Innovative Data Systems Research; 9-12 January 2011; Asimolar, CA. pp. 261-272.

[19] Tang Z, Wang Q, Cai S. Network-based inference algorithm on Hadoop. In: 20th International Symposium, ISMIS 2012; 4-7 December 2012; Macau, China. New York, USA: Springer Berlin Heidelberg. pp 367-376.