2015

HADOOP ÇATISININ BULUT ORTAMINDA GERÇEKLENMESİ VE BAŞARIM ANALİZİ

Göksu Zekiye ÖZEN

KIRGIZİSTAN TÜRKİYE MANAS ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

HADOOP ÇATISININ BULUT ORTAMINDA GERÇEKLENMESİ VE BAŞARIM ANALİZİ

Hazırlayan
Göksu Zekiye ÖZEN

Danışman
Doç. Dr. Rayımbek SULTANOV

Yüksek Lisans Tezi

Nisan 2015
BİŞKEK, KIRGIZİSTAN

KIRGIZİSTAN TÜRKİYE MANAS ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

HADOOP ÇATISININ BULUT ORTAMINDA GERÇEKLENMESİ VE BAŞARIM ANALİZİ

Hazırlayan
Göksu Zekiye ÖZEN

Danışman
Doç. Dr. Rayımbek SULTANOV

Yüksek Lisans Tezi

Nisan 2015

BİŞKEK, KIRGIZİSTAN

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Göksu Zekiye ÖZEN
İmza

YÖNERGEYE UYGUNLUK

“ Hadoop Çatısının Bulut Ortamında Gerçeklenmesi ve Başarım Analizi “ adlı Yüksek Lisans Tezi, Kırgızistan Türkiye Manas Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazım Yönergesi’ne uygun olarak hazırlanmıştır.

Göksu Zekiye ÖZEN	Doç.Dr. Rayımbek SULTANOV
İmza	İmza

Bilgisayar Mühendisliği ABD Başkanı
Doç. Dr. Rayımbek SULTANOV
İmza

KABUL VE ONAY

Doç.Dr. Rayımbek SULTANOV danışmanlığında Göksu Zekiye ÖZEN tarafından hazırlanan " Hadoop Çatısının Bulut Ortamında Gerçeklenmesi ve Başarım Analizi " adlı bu çalışma, jürimiz tarafından Kırgızistan Türkiye Manas Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği AnaBilim Dalı Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

04.06.2015

(Tez savunma sınav tarihi yazılacaktır.)

JÜRİ:

Danışman	Doç. Dr. Rayımbek SULTANOV	……..
	Prof. Dr. Sabıra NURCANOVA	……..
	Prof. Dr. Ulan BRİMKULOV	……..

04.06.2015

Prof. Dr. Zafer GÖNÜLALAN

Enstitü Müdürü

ÖNSÖZ

Çalışmalarım boyunca farklı bakış açıları ve bilimsel katkılarıyla beni aydınlatan, yakın ilgi ve yardımlarını esirgemeyen ve bu günlere gelmemde en büyük katkı sahibi sayın hocam Doç Dr. Rayımbek SULTANOV'a teşekkürü bir borç bilirim.

Deneysel çalışmalarım sırasında karşılaştığım zorlukları aşmamda yardımlarından ve desteklerinden dolayı sayın Doç. Dr. Mehmet TEKEREK'e ve bütün hocalarıma teşekkür ederim.

Ayrıca, çalışmalarım süresince sabır göstererek beni daima destekleyen başta eşim Yunus ÖZEN olmak üzere ve bugünlere gelmemde üzerimde büyük emekleri olan aileme sonsuz teşekkürlerimi sunarım.

Göksu Zekiye ÖZEN

Bişkek, Şubat 2015

HADOOP ÇATISININ BULUT ORTAMINDA GERÇEKLENMESİ VE BAŞARIM ANALİZİ

Göksu Zekiye ÖZEN

Kırgızistan Türkiye Manas Üniversitesi, Fen Bilimleri Enstitüsü

Yüksek Lisans, Nisan 2015

Danışman:Doç. Dr. Rayımbek SULTANOV

KISA 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 map aşamasında işler daha küçük parçalara ayrılır ve düğümlere dağıtılarak işlenir. Reduce aşamasında ise işlenen iş parçacıkları birleştirilerek sonuç elde edilir. İşlerin parçalanması, düğümlere dağıtılması, eş zamanlı olarak yapılan iş sayısı ve öbekteki düğüm sayısı gibi parametreler, işlerin tamamlanma süresine etki etmektedir. Bu çalışmanın amacı bir öbek üzerinde çalışan, düğüm, map ve reduce iş parçacıklarının sayısının Hadoop çatısının başarımını nasıl etkilediğini tespit etmektir. Bu amaçla 10 düğümlü bir öbek üzerinde Hadoop çatısı kurularak Piestimator, Grep, Teragen, Terasort kıyaslama araçları yardımıyla farklı düğüm, map ve reduce sayıları kullanılarak deneyler gerçekleştirilmiştir. Hadoop çatısı altında hazır gelen kıyaslama araçları ile gerçekleştirilen deneylerin sonucunda uygulamalar işlemci kullanım durumlarına göre işlemciyi az kullanan ve yoğun kullanan uygulamalar şeklinde sınıflandırılmıştır. İşlemciyi az kullanan uygulamalarda düğüm, map ve reduce sayısının artırılması işin verimliliğini artırmadığı gibi sistem kaynaklarını gereksiz yere kullanarak, iş için harcanan zamanın artmasına neden olmuştur. Bu nedenle işlemciyi az kullanan uygulamalarda düğüm, map ve reduce sayısı en az seçilmesi, işlemciyi yoğun kullanan uygulamalarda ise iş parçacıklarının işlendiği aşamaya göre map ya da reduce sayısının, düğümlerdeki toplam CPU sayısı kadar seçilmesi iş için harcanan zamanın eniyilemesi olarak bulunmuştur.

Anahtar Kelimeler: Büyük veri, Hadoop, bulut, MapReduce, kıyaslama araçları.

HADOOP FRAMEWORK IMPLEMENTATION AND PERFORMANCE ANALYSIS ON CLOUD

Goksu Zekiye OZEN

Kyrgyz Turkish Manas University Institute Of Science

Master Thesis, April 2015

Advisor: Assoc. Prof. Dr. Rayimbek SULTANOV

ABSTRACT

Hadoop framework uses 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. Parameters such as distributing method over nodes, the number of jobs held in a parallel fashion and the number of nodes in the cluster affect the execution time of jobs. Aim of this paper is to determine how number of nodes, maps and reduces affect the performance of Hadoop framework on a cloud environment. For this purpose, tests were carried out on a Hadoop cluster with 10 nodes hosted on a cloud environment by running PiEstimator, Grep, Teragen and Terasort benchmarking tools on it. These benchmarking tools available under Hadoop framework are classified as CPU-intensive and CPU-light applications as a result of tests. In CPU-light applications; increasing number of nodes, maps and reduces do not improve efficiency of these applications, even they cause increase of time spent on jobs by using system resources unnecessarily. Therefore, in CPU-light applications, selecting number of nodes, maps and reduces as minimum are found as the optimization of time spent on a process. In CPU-intensive applications, selecting number of maps or reduces equal to total number of CPUs on a cluster are found as the optimization of time spent on a process.

Keywords: Big data, Hadoop, cloud computing, MapReduce, benchmarking tools.

РЕАЛИЗАЦИЯ И АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ КАРКАСА HADOOP НА ОБЛАКЕ

Гоксу Зекийе Озен

Кыргызско Турецкий Университет "Манас", Институт Науки

Магистратурная диссертация, апрель 2015

Научный Руководитель: Доцент, Доктор Райымбек Султанов

АННОТАЦИЯ

Программный каркас Hadoop использует парадигму программирования MapReduce для обрабатывания больших данных, которые он распределяет в узлах кластера для обрабатывания и агрегации. MapReduce является одним из методов, используемых при обработке больших объемов данных, размещенных на больших компьютерных кластерах. Благодаря этому способу основная информация разбивается на маленькие куски и обрабатывается, распространяясь в узлах. Количество узлов в кластерах влияет на количество времени, требуемого для завершения работы. Цель данного исследования - определить влияние количества узлов в кластере на производительность программного каркаса Hadoop в облачной среде, с помощью различных инструментов бенчмаркинга. С этой целью были построены программные каркасы Hadoop на десятиузловом кластере и проведены испытания с помощью инструментов сравнения Piestimator, Grep, Teragen, TeraSort. В результате опытов, осуществленных инструментами подготовленными программным каркасом Hadoop, задания, согласно состоянию использования процессоров делятся частое и редкое использование. В частности, в заданиях, в которых процессор используется редко, увеличение количества узлов, map и reduce, не повышает производительность и использует источники системы в ненужны местах, расходуя больше времени. Поэтому в заданиях с редким использованием процессора минимальный выбор количества узлов map и reduce оптимизирует потраченное время на обработку данных. В заданиях с частным использованием процессора выбор количества узлов map и reduce равным общему числу CPU в кластере, оптимизирует потраченное время на обработку данных.

Ключевые Слова: большие данные, Hadoop, облако, MapReduce, бенчмаркинг.

HADOOP КАРКАССЫНЫН БУЛУТ ЧӨЙРӨСҮНДӨ ИШТЕШИ ЖАНА ЖЕТИШКЕНДИК АНАЛИЗИ

Гоксу Зекийе Озен

Кыргыз-Түрк Манас Университети, Табигий илимдер институту

Магистртуралык Диссертациялык, Апрель 2015

Илимий Жетекчи: Доцент, Доктор Райымбек Султанов

БАШ СӨЗ

Hadoop программалык каркасы чоң көлөмдүү маалыматты иштетүүдө, иштелүүчү маалыматтын кластердеги ишчи пунктардын ичинде таралышы, иштетилиши жана кайрадан чогулуп маанилуу абалга келтирилиши үчүн MapReduce программалоо парадигмасын колдонот. Чоң көлөмдүү маалыматты иштетүүдө колдонулган бир топ метод бар. MapReduce компьютерде сакталган чоң көлөмдөгү маалыматтардын иштетилишинде колдонулган техникалардын бири болуп саналат.. Бул метод менен map этабында иштер кичинекей бөлүкчөлөргө бөлүнөт жана түймөктөргө таратылып иштелет. Reduce этабында болсо иштелген бөлүкчөлөр топтолуп жыйынтык чыгарылат.

Hadoop каркассы бир кластер үстүндө көп түймөктөрдө курула алат .Кластердеги түймөк саны көбөйгөн сайын түймөктөрдүн байкоосу жана башкаруусу кыйындагандыгы үчүн виртуалдаштыруу чөйрөлөрү тандалат. Ошондуктан булут чөйрөсүндө виртуалдаштыруу программалык камсыздоосу колдонулуп 10 түймөктүү кластер түзүлгөн. Ар бир түймөккө жетүү: ишенимдүү маалымат жиберүү үчүн пароль коюу методдорун колдонгон тор протоколдору менен иш жүзүнө ашарылат. Hadoop каркассынын жетишкендигин жогорулатуу үчүн бир топ параметр тууралоолору кылына алат. Түймөктөргө Hadoop каркассынын 1.0.3 версиясы курулду жана иштин артышына жол ачпагандай негизги тууралоолор жасалды. Айрыкча түймөктөрдүн процессор, RAM ж.б. колдонуулары менен жабдык иштерини ,камтыган түрдүү каражаттар жана методдор менен каттала ала турган жана web-based катары көрүнүшүнү камсыз кылган бир байкоо каражаты колдонулду.Тажрыйбалардын иш жүзүнө ашышында Hadoop каркассында даяр болгон салыштыруу каражаттары колдонулган. Бул салыштыруу каражаттары менен көлөмдүү маалымат жыйындылары түзүлө алат жана даяр маалымат

жыйындылары үстүндө иреттөө,сөз саноо,издөө,ж.б көлөмдүү маалымат иштери жүргүзүлө алат. Айрыкча жабдыкты жана каркассты тесттен өткөргөн салыштыруу каражаттары да бар.

Тажрыйбада колдонулган салыштыруу каражаттары: Pi санын божомолдоодо колдонулган PiEstimator, колдонуучу көрсөткөн бир шаблонго жараша издөө жүргүзөө алган Grep, көлөмдүү маалымат жыйындысын түзүүнү камсыз кылган Teragen жана көлөмдүү маалымат жыйындысынын үстүндө иреттөө жүргүзгөн Terasort салыштыруу каражаттарыдыр.Тажрыйбалар бул салыштыруу каражаттарыны жардамы менен 10 түймөктүү бир кластерде ар кандай түймөк саны үстүндө map жана reduce саны. колдонулуп жүргүзүлдү жана иштердин жыйынтыктоо убакыты сакталды. Бул иште түймө,map жана reduce иш бөлүкчөлөрүнүн санынын Hadoop каркассынын жетишкендигине кандай таасир бергенин аныктоого аракет кылынган.Тажрыйбаларда колдонулган салыштыруу каражаттары,байкоо каражатынын жардамы менен каттого алынган процессор колдонууларына жараша процессорду аз колдонгон жана прцессорду көп колдонгон тиркөөлөр болуп бөлүнгөн. PiEstimator жана Grep салыштыруу каражаттарынын процессорду аз колдонгону, Teragen жана Terasort салыштыруу каражаттары болсо процессорду көп колдонгондугу тастыкталган.

Тажрыйбада процессор колдонуу шарттарына жараша колдонулган түймөк, map жана reduce сыяктуу параметрлерге жараша иштердин жыйынтыктоо убактыларыны бири-бириндеи башкача болгону байкалды.MapReduce методунда иштердин бөлүнүшу, түймөктөргө таралшы,бир убакытта кылынган иш саны жана кластердеги түймөк саны параметрлердин,иштердин жыйынтыктоо убакыты өзгөрткөнү жана Hadoop каркассынын жетишкендигине таасир бергени аныкталган.

Тажрыйбалардын жыйынтыгында:Процессорду аз колдонгон PiEstimator жана Grep салыштыруу каражаттарында да окшош түймөк саны үстүндө, map жана reduce саны арткан сайын иш үчүн коротулган убакыттын артканы ошол себептен иштин жетишкендигинин азайганы, түймөк санынын артышынын болсо бир деңгээлде иш үчүн коротулган убакыттын азайтылбагандыгы тастыкталган.Ошондуктан прцессорду аз колдонгон тиркөолөрдө түймөк, map жана reduce санины эң аз тандалышы ыңгайлуу.Процессорду көп колдонгон

Teragen жана Terasort салыштыруу каражаттарында түймөк санынын артышынын иштин жетишкендигин жогорулатканы,иш бөлүкчөлөрүнүн иштелген этабына жараша map же болбосо reduce санынын,түймөктөгү жалпы CPU санына чейин тандалышы иш үчүн короткон убакыттын оптимизациясы катары табылган

Ачкыч Сөздөр: Hadoop, булут, MapReduce, салыштыруу каражаттары.

İÇİNDEKİLER

HADOOP ÇATISININ BULUT ORTAMINDA GERÇEKLENMESİ VE BAŞARIM ANALİZİ

KISALTMALAR VE SİMGELER

Sembol	**Anlamı**
HDFS	Hadoop Distributed File System (Hadoop Dağıtık Dosya Sistemi)
CPU	Central Processing Unit (Merkezi İşlemci Birimi)
KVM	Kernel-based Virtual Machine (Çekirdek Tabanlı Sanal Makine)
GFS	Google File System, (Google Dosya Sistemi)
XML	EXtensible Markup Language (Genişletilebilir İşaretleme Dili)
XDR	eXternal Data Representation
RRDtool	Round Robin Databases Tools
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
Gmetad	Ganglia Meta Daemon
Gmond	Ganglia Monitoring Daemon
XMR	eXtensible-MapReduce
ARIA	Automatic Resource Inference and Allocation
SSD	Solid State Drive
JDK	Java Development Kit
SSH	Secure Shell

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

1. GİRİŞ VE AMAÇ

İnternet siteleri, sosyal araçlar, bloglar, akıllı telefonlar, tümleşik devreler vb. tarafından üretilen verinin boyutu hızla artmaktadır. Bu ortamlardan elde edilen yapısal ya da yapısal olmayan veri, büyük veri olarak isimlendirilir. Büyük verinin anlamlı hale getirilebilmesi için geleneksel veri işleme ve saklama yöntemleri kullanılamamaktadır [1].

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 çok önemli hale gelmiştir. Örneğin Facebook tarafından, günlük olarak işlenen veri miktarının 250 terabyte olduğu rapor edilmiştir [2].

Büyük verinin ve büyük veri analiz araçlarının kullanım alanlarından bir tanesi karar destek sistemleridir [3]. Büyük veri işleme araçları olmadan geçmişe dönük veri analizi yapılabilirken, günümüzde dağıtık bilgisayar sistemlerinin hızı sayesinde geleceğe dönük anlık tahminler yapmak mümkün hale gelmiştir.

Hadoop çatısı, MapReduce yöntemini uygulayarak dağıtık olarak büyük veri işleme yapan yaygın çatılardan biridir [4]. MapReduce, geniş bilgisayar öbekleri üzerinde barındırılan büyük verinin işlenmesinde kullanılan tekniklerden biridir. Hadoop çatısı, Hadoop Dağıtık Dosya Sistemi'ni (Hadoop Distributed File System, HDFS) kullanır [5].

Hadoop çatısı literatürde yer alan ve büyük veri işlemede yaygın kullanılan bir çatı olduğu için veri okuma, işleme, arama ve kaynak kullanımı başarımlarını etkileyen sebeplerin bilinmesi etkin Hadoop öbekleri kurulumu için önemlidir [6].

Değişen veri boyutu ve MapReduce işlerinin sayısı gibi parametrelere göre MapReduce sisteminin başarımı değişmektedir [7].

Çalışma kapsamında başarım analizini etkileyen faktörlerin belirlenebilmesi için bulut ortamında gerekli deney düzeneği kurularak çeşitli deneyler gerçekleştirilmesi ve

deneylerden elde edilen sonuçlar kıyaslanarak başarımı etkileyen faktörlerin araştırılması hedeflenmektedir.

Bu çalışmanın amacı bir öbek üzerinde çalışan düğüm sayısına göre map ve reduce iş parçacıklarının sayısının CPU kullanım durumuna bağlı olarak Hadoop çatısının başarımını nasıl etkilediğini araştırmaktır.

Bu doğrultuda aşağıdaki sorulara cevap aranmıştır:

Varsayılan olarak yapılandırılmış Hadoop çatısı altında dağıtık olarak işlenecek büyük veri işinde düğüm, map ve reduce sayısı, işin başarımını nasıl etkiler?

Bir öbek üzerinde çalışan düğümler için map ve reduce sayısı gibi parametreler hangi duruma göre belirlenmelidir?

Bu MapReduce büyük veri işlemede kullanılan uygulamalarda sistemin eniyilemesi için gereken düğüm, map ve reduce sayısı belirlenebilir mi?

2. GENEL BİLGİLER

Yapısal bir veri olmayan büyük verinin işlenmesi anlamlı hale getirebilmesi ve saklanması için geleneksel yöntemlerin kullanılamaması, veri yoğunluğunun yüksek olmasından dolayı işlenen verilerin tek bir düğümde saklanamaması gibi nedenlerden büyük veriyi işlemede farklı donanımsal ve yazılımsal çözümler bir arada kullanılmıştır. Büyük veri işlemede kullanılan birçok yöntem bulunmaktadır. Büyük veriyi işlemede, veriyi analiz etmede, saklamada ve geri getirmede en yaygın kullanılan yöntemlerden biri MapReduce yöntemidir. Google tarafından 2004 yılında geliştirilen MapReduce yönteminde veri paralel ve dağıtık olarak bir ya da birden fazla öbek üzerinde işlenebilir. Bu yöntemde işler daha küçük parçalara ayrılır ve düğümlere dağıtılarak işlenir ve işlenen iş parçacıkları toplanarak sonuç elde edilir [8].

Büyük veri işlemede kullanılan Hadoop çatısı MapReduce yöntemini ve taşınabilir dosya sistemi HDFS'yi kullanır. HDFS; büyük verinin bloklar haline getirilmesine, düğümlere dağıtılmasına, sağlıklı bir şekilde işlenmesine ve geri getirilerek saklanmasına olanak sağlar [9].

Sanallaştırma yazılımları öbekteki düğüm sayısı artıkça ya da öbeğe yeni düğümler eklenmek istendiğinde düğümlerin yönetimlerini kolaylaştırmak ve kaynakları etkin bir biçimde paylaştırmak için kullanılır Çekirdek Tabanlı Sanal Makina (Kernel-based Virtual Machine, KVM) [10] bu alanda yaygın olarak kullanılan açık kaynaklı sanal makina çözümlerinden birisidir.

Bir öbek üzerinde çalışan düğümlerin etkin bellek kullanımı, CPU kullanımı, sabit disk kullanımı ve ağ trafiği vb. başarımları Ganglia [11] izleme aracı tarafından izlenerek kayıt altına alınabilmektedir.

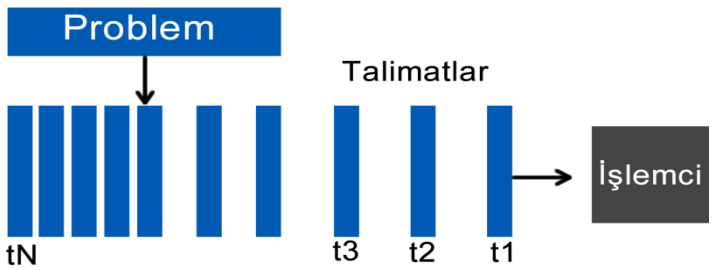
Hadoop çatısı altında büyük veriyi işlemede ve test etmede kullanılan Piestimator, Teragen, Grep, Terasort gibi MapReduce kıyaslama araçları sunulmaktadır [12]. Bu kıyaslama araçları ile veri setleri oluşturulabilmektedir veya hazır veri setleri kullanılabilmektedir. Bu veri setleri üzerinde arama, sözcük sayma, sıralama ve doğrulama gibi sık kullanılan büyük veri işlemleri gerçekleştirilebilmektedir.

Bu çalışmada Piestimator, Grep, Teragen, Terasort kıyaslama araçları kullanılarak yapılan deneylerde CPU kullanımına göre bu araçlar CPU'yu az kullanan ve yoğun kullanan şeklinde sınıflandırılmıştır ve kıyaslama araçlarının CPU kullanımlarına bağlı olarak düğüm, map ve reduce gibi parametrelerin işin başarımına olan etkisi incelenmiştir.

Piestimator, giriş verisini kendi oluşturur ve bu veri üzerinde kıyaslama işlemi yapar. Çalışma zamanında diskten okuma ya da diske yazma işlemi yapmaz. Grep, giriş dizinindeki metin dosyası içinde istenilen kelimeyi arar ve dosyada kaç defa tekrarlandığını bularak sonucu <kelime> boşluk <adedi> şeklinde çıkış dizini içindeki dosyaya yazar. Teragen, istenilen boyutta veri seti üretir ve bunu bir çıkış dizini içindeki dosyaya yazar. Terasort, giriş dizinindeki dosyalardaki veriyi sıralayarak sonucu bir çıkış dizini içine yazar [13].

2. 1. Paralel Hesaplama ve Dağıtık İşleme

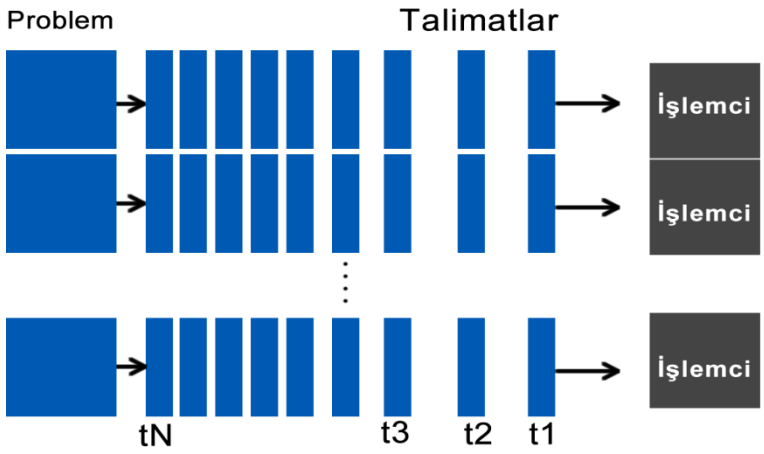
Geleneksel olarak seri hesaplama yöntemi [14] için yazılan yazılımlar tek bir bilgisayarın üzerindeki işlemcide çalıştırılırlar. Seri hesaplamada bir problem ayrık bir talimat serisine bölünür ve bu talimatlar birbiri ardına çalıştırılır yani birim zamanda tek bir talimat işlenebilir. Bu durum Şekil 2-1'de gösterilmeye çalışılmıştır.



Şekil 2-1 Seri hesaplama yönteminde bir problemin çözümü [15].

Paralel hesaplama [14, 15] ise bir sorunu çözmek için sistem kaynaklarını eş zamanlı olarak kullanır. Paralel hesaplamada problem birden fazla farklı parçaya ayrılır ve her

bir parça birden çok talimatlara bölünerek, her bir talimat farklı işlemciler üzerinde eşzamanlı olarak işlenir. Bu durum Şekil 2-2'de gösterilmeye çalışılmıştır.



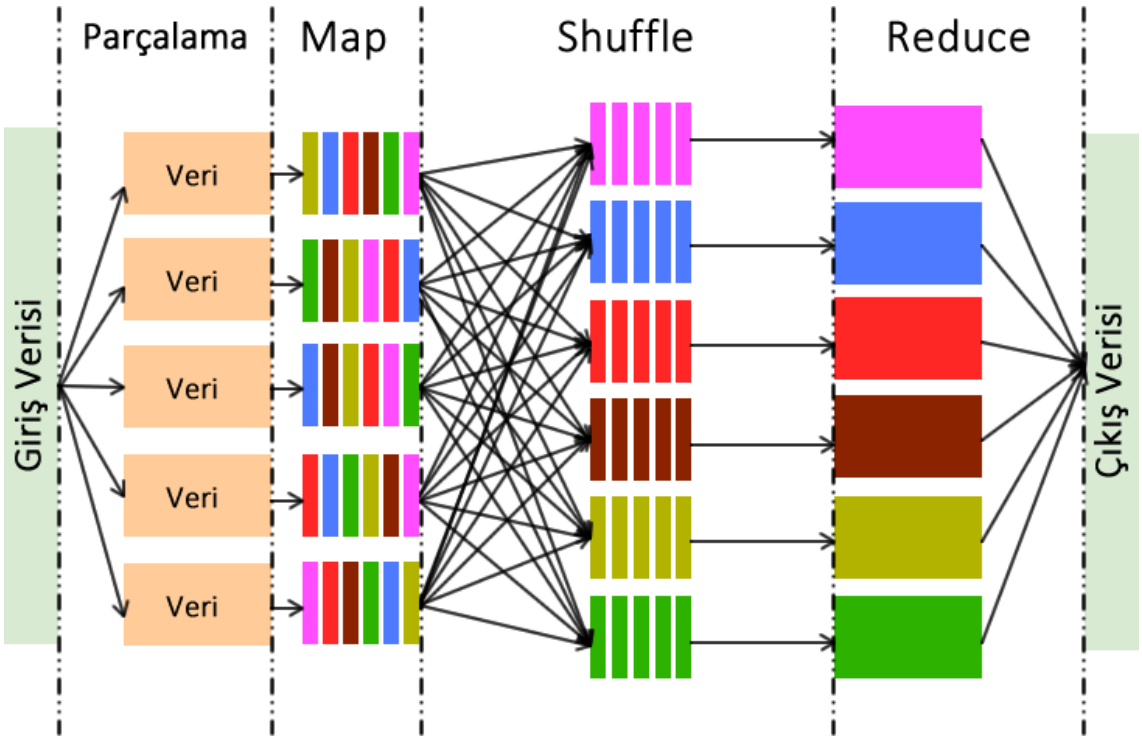
Şekil 2-2 Paralel hesaplama yönteminde bir problemin çözümü [15].

Paralel hesaplama yönteminde kullanılan sistem kaynakları; bir bilgisayar üzerinde çoklu işlemci olabileceği gibi bir ağda birden çok bilgisayar ya da bir ağda çoklu işlemcilere sahip bilgisayarlardan oluşabilir. Paralel hesaplamanın yöntemine göre sistem kaynakları paylaşılır. Bu işlemci, hafıza ya da veri yolu şeklinde olabilir [14, 15].
Paralel hesaplama yöntemiyle çözülecek olan sayısal problemler, parçalara ayrılan iş parçacıklarını eş zamanlı olarak işleyebilen, herhangi bir zaman anında çoklu program talimatları işletebilen ve tek kaynak yerine çoklu kaynak kullanıldığında işin daha kısa sürede işlenebilmesini sağlama gibi özelliklere sahiptir [15].
Paralel hesaplama atmosfer, yerküre, genetik, bilgisayar ve matematik bilimi vb. gibi yüksek işlemci gücü gerektiren ve yinelemeli olmayan gerçek dünya problemlerinin çözümünde kullanılabilir.
Dağıtık bir sistem bir amaca ulaşmak için birbirleriyle haberleşen özerk düğümlerin oluşturduğu bir ağdır. Dağıtık sistemler fiziksel olarak bir hafızayı ya da işlemciyi paylaşmazlar. Belirli bir amacı yerine getirebilmek için birbirleriyle mesajlaşabilir, ya da birbirlerine ağ üzerinden veri transfer edebilirler. Haberleşmede kullanılan mesajlar; veri paketleri, belirli değişkenleri içeren işletilebilir prosedürler ya da sinyaller olabilir.

Dağıtık sistemlerde yer alan düğümlerin görevleri sahip oldukları yazılım veya donanım gibi sistem özelliklerine göre birbirinden farklı olabilir [16].

2. 2. MapReduce Yöntemi

Map aşamasında master düğüm verileri alıp küçük parçalara ayırarak slave düğümlere dağıtır ve reduce aşamasında ise tamamlanan işler birleştirilerek sonuç elde edilir.
MapReduce yöntemi paralel işlenebilen problemlerdeki büyük veri setlerini kullanarak bir öbek ya da bir grid (çoklu öbek) üzerinde dağıtık ve paralel olarak işler [17].
Öbek aynı yerel alan ağı içinde benzer donanımları kullanan düğümlerdir. Grid ise farklı coğrafik bölgelerde yer alabilen ve ara donanım ve yazılımlarla birleştirilmiş ve farklı donanımları içerebilen dağıtık sistemlerdir [18].
MapReduce yöntemi veriyi anahtar-değer (key-value) çiftlerine ayırarak işler. MapReduce yöntemi map, shuffle/sort ve reduce olmak üzere üç aşamadan oluşur. İlk olarak master düğüm verileri alıp daha küçük parçalara ayırarak slave düğümlere dağıtır. Master düğüm dağıtma işlemini gerçekleştirirken iletim mesafesini azaltmak için veriyi yerelleştirerek en yakındaki slave düğümlere atamayı tercih eder ve verinin bozulması ihtimaline karşı; verinin tüm düğümlerde yedeklenmesini sağlar. Map aşaması tüm düğümler üzerinde çalıştırılır ve yerelleştirilmiş veri anahtar-değer çiftlerine ayırılır. Her farklı veri tekil bir anahtar değerine sahiptir. Bu aşamanın sonunda oluşan ara veri geçici dizinde tutulur. İkinci aşama shuffle/sort aşamasıdır. Bu aşamada; tüm düğümler üzerinde map aşamasında anahtar-değer çiftlerine ayrılan tüm ara veriler, aynı anahtara sahip ara veriler aynı düğüm üzerinde yer alacak şekilde yeniden dağıtılır ve sıralanır. Reduce aşamasında ise aynı anahtara sahip değerler toplanarak sonuç oluşturulur [17].
Giriş verisinin master düğüm tarafından parçalanarak düğümlere dağıtılması, map, shuffle ve reduce aşamaları Şekil 2-3'de gösterilmiştir.



Şekil 2-3 MapReduce aşamaları.

2. 3. Hadoop Çatısı

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) [19] tutarak, arama indekslerinden anlamlı sonuçlar çıkarmak için kullanılmıştır. Hadoop 2004 yılında bu projeden ayrılarak ayrı bir çatı olarak devam etmiştir [20]. Hadoop çatısı, MapReduce yöntemini uygulayarak paralel ve dağıtık olarak büyük veri işleme yapan yaygın çatılardan biridir [4, 9].

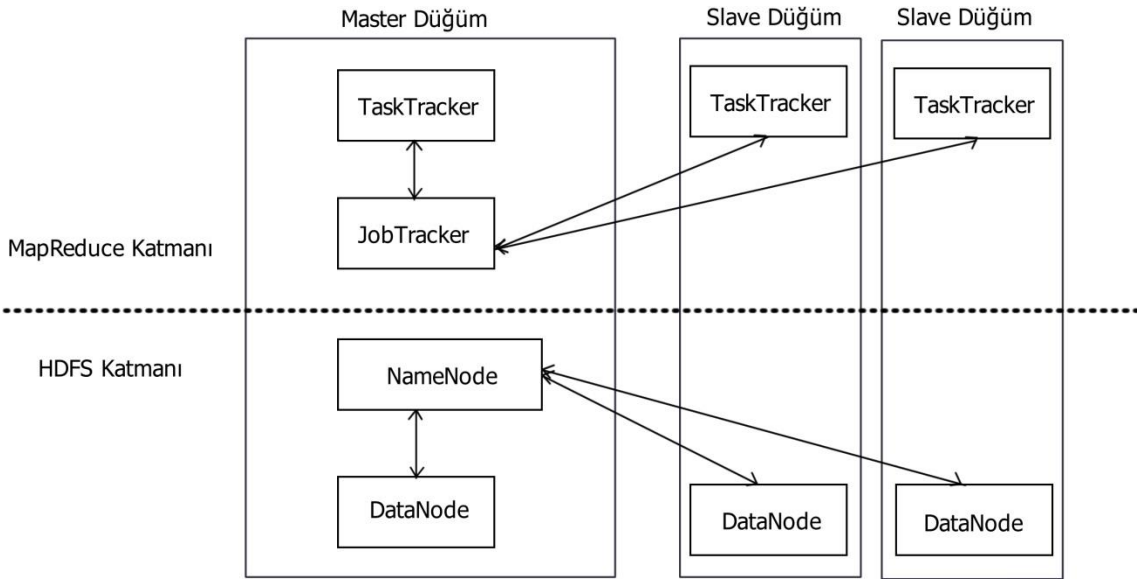
Hadoop çatısı, master - slave mimarisine göre tasarlanmıştır. Bir öbek üzerinde bir düğüm master, geri kalan düğümler slave olarak çalışır.

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 [21].
Hadoop çatısı, Hadoop Dağıtık Dosya Sistemini (Hadoop Distributed File System, HDFS) kullanır.

2. 3. 1. HDFS dosya sistemi

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 [22].
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 [5]. Şekil 2-4'de Hadoop (sürüm 1.x) mimarisi gösterilmektedir.



Şekil 2-4 Hadoop (sürüm 1.x) mimarisi. Mapreduce katmanı ve HDFS katmanı bileşenleri.

2. 4. Sanallaştırma Ortamı

Büyük veri işlemede bir ya da birden fazla öbek kullanılabilir. Öbekteki düğüm sayısı arttıkça yönetimi zorlaşacağından dolayı kurulum, dağıtım, çizelgeleme ve etkin kaynak yönetimi için sanallaştırma çözümleri geliştirilmiştir. Çekirdek Tabanlı Sanal Makina (Kernel-based Virtual Machine, KVM) yaygın olarak kullanılan açık kaynaklı sanal makina çözümlerinden birisidir. KVM; Linux için tüm x86 platformlarda (AMD, Intel vd.) full sanallaştırma çözümleri sunan bir ortamdır. Yüklenebilir çekirdek modülü sayesinde çekirdek sanallaştırma altyapısı ve işlemci özel modülleri sağlar. KVM açık kaynaklı olması nedeniyle çeşitli açık kaynaklı bulut sağlayıcılar, OpenStack, Docker, Bare metal sunucular vd. ile uyumlu çalışır [23].

2. 5. Ganglia İzleme Aracı

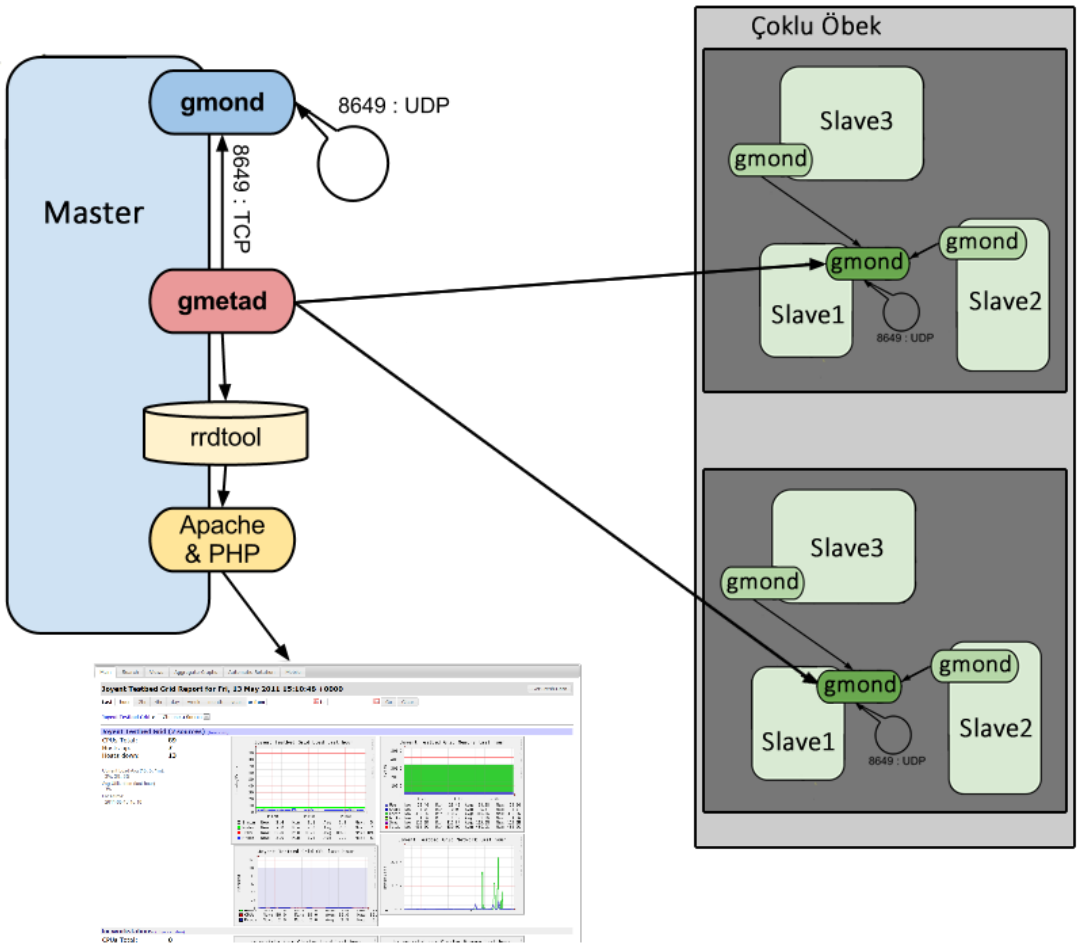
İzleme ve takip işlemleri öbek sistemlerinde (grid) önemli bir yere sahiptir. Öbek sistemlerde izleme araçları kullanılmadığında sistemin durumunu, sağlıklı çalışıp çalışmadığını ve başarımını belirlemek neredeyse imkânsızdır. İzleme verileri, dağıtık

bir sistemde, arıza tespiti ve çizelgelemede, başarım analizi ve başarım eniyilemesi için kullanılabilir [24].

Ganglia, bulut ya da grid gibi yüksek başarımlı hesaplama sistemleri için ölçeklenebilir dağıtık izleme sistemidir [11, 24, 25]. Ganglia veri gösterimi için (EXtensible Markup Language, XML), veri iletimi için (eXternal Data Representation, XDR), veriyi depolamak ve görselleştirmek için de (Round Robin Databases Tools, RRDtool) kullanır [24, 25].

2. 5. 1. Ganglia izleme aracı mimarisi

Ganglia izleme aracında öbekteki tüm düğümler ağaç yapısı şeklinde birbirine bağlıdırlar. Ganglia izleme aracı gmond (Ganglia Monitoring Daemon / Ganglia İzleme Arkaplan Uygulaması) ve gmetad (Ganglia Meta Daemon / Ganglia Meta Arkaplan Uygulaması) olarak adlandırılan iki ana arka plan uygulaması, Ganglia php tabanlı ön uç uygulama ve diğer küçük yardımcı uygulamalardan oluşur. Gmond, tüm düğümler üzerinde çalışarak düğümü ve düğüm üzerindeki değişiklileri izleme ve bildirmeden, diğer tüm ganglia düğümlerinin durumlarını tekli ya da çoklu dağıtım kanalı üzerinden dinleme ve XDR formatında UDP mesajı olarak göndermeden ve öbek durumunun XML olarak tanımlanarak TCP üzerinden gönderilmesinden sorumludur. Gmetad arka plan uygulaması ise sadece master düğüm üzerinde çalışır ve periyodik olarak tüm öbeklerden gelen bilgileri toplar, XML olarak toplanan verileri ayrıştırır ve tüm anlık sayısal değerleri Round Robin olarak veri tabanına kaydeder. Ganglia php tabanlı ön uç uygulama ise toplanan tüm bilgilerin anlamlı grafikler şeklinde bir web sayfası olarak gösterilmesini sağlar [25]. Şekil 2-5'de çoklu öbek üzerinde çalışan Ganglia izleme aracının mimarisi gösterilmektedir.



Şekil 2-5 Ganglia izleme aracı mimarisi.

2. 6. Alan Yazın Çalışmaları

Alan yazınında Hadoop çatısının başarımını etkileyen faktörlerle ilgili çeşitli çalışmalar yer almaktadır. 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 sunulmuştur [26]. Genişletilebilir MapReduce (eXtensible-MapReduce, XMR); verinin dağıtılması, map ve reduce iş parçacıklarının sayısının belirlenmesi ve karıştırma aşamasında bir yönlendirme çizelgeleyici oluşturma modüllerinden oluşan bir modeldir [27]. MapReduce yönteminin iş akışını doğrudan etkileyen aşamalarına odaklanmıştır. Hadoop koduna eklenti yapmaktadır. Starfish; büyük veri analizi için geliştirilmiş bir otomatik ayarlama uygulamasıdır. Hadoop sisteminin başarımını artırmak için çeşitli

ayarları otomatik olarak düzenlemektedir. İşlerin çalışması ile ilgili izleme verilerini çalışma zamanında toplar. Starfish topladığı verilere dayanarak birçok Hadoop parametresini otomatik ayarlayan karmaşık bir sistemdir. Uygulamanın kendisi de sisteme ayrıca yük getirmektedir. Bununla beraber Starfish ile ayarlama yapmak için uygulamanın Hadoop üzerinde belirli bir süre çalışması gerekmektedir [28]. (Automatic Resource Inference and Allocation, ARIA) ise Hadoop sisteminin daha önce yaptığı işlerle ilgili sistemde kayıtlı bilgilerden yola çıkarak iş profilleri çıkaran bir çatıdır. Bu bilgileri kullanarak, yeni başlayacak bir işin tamamlanma süresini tahmin etmeye çalışır [29]. Zhang ve arkadaşları tarafından geliştirilen parametrelendirilebilen kıyaslama çatısı da Hadoop sisteminin geçmişte yaptığı işleri inceleyerek bir platform başarım modeli çıkarmaktadır. Bunun yanında mevcut işin çalışmasını da bir süre gözleyerek MapReduce başarım modelini çıkarmaktadır. Bu iki modeli kullanarak işin tamamlanma süresini tahmin etmeye çalışmaktadır. Bir izleme çatısının sistemin çalışma başarımı üzerindeki etkisini azaltmak için Hadoop koduna çeşitli sayaçlar yerleştirmişlerdir. Hadoop kodunun yeniden derlenmesini de gerektirdiği için karmaşık bir yöntemdir [30]. Babu ve arkadaşları büyük veri setleri üzerinde Hadoop ayarlarının etkisini ortaya koymaktadır [31]. Fadika ve arkadaşları Hadoop ile birlikte farklı iki tane MapReduce çatısını karşılaştırarak hangi çatının hangi işlerde yüksek çalışma başarımı gösterdiğini ölçen deneyler yapmışlardır. Yaptıkları çalışmada denenen uygulamalar; CPU yoğun uygulamalar, bellek yoğun uygulamalar ve veri yoğun uygulamalar olarak gruplanmıştır [32]. Bu çalışma ise bulut ortamında gerçekleştirilmiştir. Ayrıca öbekteki düğüm sayısı 1'den 10'a kadar artırılmıştır. Hadoop çatısının varsayılan ayarları kullanılmış ve işlemci kullanım durumuna göre sınıflandırılan Piestimator, grep, teragen ve terasort kıyaslama araçlarıyla farklı düğüm sayısında çeşitli deneyler gerçekleştirilmiş düğüm, map ve reduce sayısının etkinliği ve bulut üzerinde Hadoop çatısının başarımı araştırılmıştır.

3. GEREÇ VE YÖNTEM

Bu çalışmada Hadoop çatısı altında sunulan Piestimator, Teragen, Grep, Terasort kıyaslama araçları, Hadoop çatısının varsayılan ayarlarıyla aynı deney düzenekleri üzerinde farklı düğüm, map ve reduce sayısı kullanılarak test edilmiştir.

Hadoop çatısında işlenen verinin türüne ve CPU'yu kullanma durumuna göre işin başarımını artırmaya yönelik parametre ayarları yapılabilmektedir [28]. Kıyaslama araçları, CPU kullanım durumuna göre sınıflandırılmıştır ve başarımı artırma yönünde herhangi bir parametre ayarı yapılmamıştır. Yapılan deneylerde; düğüm sayısına göre, kullanılan map ve reduce sayısının sistemin başarımını nasıl etkilediği incelenmiştir.

Bu kıyaslama araçlarının CPU kullanımları göz önüne alınarak düğüm, map ve reduce sayısına göre sistemin eniyilemesine ulaşılması amaçlanmıştır.

3. 1. Deney Ortamının Kurulması

Çalışmadaki bütün deneyler DigitalOcean [33] 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 CPU, 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. Düğümlerden bir tanesi ana düğüm olarak, diğer 9 düğüm ise işçi düğüm olarak ayarlanmıştır. Düğümlere erişim Güvenli Kabuk (Secure Shell, SSH) sunucu yazılımıyla gerçekleştirilmiştir. Tüm düğümlerde Hadoop konfigürasyon dizini altında yer alan core-site.xml, mapred-site.xml ve hdfs-site.xml dosyaları herhangi bir başarım artışı sağlamayacak şekilde, Hadoop çatısının çoklu düğüm üzerinde çalışması için gerekli temel ayarları içeren

parametreler kullanılarak Tablo 3-1, Tablo 3-2 ve Tablo 3-3'te görüldüğü gibi yapılandırılmıştır.

Tablo 3-1 core-site.xml ayarları.

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

Tablo 3-2 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 3-3 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 üzerindeki yeri
dfs.name.dir	${hadoop.tmp.dir}/dfs/name	NameNode'nin yerel dosya sistemi üzerindeki yeri

3. 2. Deneylerin Gerçekleştirilmesi

Tüm kıyaslama araçları için yapılan deneyler 10 düğümlü bir öbek üzerinde farklı düğüm sayısı üzerinde farklı map ve reduce sayıları kullanılarak gerçekleştirilmiştir. Deneylerde kullanılan kıyaslama araçlarının her birinin CPU kullanımları Ganglia

izleme aracıyla kaydedilmiştir. CPU kullanımları, her deneyde yer alan düğümlerin kullandığı ortalama CPU miktarı şeklinde ölçülmüştür. Deneylerde kullanılan kıyaslama araçları ve bu araçlar hakkında açıklamalar alt başlıklar halinde sunulmuştur.

3. 2. 1. Piestimator kıyaslama aracı

Piestimator kıyaslama aracı Monte-Carlo yöntemini kullanarak pi sayısı tahmini yapmaktadır. Monte Carlo yönteminde bir karenin içine çizilmiş dörtte bir daire dilimi düşünülerek bu karenin içine rastgele üretilen girişler yerleştirilir. Daire diliminin içinde kalan girişlerin toplamı tüm girişlere bölünür ve sonuç 4 ile çarpılır. Daire dilimi içinde üretilen girişlerin toplamına m dersek, tüm girişlerin toplamına da n dersek Eşitlik 1'i elde etmiş oluruz.

$$\pi = \frac{m}{n} \times 4 \qquad (1)$$

Map aşamasında 0 ve 1 şeklinde rastgele girişler oluşturulur. 1'ler çemberin içinde kalan giriş sayısını, 0'lar ise çemberin dışında kalan giriş sayısını ifade eder. Reduce aşamasında ise çemberin içinde ve dışında kalan girişler toplanır, çemberin içinde kalan girişler toplam girişe oranlanır ve 4 ile çarpılarak pi sayısı elde edilir [34].
Pi sayısının tahmini için yapılan deneyler 1, 2, 5 ve 10 düğüm üzerinde sırasıyla 1, 10 ve 100 map kullanılarak gerçekleştirilmiştir. Örnekleme sayısı giriş sayısını değiştirmeyecek şekilde sırasıyla 10000000, 1000000, 100000 seçilmiştir. Giriş sayısı o deneyde kullanılan map ve örnekleme sayısının çarpılmasıyla elde edilir.

3. 2. 2. Grep kıyaslama aracı

Grep kıyaslama aracı giriş dizinindeki metinleri map aşamasında satır satır alarak kullanıcının girdiği kurallı ifade ile eşleşen ifadeleri, (eşleşen ifade, değer çifti) şeklinde ayırır. Reduce aşamasında ise eşleşen ifadelerdeki değerler toplanır ve daha sonra çıkış dizinindeki bir dosyaya <eşleşen ifade>boşluk<adedi> şeklinde yazılır [13].
Grep kıyaslama aracı 10 düğümlü bir öbek üzerinde Gutenberg [35] projesinden alınmış çeşitli metin dosyalarının birleştirilerek ve tekrarlanarak çoğaltılmasıyla elde edilen 1

GB metin dosyası üzerinde çalıştırılmıştır. Deneyler 10 düğümlü bir öbek üzerinde sırasıyla 1, 2, 5 ve 10 düğüm üzerinde gerçekleştirilmiştir. Bu düğümler üzerinde sırasıyla 1, 10 ve 100 map sayıları kullanılmıştır. Her map değeri için de sırasıyla 1, 2, 5, 10, 20, 50 ve 100 reduce sayısı kullanılmıştır.

Reduce sayısı kıyaslama aracı çalıştırılırken bir parametre olarak verilmiştir. Map sayısı da komut satırından belirlenebilmesine rağmen bu parametre Grep kıyaslama aracında kullanılamaz. Map sayısı, giriş dizinindeki metin dosyalarının ve HDFS dosya sisteminin kullandığı blok boyutuna bağlıdır [26]. HDFS dosya sistemine kopyalanan dosyanın blok boyutu, kopyalama esnasında parametre yardımıyla değiştirilerek map sayısı ayarlanmıştır.

3. 2. 3. Teragen kıyaslama aracı

Teragen kıyaslama aracı istenilen büyüklükte rastgele veri üretmekte kullanılır [12, 13, 36]. Üretilen verinin HDFS dosya sistemine hangi blok boyutunda yazılacağı belirtilebilir. Bu kıyaslama aracında reduce aşaması çalıştırılmaz. Map sayısı Teragen çalıştırılırken bir parametre olarak verilmiştir. Teragen 10 düğümlü bir öbek üzerinde sırasıyla 1, 2, 5 ve 10 düğüm üzerinde gerçekleştirilmiştir. Bu düğümler üzerinde sırasıyla 1, 2, 5, 10, 20, 50 ve 100 map sayısı kullanılmıştır. Her bir deneyde 10 GB'lik rastgele veri üretilmiştir.

3. 2. 4. Terasort kıyaslama aracı

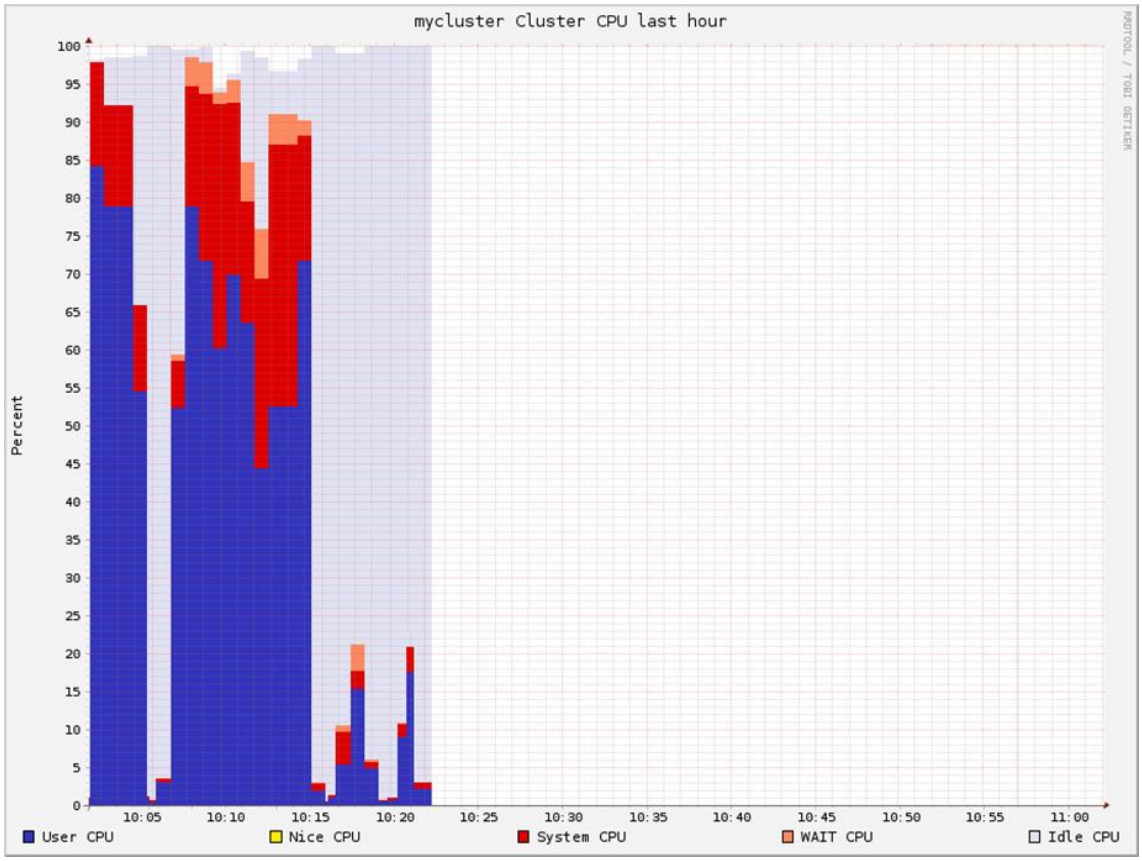
Terasort kıyaslama aracı bir giriş dizini içinde yer alan dosyadaki verileri sıralar ve çıkış dizinindeki bir dosyaya yazar [12, 13]. Terasort deneyinde reduce sayısı kıyaslama aracı çalıştırılırken bir parametre olarak verilmiştir. Teragen ile üretilen 10 GB'lik veri Terasort ile sıralanarak çıkış dizinindeki bir dosyaya yazılmıştır. Sıralama işlemi sonrasında işin doğruluğunu test etmek amacıyla Teravalidate çalıştırılmıştır. Terasort deneyinde 1, 2, 5 ve 10 düğüm üzerinde sırasıyla 1, 2, 5, 10, 20, 50 ve 100 reduce sayısı kullanılmıştır. Her reduce değeri için de sırasıyla 2, 10, 20, 50, 100 ve 150 map sayısı kullanılmıştır.

4. BULGULAR

CPU yoğun uygulamalar, CPU'yu %100'e yakın kullanma eğiliminde olan uygulamalardır. CPU'yu az kullanan uygulamaların CPU kullanımları %5 ile %15 civarında olma eğilimindedir [15]. Bu çalışmada deneyler CPU'yu az kullanan kıyaslama araçları ve CPU yoğun kıyaslama araçları şeklinde sınıflandırılarak yapılmıştır.

Şekil 4-1'de Ganglia izleme aracı ile elde edilen CPU kullanım yüzdeleri gösterilmektedir. 10:00-10:05 zaman aralığında 2 düğüm üzerinde 20 map kullanılarak yapılan Teragen deneyi CPU kullanım yüzdesi. 10:06-10:14 zaman aralığında 5 düğüm üzerinde, 5 reduce kullanılarak yapılan Terasort deneyi CPU kullanım yüzdesi. 10:16-10:19 zaman aralığında 5 düğüm üzerinde, 1 map kullanılarak yapılan Grep deneyi CPU kullanım yüzdesi. 10:20:30-10:21:20 zaman aralığında 1 düğüm üzerinde, 1 map kullanılarak yapılan Piestimator deneyi CPU kullanım yüzdesi. Bu grafikler işlem için harcanan sürenin az olduğu deneylerden seçilmiştir. Terasort ve Teragen deneyi sonuçları incelendiğinde her iki uygulamanın da CPU kullanımının %100'e yakın olduğu görülmektedir. Grep ve Piestimator deneyi sonuçlarına göre CPU kullanımının %15'e yakın olduğu görülmektedir.

Şekil 4-1'e göre Teragen ve Terasort kıyaslama araçlarının CPU'yu yoğun kullandıkları, Grep ve Piestimator kıyaslama araçlarının CPU'yu az kullandıkları görülmüştür.



Şekil 4-1 Teragen, Terasort, Grep ve Piestimator kıyaslama araçlarının Ganglia izleme aracı ile elde edilmiş CPU kullanım grafikleri (yüzde olarak).

4. 1. CPU'yu Az Kullanan Uygulamalar

Piestimator ve Grep kıyaslama araçları için yapılan deneylerin sonuçları grafikler ve tablolar halinde verilerek ayrı alt başlıklar altında incelenmiştir.

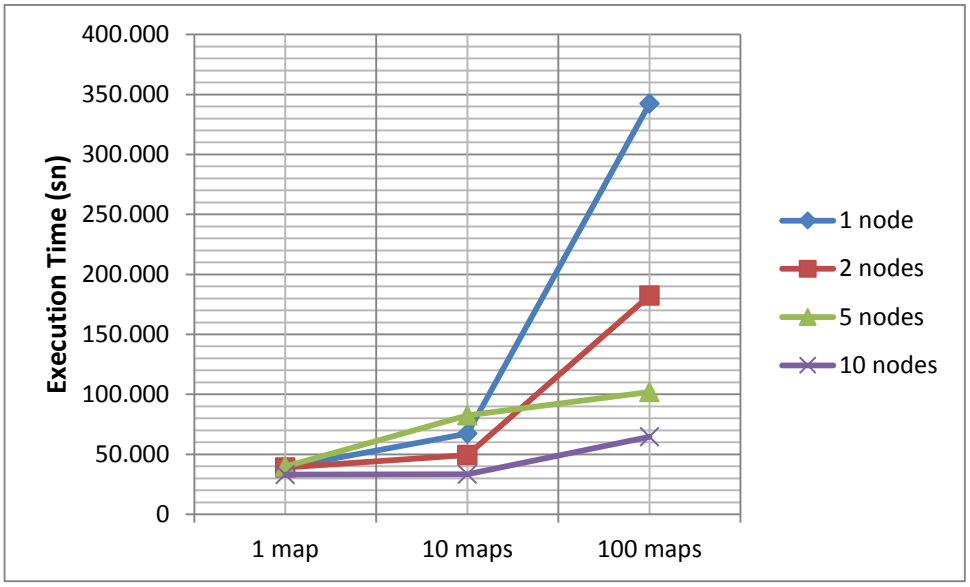
4. 1. 1. Piestimator kıyaslama aracı

Üretilen giriş sayısına bağlı olarak pi sayısını tahmin etmek için gerçekleştirilen Piestimator deney sonuçları Tablo 4-1' de sunulmaktadır. Giriş sayısının yüksek olması, tahmini yapılacak pi sayısının gerçek değerine daha yakın olmasını sağlamaktadır. Örneğin Piestimator deneyinde giriş sayısı 100 seçildiğinde pi sayısının yaklaşık tahmini değeri 3,16 iken; 10.000.000 seçildiğinde pi sayısının yaklaşık tahmini değeri 3,14 olarak bulunmuştur.

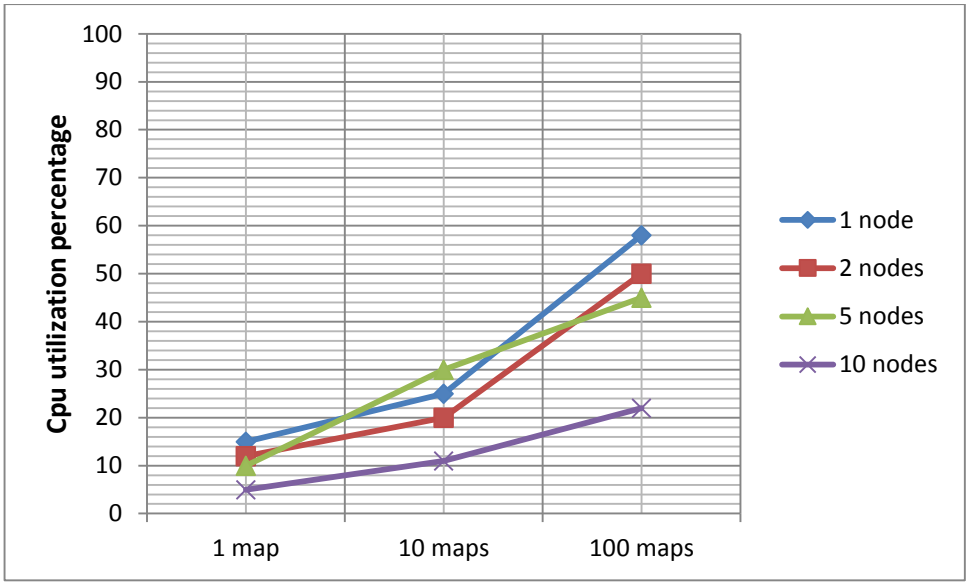
Farklı düğüm sayısı üzerinde, farklı map ve örnekleme sayılarında gerçekleştirilen Piestimator deneylerinde harcanan süreler Şekil 4-2'de sunulmaktadır. İşin tamamlanma süresinin 1 map ve 10.000.000 örnekleme sayısında tüm düğümlerde birbirine yakın olduğu görülmüştür. Map sayısının artırılmasının işlem süresini artırdığı gözlenmiştir. 10 düğüm üzerinde 1 map kullanıldığında işin tamamlanma süresi 33,018 sn olarak bulunmuştur. Bu değer deneylerdeki en düşük değerdir. 1 düğüm üzerinde 1 map kullanıldığında işin tamamlanma süresi 38,950 sn olarak bulunmuştur. 1 düğüm ile 10 düğüm arasında yaklaşık 6 sn'lik fark bulunmaktadır. Piestimator için kullanılan sistem kaynakları 10 kat artırılmasına rağmen işin tamamlanma süresinin %15 azaldığı görülmüştür. Yapılan kaynak artışına göre elde edilen başarım kazancı çok düşüktür. Aynı giriş sayısında işin başarımını etkileyen parametrenin map olduğu tespit edilmiştir. Şekil 4-3'de farklı düğüm sayısı üzerinde kullanılan map sayısına bağlı olarak CPU kullanımları gösterilmiştir. Map sayısının artırılmasının CPU kullanımını da artırdığı gözlenmiştir. CPU'yu az kullanan bir uygulamada CPU kullanımının artmış olmasının işin tamamlanma süresini azaltmadığı gibi aksine artırdığı görülmüştür. CPU kullanımının en az olduğu durumda işin tamamlanma süresinin en kısa olduğu gözlenmiştir.

Tablo 4-1 Piestimator kıyaslama aracı deney sonuçları.

Düğüm Sayısı	**Map Sayısı**	**Örnekleme Sayısı**	**İş için Harcanan Zaman (sn)**	**İşlemci Kullanımı %**
1	1	10000000	38,950	15
1	10	1000000	67,227	25
1	100	100000	342,554	58
2	1	10000000	39,037	12
2	10	1000000	49,296	20
2	100	100000	182,249	50
5	1	10000000	34,291	10
5	10	1000000	34,535	30
5	100	100000	97,677	45
10	1	100000000	33,018	5
10	10	1000000	33,242	11
10	100	100000	64,457	22



Şekil 4-2 Piestimator kıyaslama aracı farklı düğüm sayısı üzerinde, farklı map ve örnekleme sayılarında gerçekleştirilen deneylerde iş için harcanan zaman.



Şekil 4-3 Piestimator kıyaslama aracı farklı düğüm sayısı üzerinde, farklı map ve örnekleme sayılarında gerçekleştirilen deneylerde CPU kullanımı.

4. 1. 2. Grep kıyaslama aracı

1, 2, 5 ve 10 düğüm üzerinde 10 map ile sırasıyla 1, 2, 5, 10, 20, 50 ve 100 reduce sayısı kullanılarak gerçekleştirilen deneylerin sonuçları Tablo 4-2'de gösterilmiştir. Reduce

sayısı ile işlerin tamamlanma süresinin grafiği Şekil 4-4'te gösterilmiştir. Reduce sayısı ile CPU kullanımları grafiği Şekil 4-5'te gösterilmiştir. Şekil 4-4 ve Şekil 4-5 verilerine göre reduce sayısının artırılmasının işlem zamanını ve CPU kullanımını artırdığı gözlenmiştir. Benzer deneyler diğer düğüm ve map sayılarında da gerçekleştirilmiştir ancak reduce sayısının artırılmasının her deneyde işlem zamanını ve CPU kullanımını artırdığı gözlenmiştir. Bu nedenle farklı düğüm ve map sayısı ile gerçekleştirilen deneylerde gösterilen sonuçlar 1 adet reduce kullanımına göre sunulmuştur.

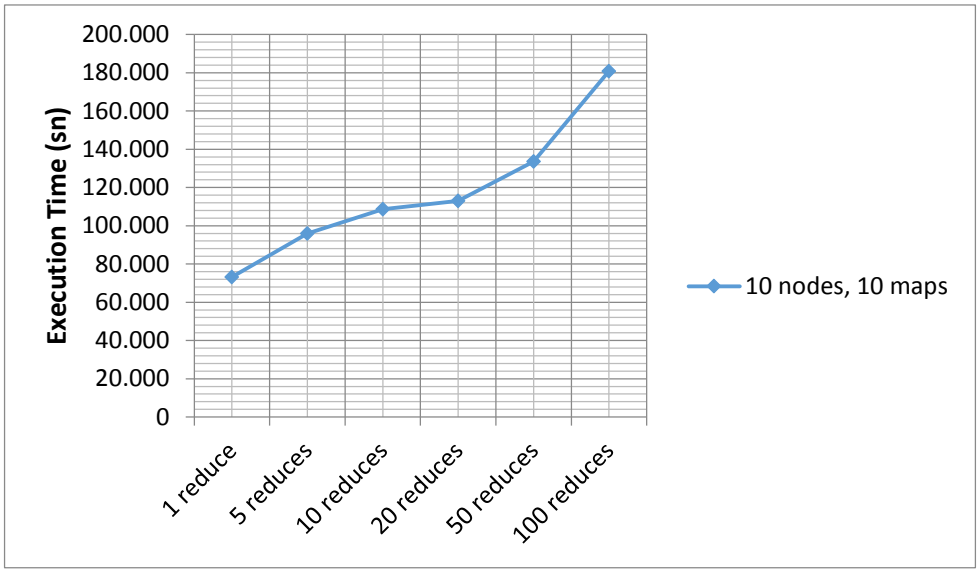
Tablo 4-3'de farklı düğüm, map ve 1 adet reduce sayısı kullanılarak gerçekleştirilen Grep deneylerinde iş için harcanan süreler ve CPU kullanımları sunulmuştur.

Şekil 4-5'de farklı düğüm ve map sayılarında gerçekleştirilen Grep deneyi sonuçları incelendiğinde işlerin tamamlanma süresinin başarımının aynı map ve reduce sayısında tüm düğümlerde birbirine çok yakın olduğu ve map sayısının artırılmasının işlem zamanını artırdığı gözlenmiştir. 1 düğüm, 1 map üzerinde 67,523 sn ile işin tamamlanma süresi en iyi olarak bulunmuştur. Düğüm sayısının arttırılmasının işlem zamanını azaltmadığı görülmüştür.

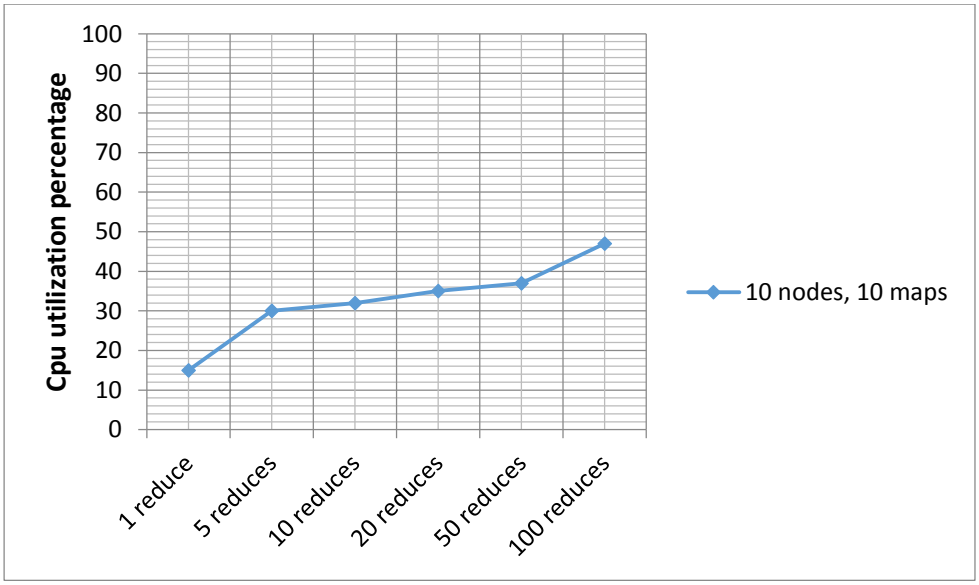
Şekil 4-6. verileri, farklı düğümler üzerinde, kullanılan map sayısına bağlı olarak CPU kullanımlarını göstermektedir. CPU'yu az kullanan bir uygulamada CPU kullanımının artışının işin tamamlanma süresini azaltmadığı gibi aksine artırdığı gözlenmiştir.

Tablo 4-2 10 düğüm üzerinde, 10 map ve farklı reduce sayısı kullanılarak gerçekleştirilen Grep deneyi sonuçları.

Reduce Sayısı	**İş İçin Harcanan Zaman (sn)**	**İşlemci Kullanımı %**
1	73,203	15
5	95,879	30
10	108,650	32
20	112,925	35
50	133,527	37
100	180,870	47



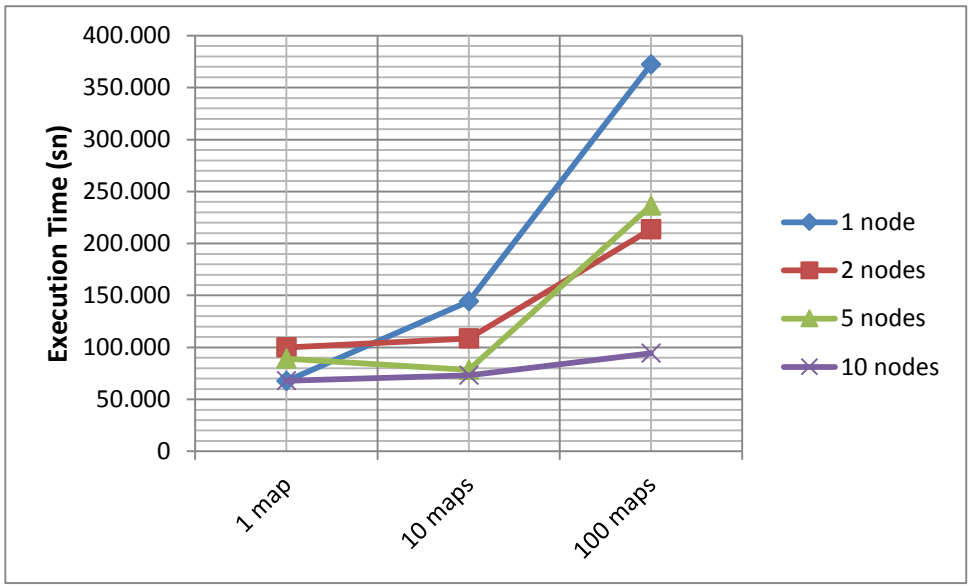
Şekil 4-4 Grep kıyaslama aracının 10 düğüm üzerinde 10 map ile farklı reduce sayıları kullanılarak gerçekleştirilen deneylerde iş için harcanan süreler.



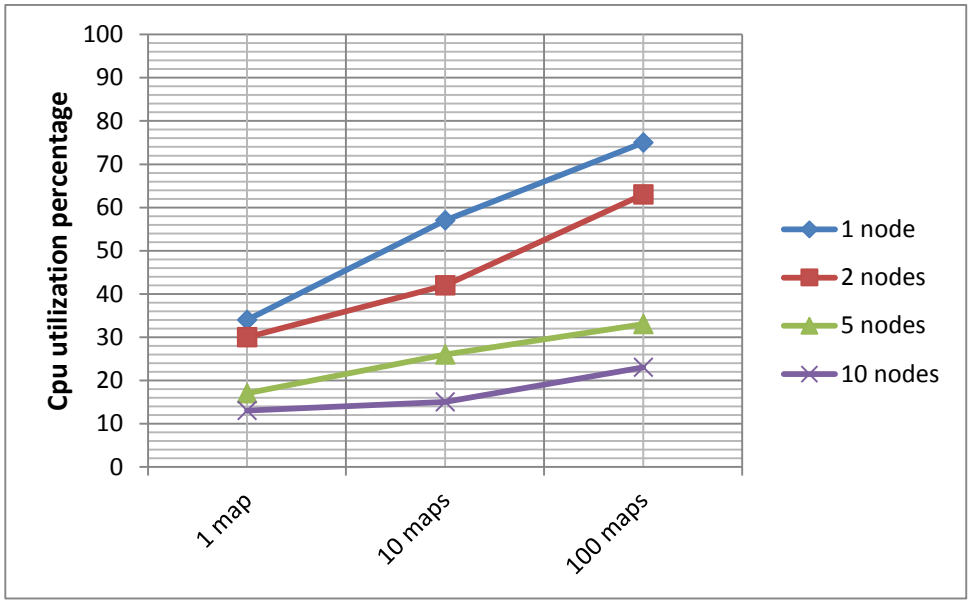
Şekil 4-5 Grep kıyaslama aracının 10 düğüm üzerinde 10 map ile farklı reduce sayıları kullanılarak gerçekleştirilen deneylerde CPU kullanım yüzdeleri.

Tablo 4-3 Grep kıyaslama aracının farklı düğüm ve map sayılarında ve 1 reduce sayısında gerçekleştirilen deney sonuçları.

Düğüm Sayısı	Map Sayısı	İş için Harcanan Zaman (sn)	İşlemci Kullanımı %
1	1	67,523	34
1	10	144,270	57
1	100	372,371	75
2	1	100,071	30
2	10	108,598	42
2	100	213,742	63
5	1	89,143	17
5	10	78,442	26
5	100	236,504	33
10	1	67,798	13
10	10	73,203	15
10	100	94,336	23



Şekil 4-6 Grep kıyaslama aracı ile farklı düğüm ve map sayılarında gerçekleştirilen deneylerde iş için harcanan süreler.



Şekil 4-7 Grep kıyaslama aracı farklı düğüm, map ve örnekleme sayılarında gerçekleştirilen deneylerde CPU kullanım yüzdeleri.

4. 2. CPU'yu Yoğun Kullanan Uygulamalar

Teragen ve Terasort kıyaslama araçları için yapılan deneyler, grafikler ve tablolar şeklinde verilerek ayrı alt başlıklar altında incelenmiştir.

4. 2. 1. Teragen kıyaslama aracı

Teragen deneyinde farklı düğüm ve map sayıları kullanılarak gerçekleştirilen deneylerde işin tamamlanma süresi ve CPU kullanımları Tablo 4-4'te gösterilmiştir.
Teragen kıyaslama aracında reduce aşaması çalıştırılmamaktadır. Bu nedenle bu kıyaslama aracı için reduce sayısına bakılmamıştır. Farklı düğüm sayısı üzerinde farklı map sayıları kullanılarak gerçekleştirilen deneylerde işin tamamlanma süreleri Şekil 4-8'de gösterilmiştir. Şekil 4-8 verileri incelendiğinde iş için harcanan zamanının en iyi olduğu durum düğüm sayısı kadar map sayısı seçildiğinde olduğu görülmüştür.

Deneylerin gerçekleştirildiği bulut sistemi üzerinde seçilen düğümlerin CPU sayısı 1 olduğundan en iyi durum CPU sayısı kadar map sayısı seçilmesidir.

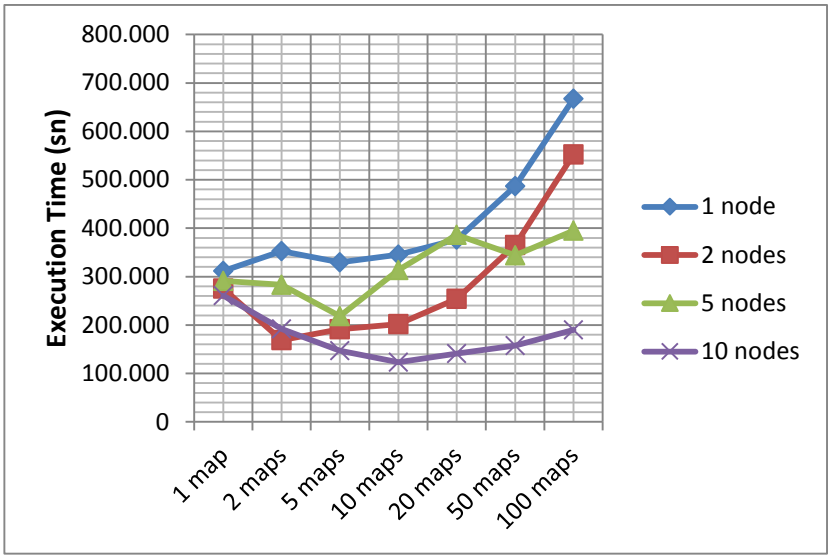
Şekil 4-9'da her deneyde iş için harcanan sürenin en az olduğu durum, CPU kullanımının en fazla olduğu durum olarak tespit edilmiştir. CPU'yu yoğun kullanan kıyaslama araçlarında CPU kullanımının en fazla olduğu durum sistemin eniyilemesi olarak bulunmuştur. Düğüm sayısının artırılmasının işlemin başarımını arttırdığı görülmüştür. 1 düğüm, 1 map sayısında işlemin tamamlanma süresi en iyi 311,755 sn iken yaklaşık %60 oranında verimlilik artışı ile 10 düğüm, 10 map sayısında en iyi 123,035 sn olduğu görülmüştür.

Teragen deneyinde sistemin eniyilemesi için düğüm ve map sayısı arasındaki ilişki Eşitlik 2'de sunulmaktadır. İfadede Mn map sayısını, Nn düğüm sayısını ve CPUn ise düğümde kullanılan CPU sayısını ifade etmektedir.

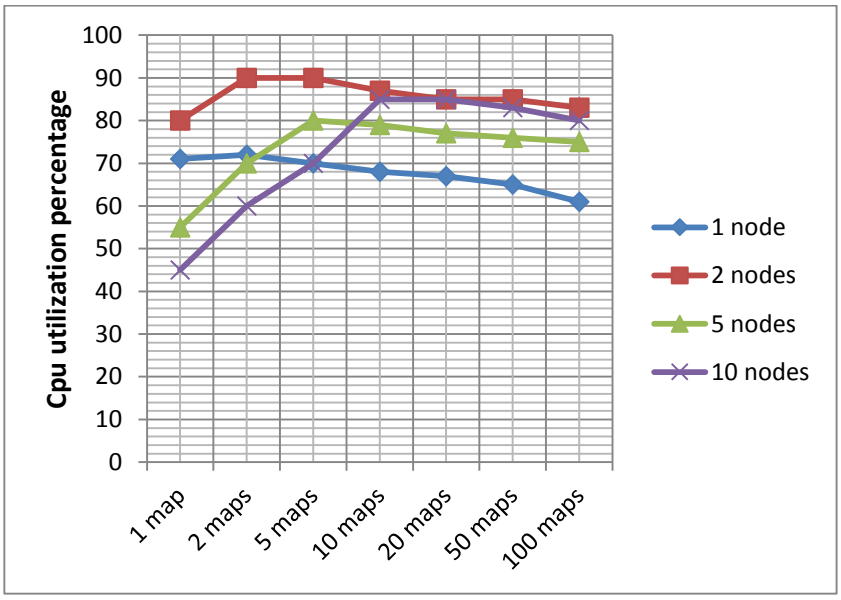
$$Mn = Nn \times CPUn \quad \textbf{(2)}$$

Tablo 4-4 Farklı düğüm ve map sayıları kullanılarak gerçekleştirilen Teragen kıyaslama aracı deneyi sonuçları.

Düğüm Sayısı	Map Sayısı	İş İçin Harcanan Zaman (sn)	İşlemci Kullanımı %
1	1	311,755	71
1	2	352,690	72
1	5	329,204	70
1	10	345,328	68
1	20	376,176	67
1	50	486,542	65
1	100	667,008	61
2	1	275,172	80
2	2	168,923	90
2	5	191,019	90
2	10	201,895	87
2	20	254,292	85
2	50	364,887	85
2	100	552,423	83
5	1	290,929	55
5	2	283,276	70
5	5	218,111	80
5	10	313,331	79
5	20	386,597	77
5	50	344,001	76
5	100	395,095	75
10	1	260,657	45
10	2	192,147	60
10	5	147,192	70
10	10	123,035	85
10	20	141,008	85
10	50	157,607	83
10	100	190,236	80



Şekil 4-8 Teragen kıyaslama aracının farklı düğüm ve map sayılarında gerçekleştirilen deneylerde iş için harcanan süreler.



Şekil 4-9 Teragen kıyaslama aracının farklı düğüm ve map sayılarında gerçekleştirilen deneylerde CPU kullanımları.

4. 2. 2. Terasort kıyaslama aracı

Terasort kıyaslama aracı deneyi sırasıyla 10, 20, 50, 100 ve 150 map sayısıyla farklı düğüm ve reduce sayısı kullanılarak gerçekleştirilmiştir. Farklı map sayısı kullanılarak

gerçekleştirilen deneylerde çalışma süresi ve CPU kullanımında büyük bir farklılık olmadığı görülmüştür. Tablo ve grafiklerde verilen sonuçların tümü HDFS dosya sisteminin varsayılan blok boyutuna (64 MegaByte) bağlı olarak 150 map olarak seçilmiştir.

Terasort kıyaslama aracı için farklı düğüm sayısı üzerinde, farklı reduce sayıları kullanılarak işin tamamlanma süresi ve CPU kullanımları Tablo 4-5'te gösterilmiştir.

Şekil 4-10 verilerine göre iş için harcanan zamanının en iyi olduğu durum düğüm sayısı kadar reduce sayısı seçildiği durumdur. Deneylerin gerçekleştirildiği bulut üzerinde seçilen sanal makinaların CPU sayısı 1 olduğundan en iyi durum CPU sayısı kadar reduce sayısı seçilmesidir.

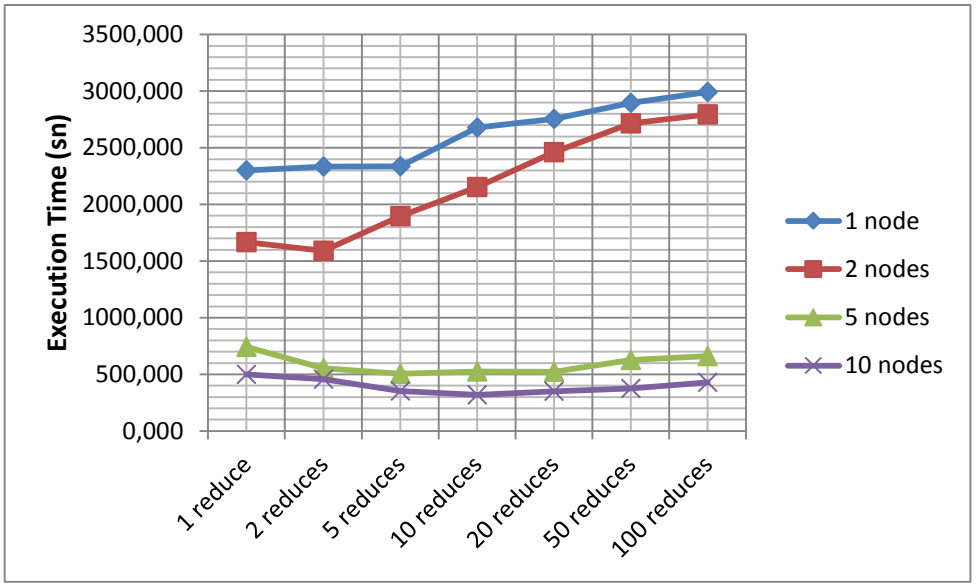
Şekil 4-11'de her deneyde iş için harcanan sürenin en az olduğu durum, CPU kullanımının en fazla olduğu durum olarak tespit edilmiştir. CPU'yu yoğun kullanan kıyaslama araçlarında CPU kullanımının en fazla olduğu durum sistemin eniyilemesi olarak bulunmuştur. Düğüm sayısının artırılmasının işlemin başarımını artırdığı görülmüştür. 1 düğüm ve 1 reduce sayısında işlemin tamamlanma süresinin en iyi 2299.372 sn iken yaklaşık %87 oranında verimlilik artışı ile 10 düğüm 10 reduce sayısında en iyi 318.537 sn olduğu görülmüştür.

Terasort deneyinde sistemin eniyilemesi için düğüm ve map sayısı arasındaki ilişki Eşitlik 3'de sunulmaktadır. İfadede Rn reduce sayısını, Nn düğüm sayısını ve CPUn ise düğümde kullanılan CPU sayısını ifade etmektedir.

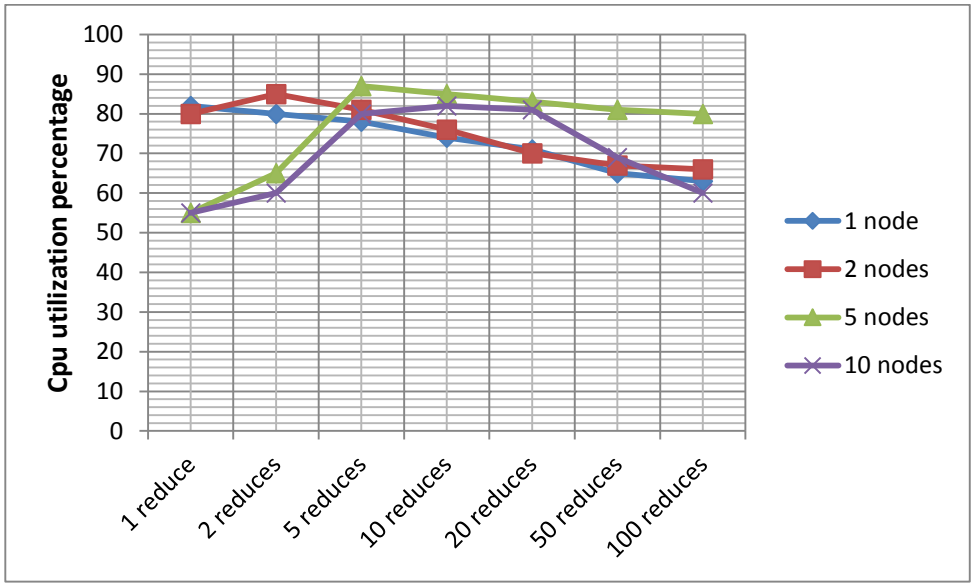
$$Rn = Nn \times CPUn \quad \textbf{(3)}$$

Tablo 4-5 Farklı düğüm ve reduce sayıları kullanılarak gerçekleştirilen Terasort kıyaslama aracı deneyi sonuçları.

Düğüm Sayısı	Reduce Sayısı	İş İçin Harcanan Zaman (sn)	İşlemci Kullanımı %
1	1	2299.372	82
1	2	2332.534	80
1	5	2336.753	78
1	10	2677.282	74
1	20	2756.057	71
1	50	2896.722	65
1	100	2991.293	63
2	1	1667.460	80
2	2	1590.587	85
2	5	1897.101	81
2	10	2151.697	76
2	20	2459.948	70
2	50	2714.507	67
2	100	2794.860	66
5	1	742.932	55
5	2	554.533	65
5	5	504.177	87
5	10	525.402	85
5	20	523.647	83
5	50	627.607	81
5	100	660.713	80
10	1	501.013	55
10	2	456.560	60
10	5	352.554	80
10	10	318.537	82
10	20	349.248	81
10	50	375.257	60
10	100	428.388	69



Şekil 4-10 Terasort kıyaslama aracının farklı düğüm ve reduce sayılarında gerçekleştirilen deneylerde iş için harcanan süreler.



Şekil 4-11 Terasort kıyaslama aracının farklı düğüm ve reduce sayılarında gerçekleştirilen deneylerde CPU kullanımları.

5. SONUÇ VE ÖNERİLER

Bu çalışmada; bir bulut sisteminde, KVM sanallaştırma yazılımı kullanılarak 10 düğümlü bir öbek oluşturulmuştur. Bu düğümler üzerinde Hadoop çatısı altında gelen kıyaslama araçları yardımıyla hazır veri setleri kullanılarak farklı map ve reduce sayısında deneyler gerçekleştirilmiştir. Deneylerde CPU kullanım durumuna göre düğüm, map ve reduce sayısının işin verimliliğini nasıl etkilediği incelenmiştir.

Alan yazında Hadoop çatısının başarımını ölçmeye yönelik çalışmalarda Hadoop çatısının başarımını ölçmek için çatıya ya da koda çeşitli eklentiler yapmak hem sisteme ek yükler getirmektedir hem de karmaşıklığı artırmaktadır. Çalışmada Hadoop çatısının performans artışını sağlamayacak şekilde düğüm, map ve reduce sayılarının işin başarımını ne kadar etkilediği ölçülerek, parametrelerin kullanım miktarları şu şekilde belirlenmiştir: Piestimator ve Grep gibi kıyaslama araçlarının deney sonuçlarına göre CPU'yu az kullandıkları görülmüştür. Az CPU gücü gerektiren bir işin dağıtık olarak yapılabilmesi için öbek kullanmaya gerek olmadığı söylenebilir. Tek düğüm üzerinde map işlemlerinin sayısı en az olduğunda işin en kısa sürede bittiği gözlenmektedir. Buradan, düşük CPU gücü gerektiren bir iş dağıtık olarak çok düğümlü öbek üzerinde yapılırsa işlerin ayrıştırılması ve toparlanması için harcanan süre, toplam işlem süresine dâhil olduğundan, toplamda harcanan süreyi kısaltmak yerine artırmaktadır. Buna göre CPU'yu az kullanan uygulamalarda sistemin eniyilemesi için düğüm, map ve reduce sayısı en az seçilmelidir.

Teragen ve Terasort gibi kıyaslama araçlarının deney sonuçlarına göre CPU'yu yoğun kullandıkları görülmüştür. CPU yoğun uygulamalarda düğüm sayısının ve CPU kullanımının artmasının işin verimliliğini artırdığı görülmüştür. İş parçacıklarının işlendiği aşamaya göre map ya da reduce sayısının, düğümlerdeki toplam CPU sayısı kadar seçilmesi iş için harcanan zamanın eniyilemesi olarak bulunmuştur.

Hadoop çatısının başarımını artırmak için başka parametreler de bulunmaktadır. Gelecek çalışmalarda düğüm sayısı ile bu parametreler beraber kullanıldığında işin başarımını daha da artırmak mümkün olabilir. Kullanılan büyük verinin, kıyaslama araçlarının ürettiği veri ile sınırlı olduğu göz önüne alınarak, farklı veri setleri ile belirlenen parametreler kullanılarak işin başarımı tekrar ölçülebilir.

KAYNAKLAR

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

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

[3] 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 Ocak. 2013; Bangkok, Tayland. New York, NY, USA: IEEE. pp. 28-30.

[4] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM 2008; 51: 107-113..

[5] HDFS Architecture, http://hadoop.apache.org/docs/r1.0.4/hdfsdesign.html, erişim tarihi 5 Mart 2014.

[6] Pavlo A, Paulson E. A comparison of approaches to large scale data analysis. In: SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data; 29 June 2009 – 02 July 2009; Providence, RI, USA. New York, NY, USA: ACM. pp. 165-178.

[7] Kim K, Jeon K, Han H, Kim SG. Mrbench: A benchmark for mapreduce framework. In: 14th IEEE International Conference ICPADS'08 on Parallel and Distributed Systems; 8-10 December 2008; Melbourne, VIC, Australia. New York, NY, USA:IEEE. pp. 11-18.

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

[9] Sammer E. Hadoop Operations. CA, USA: O'Reilly Media, 2012.

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

[11] Massie ML, Chun BN, Culler DE. The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing 2004; 30: 817-840.M. L.

[12] O. O'Malley, TeraByte Sort on Apache Hadoop, http://sortbenchmark.org/YahooHadoop.pdf, 2008, erişim tarihi 12 Haziran 2014.

[13] Maurya M, Mahajan S. Performance Analysis of MapReduce Programs on Hadoop Cluster. In: 2012 World Congress on Information and Communication Technologies (WICT); 30 October 2012-02 November 2012; Trivandrum, Indian. New York, NY, USA: IEEE. pp. 505-510.

[14] Grama A, Gupta A, Karypis G, Kumar V. Introduction to Parallel Computing (Second Edition). Boston, USA:Addision Wesley, 2003.

[15] Barney B. Introduction to Parallel Computing, https://computing.llnl.gov/tutorials/parallel_comp/, değiştirme tarihi 13 Kasım 2014, erişim tarihi 12 Ocak 2015.

[16] Distributed and Parallel Computing, http://wla.berkeley.edu/~cs61a/fa11/lectures/communication.html#the-problem-with-shared-state, erişim tarihi 15 Ocak 2015.

[17] Miner D, Shook A. MapReduce Design Patterns. CA, USA: O'Reilly Media, 2012.

[18] Kiranjot K, Anjandeep KR. A comparative analysis: grid, cluster and cloud computing. International Journal of Advanced Research in Computer and Communication Engineering 2014;3(3):5731-5734.

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

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

[21] Apache Hadoop, http://wiki.apache.org/hadoop, erişim tarihi 08 Eylül.2014.

[22] Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST); 3-7 May 2010; Incline Village, NV. New York, NY, USA: IEEE. pp. 1-10.

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

[24] Massie M, Li B, Nicholes B, Vuksan EV. Monitoring with Ganglia. CA, USA: O'Reilly Media, 2012.

[25] Ganglia Monitoring System, http://ganglia.sourceforge.net/, erişim tarihi 20 Eylül 2014.

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

[27] Premchaiswadi W, Romsaiyud W. Optimizing and Tuning MapReduce Jobs to Improve the Large-Scale Data Analysis Process. International Journal of Intelligent Systems 2013; 28: 185-200.

[28] 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 (CIDR'11); 9-12 January 2011; Asimolar, CA. pp. 261–272.

[29] Verma A, Cherkasova L, Campbell RH. ARIA: automatic resource inference and allocation for MapReduce environments. In: Proceedings of the 8th ACM International Conference on Autonomic Computing; 14-18 June 2011; Karlsruhe, Germany. New York, NY, USA: ACM. pp. 235–244.

[30] Zhang Z, Cherkasova L, Loo BT. Benchmarking approach for designing a MapReduce performance model. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE'13); 21-24 April 2013; Prague, Czech Republic. New York, NY, USA: ACM. pp. 253–258.

[31] Babu S. Towards automatic optimization of mapreduce programs. In: Proceedings of the 1st ACM symposium on Cloud computing; 10-11 June 2010; Indianapolis, Indiana, USA. New York, NY, USA: ACM. pp. 137–142.

[32] Fadika Z, Dede E, Govindaraju M, Ramakrishnan L. Benchmarking MapReduce Implementations for Application Usage Scenarios. In: 12th IEEE/ACM International Conference, Grid Computing (GRID); 21-23 September 2011; Lyon, France. New York, NY, USA: IEEE. pp. 90-97.

[33] DigitalOcean Cloud Service, http://www.digitalocean.com, erişim tarihi 17 Ağustos 2014.

[34] J. Venner, Pro Hadoop, Apress, USA, 2009.

[35] Gutenberg Project, http://www.gutenberg.org, son erişim 12 Temmuz 2014.

[36] White T. Hadoop Definitive Guide. 2nd Edition. Sebastopol, CA, USA: O'Relly Media, 2010.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı, Soyadı:	Göksu Zekiye ÖZEN
Uyruğu	Türk
Doğum Tarihi ve Yeri:	01-07-1980 Bakırköy
Medeni Durumu	Evli
Tel:	+996 702 515839
Fax:	------
Email	goksu@goksu.info
Yazışma Adresi	------

EĞİTİM

Derece	**Kurum**	**Mezuniyet Tarihi**
Yüksek Lisans	Kırgızistan-Türkiye Manas Ü.	……..
Lisans	Kocaeli Üniversitesi.	2003
Lise	Bağcılar Teknik Lisesi	1998

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görev**
2001	B&M Bilişim Hizmetleri.	Staj
2004	Körfez Milangaz Hacer Demirören ÇPL.	Öğretmen
2009	İzmit Kız Teknik ve Anadolu Meslek Lisesi	Öğretmen
2011	Kartepe Teknik ve Endüstri Meslek Lisesi	Öğretmen
2012	Kırgız-Türk Anadolu Kız Meslek Lisesi	Öğretmen

YABANCI DİL

Kırgızca
İngilizce