T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK VERİ ANALİZİNDE HARİTALAMA (MAPPİNG) İNDİRGEMESİ İÇİN YENİ BİR YÖNTEM GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Suat ERDOĞAN

Enstitü Anabilim Dalı	**:**	**ENDÜSTRİ MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Safiye TURGAY**

Haziran 2020

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.

Suat ERDOĞAN

19.06.2020

TEŞEKKÜR

Hayat boyunca desteğini benden esirgemeyen sevgili annem Hatice ERDOĞAN, kız kardeşim Gamze ERDOĞAN'a tüm kalbimle teşekkürlerimi sunarım. Ayrıca yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Doç. Dr. Safiye TURGAY'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

5V	: Büyük veri yapısının 5 özelliği.
HDFS	: Hadoop distribted file system.
YARN	: Hadoop yarn.

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Harita indirgemesi ve sıralanması, Büyük veri analizi, Harita sınırlama, Harita indeksleme, Harita verilerinin değer noktası

Günümüzde hızla gelişen teknoloji ile birlikte veri hacmi ve veri paylaşımları da her geçen gün artmaktadır. Büyük verilerin daha hızlı ve etkin işlenerek analiz edilmesi sürecinde, veri haritalama ve indirgenmesi oldukça önemlidir. Büyük veri analizinde veri haritalama dizisi ve küçültme, belirli bir algoritma yapısı kullanarak çalışır ve girdileri bir değer listesine, parametre olarak gönderir. Ara sonuç listesi için girilen sistemde yer alan listedeki tüm değerler dönüştürülerek oluşturulur. Haritalama (Map) işleminde, haritalama listesinin yapısında, veriler kapladıkları alan ve tekrar sayıları dikkate alınarak hızlı sıralama işlemlerine tabi tutulur. Az miktarda olan verinin işlenmesi daha az zaman tüketimi, bellek tüketimi, işlemci tüketimi ve disk tüketimi gibi konularda maliyet azaltıcı etki göstermektedir. Bu çalışmada, önerilen algoritma ile veri sıralaması, veri azaltımı işlemleri daha etkin bir şekilde gerçekleştirilmiştir. Algoritmanın analiz sonuç değerleri sonuç kısmında verilmiştir. Sistem içerisinde veri analizine hazırlanacak olan veriler öncelikle sıralanabilir özellikleri kontrol edilmiş ve daha sonra işlem uygulanmıştır. Çok miktarda veri olması durumunda, veriler çok daha fazla maliyet ile işlenecektir. Azaltım uygulanacak veriler, veri büyüklüğüne ve değerliğine sahip yapıları dikkate alınarak azaltma işleminin her veriye uygulanması sağlanmıştır. Bu işlem örneklerin seçimini kolaylaştırmıştır. Bu işlem ile aynı zamanda örneklerin seçim işlemi de kolaylaşmıştır.

Tez içinde amaçlanan yapıyı gerçekleştirecek bir yazılım geliştirilmiştir. Yazılım Hadoop içindeki sınıflar kullanılarak önerilen algoritma yapısına göre düzenlenerek yeni bir altprogram oluşmuştur. Bunun için Hadoop kütüphanesinden yararlanılmıştır. Çalışma içinde yazılıma aktarılan değer dosyası öncelikle belli sınırlara indirgenmiş ve ardından sıralama ve haritalama yapılmıştır. Haritalama çıktısı içerisinde indeksleme yapılmıştır. Paralel olarak çalışan haritalama ve indirgeme sınıflarında, indirgeme aşamasında veriler değerlendirilmiş ve değerlendirme sonucu oluşan bir çıktı dosyası üretilmiştir.

DEVELOPING A NEW METHOD FOR MAPPING DOWNLOAD IN LARGE DATA ANALYSIS

SUMMARY

Keywords: Map reduce and sorting, Big data analysis, Map bloced, Map index, Map's value point

Today, with the rapidly developing technology, data volume and data sharing are increasing day by day. Data mapping and reduction is very important in the process of analyzing the big data respect of the faster and more efficiently. In big data analysis, data mapping sequence and reduction is worked by using a certain algorithm structure and introduced then sended the inputs to a value list as a parameter. The intermediate result list is created by converting all values in the list included in the entered system. Time of the mapping (Map) process algorithm developed in the structure of the mapping list divide and obtain operations are performed. The sort depends on the bayt value that each data generates. In the case of small volumes of data, the data's result cost reduction, have less time consumption, memory consumption, processor consumption, and disk consumption. In this study, a more effective analysis process has been carried out with the proposed data sorting and reduction algorithm. The data to be prepared for data analysis in the system must have a sortable feature. If there is a large amount of data, the data processed at a much higher cost. The data mitigated must have data size and value, so that the reduction can be applied to each data. This can be facilitated the selection of examples. This process also facilitates the selection of the samples.

The software has been developed to perform the intended structure in this thesis. The software was formed by crushing, added new codes as a procedure and reorganizing classes in Hadoop. Hadoop's library was used for this purpose. In the study, the value file transferred to the software is reduced to certain limits, then sorting and mapping is performed. Indexing used in the printout with the map. Data is evaluated during the reduction phase with paralel running map and reduction classes, and an output file consisting of the evaluation result is produced.

BÖLÜM 1. GİRİŞ

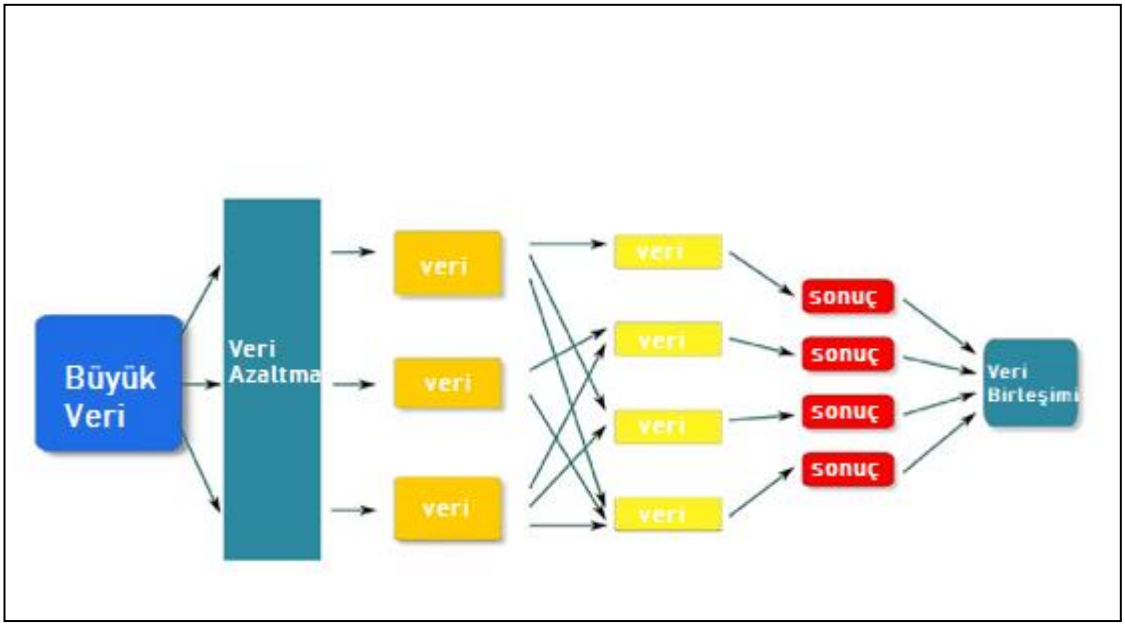
Yaşantımızda, bilgisayar ve internetin yaygın olarak kullanılması en zaruri, ihtiyaçlardan birisi haline gelmiştir. Beraberinde, veri yoğunluğu ve trafiği de çok hızlı bir biçimde, artmaya başlamıştır. Bu yoğun veri akışı içerisinde verilerin analiz edilmesi ve değerlendirilmesi oldukça önemlidir. Veri haritalama, verilerin özelliklerine göre sıralanması ve verilerin indirgenmesi, veri analizi içerisinde en önemli kavramlar arasında yer almakta ve geliştirilen sıralama algoritması ile veri analizine önemli bir katkı sağlamaktadır. Veri kavramı çok eskilere dayanıyor olsa da teknoloji bizlere bu kavramı kullanmamızda daha fazla zorunluluk oluşturmuştur. Günümüzde yaşanan teknolojinin hızlı gelişimi bu kavramında hayatımızın her alanında bizle birlikte olmasını sağlamıştır.

Günümüzde yoğunlaşan bu veriler ayrıca artan veri saklama imkanları ile veri kapasitelerinin inanılmaz boyutlara ulaşmasını sağlamıştır. Büyük veri, sosyal medyadan, bulut teknolojilerinden ve benzeri birçok noktadan sürekli, dinamik olarak durmadan artan yığındır. Verilerin saklanmasının zaman içinde maliyetinin azalmasının yanı sıra, kapasitenin her geçen gün geometrik olarak artması ile günümüzün sürekli çözümsel yaklaşımlarına karşı, teknolojinin etkin kullanılmasına rağmen, sürekli artan bir hızda çözümlenemeyen bir çözümdür.

Büyük veri, bilginin sınırlarından veya fazlalığından ziyade bu bilgiyi nasıl anlamlı hale getirileceği ile ilgilenmektedir. Kaynaklardan elde edilen verilere uygulanan bazı algoritma ve yöntemler ile daha anlamlı ve işlenebilir cevapların bulunması amaçlanmaktadır.

Haritalama ve indirgeme ilk olarak Google tarafından geliştirilmiş ve yapısal olarak iki temel parçanın paralel olarak uygulanmasından oluşan bir mimari modeldir. Bu

uygulamalar ise haritalama ve indirgeme (küçültme) işlemleridir. Kullanıcı tarafından sağlanan girdi, haritalama işlemi ile veri kümelerine dönüştürülür ve sonrasında analiz edilir. Uygulama işleminde ise haritalama işlemleri tarafından elde edilen çıktılar toplanır ve kullanıcı talebine göre şekillendirilir ve karar kuralı yapısı elde edilir. Hadoop çalışma zamanı, ortamı sağlar ve geliştiricilerin yalnızca giriş verilerini sağlaması, haritayı belirtmesi ve yürütülmesi gereken işlevlerin azaltmasını sağlamaktadır. Böylece verinin, zamana bağlı olan ve olmayan maliyetlerini azaltmaktadır. Apache Hadoop projesinin sponsoru Yahoo!, projeyi veri işlemek için ve hazır bir bulut bilişim platformuna dönüştürmek için büyük çaba harcamıştır. Hadoop bulut sistemi ile veri yığınlarının analizinde mükemmel bir araca dönüşmüştür. Yahoo! akademik kurumlar içinde mevcut olan dünyanın en büyük Hadoop kümesini elinde bulundurmaktadır. Bu çalışmada bu yüzden Yahoo'nun Hadoop yapısı incelenerek, indirgeme algoritması bu uygulama için geliştirilmiştir.
Büyük veri ve veri analizi birbirini tamamlayan iki temel bileşendir. Veri sınırlama, analiz öncesi maliyet azalımı için kullanılmıştır. Genel yaklaşımımızda büyük verilerin işlenmesi sürecinde aşağıda yer alan hedefler dikkate alınmıştır. Veri azaltma işlemi, yapılan bu çalışmada ile Şekil 1.1.'de olduğu gibi kullanılmıştır.



Şekil 1.1. Veri indirgeme işleminde izlenen yol.

Bunlar ise;

1. Gerçek zamanlı verilerin temizlenmesi ve istenmeyen her türlü veriden ayıklanması ve bu sayede her türlü donanımda daha az maliyetle işlenmesi
2. İşlenmeden önce gereksiz verilerin tespit edilmesi
3. Daha önemli verilerin öncelikli olarak ele alınması
4. Daha az maliyet gerektiren verilerin öncelikle ele alınması.

1.1. Amaç

Bu tezde verilerin en az maliyetle işlenmesi ve sıralanması amaçlanmıştır. Literatür çalışmalarda ortaya çıkan haritalama ve indirgeme ve sıralanmasına yeni bir yaklaşımla, veri maliyeti göz önünde tutularak katkı sağlamasına çalışılmıştır. Büyük verinin hiç durmadan artan miktarı kaçınılmaz sorunları beraberinde getirmektedir. Veri analizi maliyetini düşürmek için uygun algoritmalar ile veri azaltım yöntemleri kullanılarak sıralanması ile değerli olan veriye ulaşım şansını arttırmaktadır. Ayrıca değeri yüksek olan verilerin, bir veri setinde öncelikli olarak ele alınması için indekslenmesi ve puanlanması, yüksek verilerin ön plana çıkmasında sıralama aşamasında çok önemli ve kritiktir. Bu çalışmada sıralama ve indirgeme işlemini etkin bir biçimde gerçekleştiren algoritma geliştirilmiş ve Hadoop kütüphanesinde yer alan alt program yazılmıştır. İşlem öncesinde haritalama aşamasından önce veriler indirgenmiş (azaltılmış) ve ardından sıralama işlemine geçilmiştir. Her veri kendi boyutuna göre sıralanmış ve tekrarlı veri sayısına göre bulunarak puanlanarak indekslenmiştir. Bu puanlama ve sıralama aşamasından sonra indirgenen devredilen veriler burada önem derecesine göre dört parçaya ayrılmış ve çıktılar elde edilmiştir.

1.2. Kapsam

Bu tez çalışması 6 bölümden oluşmaktadır. 2. bölümde daha önce bu alanda yapılmış çalışmalara değinilmiştir. Bu bölümde aynı zamanda, büyük verinin özellikleri ile

veri sınırlama ve bunun öneminden bahsedilmiştir. 3. bölümde tezde kullanılan ise sınırları tespit edilen verinin sıralanmasından bahsedilerek, veri haritalama ve indirgeme hakkında bilgi sunulmuştur. 4. bölümde sıralama yaklaşımı ele alınmıştır. 5. bölümde yapılan uygulamanın işlem basamakları ve çıktılar verilmiştir. 6. bölümde sonuç kısmında yer almaktadır.

BÖLÜM 2. KAYNAK ARAŞTIRMASI

Büyük veri analizi her geçen gün artan veri trafiğinde artan bir öneme sahiptir. Özellikle bu verilerin saklanması, depolanması ve analiz edilmesi, bu verilerden anlamlı sonuçların çıkartılması açısından oldukça önemlidir. Bununla birlikte, sosyal medyadan, bulut bilişim uygulamalarından sürekli ve dinamik bir biçimde her geçen gün üstel olarak artan bu verilerin saklanması da ayrı bir problemi beraberinde getirmektedir. Günümüzde teknolojinin hızlı bir şekilde gelişmesine karşın çok daha hızlı artan bu verilerin saklanması, depolanması ve analiz edilmesi de yakın zamanda çözümlenemeyen bir problem olarak karşımıza çıkmaktadır.

Veri haritalama ve indirgeme ile ilgili olarak yapılan bazı çalışmalara örnek olarak, Dean ve Ghemawat haritalama ve indirgeme uygulamasını büyük bir emtia makinesi kümeleri ile birlikte çalışarak gerçekleştirmiştir. Amaç yüksek ölçeklenebilirlik ile yapılan hesaplama işlemlerinde binlerce makinede olan verilerin işlenmesidir. Yapıda birçok haritalama ve indirgeme programı uygulanmış ve işlemler her gün Google'ın kümelerinde yürütülerek uygulanmıştır [1]. Yang, Parker ve arkadaşları haritalama-indirgeme ve birleştirme fonksiyonlarını içeren yeni bir model geliştirmiş ve haritalama-indirgeme işlemini de geliştirerek, harita birleştirme aşamasını azaltmış ve zaten bölümlenmiş, sıralanmış verileri verimli bir şekilde bir araya getirerek yeni bir harita yapısı oluşturmuşlardır [2]. Zaharia ve arkadaşları, Hadoop'un performansını, küme düğümlerinin homojen olması sayesinde görevler doğrusal ilerleme sağlamış ve bu varsayımların üzerinde çalışmışlardır. Homojenlik varsayımları her zaman tutmasa da bu yeni bir zamanlama algoritması ile yeni yapısal homojen ortamlarda çözümler önermişlerdir [3]. İndeksleme, arama motoru ve içine veri işleme fonksiyonlarını çözüm yollarını keşfetmek için Hadoop haritalama ve indirgeme ve sıralama yapısını kullanılarak verileri verimli bir şekilde haritaya ayırabilen ve modülleri azaltabilen birleştirme aşamasını haritalama vw indirgeme aşamasına eklemişlerdir. Bu işlem

esnasında kullandıkları fonksiyonlar ise, çaprazlama, haritalama ve indirgeme ve birleştirme fonksiyonlarıdır. Bu verimli dizin dosyaları çapraz geçişler ile veri bölümlerini seçer, sayıları sınırlar ve haritalama için verinin azaltılması ve birleştirilmesini savunmuştur [4]. Kolb ve arkadaşları, haritalama-indirgeme programlama modelini, paralel nesne çözümü için kullanarak zorlukları ve olası çözümler için çok girişli sıralama ile en yakın komşu yaklaşımını kullanarak engellemeyi amaçlayan araştırmaları belirlemişlerdir [5]. Verma ve arkadaşları, haritalama-indirgeme modeli arasında bir engel kullanarak harita ve küçültme aşamalarında, iki parçalı programlama ve uygulama ile her ikisinde de basitlik ve kolaylık sağlamıştır [6]. Bununla birlikte birçok durumda, bu engel durumu performansı arttırır. Haritalama ve indirgemedeki engelleri ortadan kaldırarak, verimliliğini artıracak şekilde bir yöntem geliştirmek amaçlanmıştır. Goodrich ve arkadaşları haritalama ve indirgeme işlemini algoritmatik bir bakış açısıyla araştırmış ve en uygun çözümlerle yaklaşarak sıralama ve çoklu aramada en uygun benzetim modelini tasarlamıştır [7]. Herodotou ve Babu haritalama ve indirgeme işleminde maliyeti minimize etmek için (what if) tahmin motorunu önermiş ve tüm bileşenler için prototip elde etmiştir [8]. Hadoop haritalama-indirgeme sistemindeki her bileşenin etkinliği bir temsilci kullanılarak kapsamlı bir değerlendirme ile gösterilmiştir. Holmes [9] Hadoop yazılımını geliştirerek problem/çözüm biçimi ve kıyaslama paketi geliştirmiş ve haritalama-indirgeme programlarının çeşitli uygulama etki alanlarından bahsetmiştir. Bu uygulama özellikleri ile yüksek/düşük hesaplama ve yüksek/düşük karıştırma hacimlerini dikkate alarak değerlendirme kriterleri geliştirmiştir. Hadoop dağıtımının bir parçası olarak değil dağılımlar ve kriterler, gerekli görevleri azaltmak için geliştirmişlerdir. Slagter ve arkadaşları, yük dengeleme ve bellek tüketimine göre geliştirilmiş bir bölümleme işlemini ele alan, yeni bir algoritma geliştirilmişlerdir. Önerilen algoritmanın sonucu olarak daha hızlı, daha fazla bellek ve daha doğru sonuçlar ortaya çıkmaktadır [10]. Gopalani ve Arora büyük very yapısıyla ilgili dünya çapında bir araştırma yapmışlardır. Hadoop haritalama-indirgeme ve yakın zamanda tanıtılan Apache Spark karşılaştırılmıştır. Bu seçeneklerin her ikisi de büyük veri kavramına dayanmakta ve birlikte performansları uygulama altındaki kullanım durumuna göre önemli ölçüde değişiklik göstermektedir [11]. Bu araştırmacılar, bu iki seçeneği kümeleme için standart bir makine öğrenme algoritması (K-en yakın komşu)

kullanarak performans analizlerini karşılaştırmıştır. Gerçek zamanlı senaryoda, kullanılan verilerin hacmi zamanla doğrusal olarak artması durumu incelenmiştir. Sosyal medya başta olmak üzere kontrol edilemeyen verilerin büyümesi sorunsal durumunu yönetmek için büyük miktarda oluşan veri, önerilen yöntem ile dağıtılmış kümeler üzerinden küçük parçalara ayrılmıştır. Paralel olarak veri işlenecek ve sonrada işlenmiş veri kümeleri aralarında toplanmıştır. Haritalama, indirgeme, filtreleme, toplama görevini gerçekleştirmek ve verimli depolama yapısını korumak için kullanmıştır. Önerilen yöntem doğal dil yoluyla duygu analizi gibi teknikler kullanılarak geliştirilmiştir, veriler belirteçlere ve ifade tabanlı kümelere ayrıştırılmıştır. McCreadie ve arkadaşları, haritalama ve indirgemeyi yoğun veri dağıtımı için bir çerçeve olarak önermiştir. Haritalama-indirgeme endeksleme stratejisi değerlendirilmiştir [12]. Sonuç aktarımın en aza indirgenmesinin önemi kanıtlanmıştır.

Hadoop platformu çeşitli uygulama alanlarına ait büyük verilerin analiz sürecinde de kullanılmıştır. Bunlara örnek olarak Bakratsas ve arkadaşları Hadoop platformunu sosyal analizi için kullanmışlardır [13, 37]. Bununla birlikte, Auradkar ve diğerleri Mekansal Hodoop uygulaması ile büyük veri analizinde harita azatlımı uygulaması kullanılmıştır [14]. Lu ve Feng Hadoop'ta resim dosyalarının depolanması ve Hadoop'un resim dosyalarını işleyebilmesi için giriş ve çıkış formatlarının tanıtılması sürecini ele almıştır [15]. Babar ve arkadaşları Hadoop tabanlı akıllı kentsel veri yönetim sistemi önermişlerdir. Bu veri yönetim sistemi özellikle hava kirliliği verilerinin anlık işlenmesi ve hangi saatlerde havanın kirliği olduğu hakkında bir uyarı vermesi şeklinde çalışmaktadır. Bu çalışmada aynı zamanda paralel algoritma kullanılarak verimli veri yüklemesi sağlanmıştır [16]. Tallada ve diğerleri Büyük kozmik veri kümelerinin etkileşimli keşfi ve dağıtımı açısından, Hadoop tabanlı bir web uygulaması geliştirilmiştir. CosmoHub ile astronomik verilerin analizi hedeflenmiştir [17].

Bazı çalışmalar ise tamamen Hadoop programının çalışması esnasındaki enerji tüketimi üzerine yoğunlaşmıştır. Bu çalışmalara örnek olarak Hadoop kullanımındaki

enerji tüketimini CPU kullanım sıklık aralığına göre analiz edilmiştir [18, 19]. Wu ve diğerleri, Hadoop'un enerji verimliliği incelenmiştir [20].

Bazı çalışmalar ise veri madenciliği uygulamaları ile birleştirmiştir. Bunlara örnek olarak, Pradeep ve Sundar QUAC mimari yapısını önererek sorgu, kullanıcı ve veri yönetimi modüllerini ele alınarak, bellek kullanımı ve işlem hızını artıran bir mimari yapı tasarlanmıştır. Hadoop'ta güvenli veri depolaması amacıyla bir dizi küme oluşturulmuş ve yönetilmiştir [21]. Bellek kullanımı, zamanlama ve işlem süresi açısından fonksiyonların değerlendirilmesi süreci ele alınmıştır.

Map-Reduce (MR), büyük veri kümelerini işleme yeteneği nedeniyle, tanıtımından bu yana çok popüler hale gelen dağıtılmış bir programlama çerçevesidir. MR, paralel programlamanın birçok yönetim yönü hakkında endişelenmeden dağıtılmış uygulamaları uygulamak için sağlam ve basit bir mekanizma sağlar (örn. İşleri başlatmak, veri dağıtımı, iş senkronizasyonu). Diğer yandan, basitlik ve esnekliği ile Kaynak Tanımlama Çerçevesi (RDF), web-semantiği temsil etmek için çok önemli olan yarı yapılandırılmış ve yapılandırılmamış verileri temsil edebilir. SPARQL, RDF formatında saklanan verileri almayı ve değiştirmeyi amaçlayan bir sorgu dilidir ve ayrıca "Büyük Veri" uygulamalarını desteklemektedir [22]. Büyük veriler, yakalamak, yönetmek, depolamak, dağıtmak ve yüksek hızda farklı yapılara sahip petabyte veya daha büyük boyutlu veri setlerini analiz edilebilir. Büyük veriler yapılandırılmış, yapılandırılmamış veya yarı yapılandırılmış olabilir [23].

Hadoop, büyük miktarda veriyi ucuz ve verimli bir şekilde işlemek için kullanılan açık kaynaklı bir çerçevedir ve iş planlaması, büyük veri işlemede yüksek performans elde etmek için önemli bir faktördür. Bu makale büyük verilere genel bir bakış sağlar ve büyük verilerdeki sorunları ve zorlukları vurgular. Daha sonra Hadoop Dağıtılmış Dosya Sistemi (HDFS), Hadoop Map Reduce ve İş Takibi, Görev Takibi, Ad Düğümü, Veri Düğümü, vb. gibi büyük verilerdeki iş çizelgeleme algoritmalarının performansını etkileyen çeşitli parametreler. Bu yazının temel amacı, iş çizelgeleme algoritmaları ile birlikte karşılaştırmalı bir çalışma sunmaktır. Hadoop ortamında deneysel sonuçlar. Gerçek zamanlı akış verilerinden bilgi keşfi ihtiyacı günümüzde

sürekli artmaktadır ve madencilik modelleri için işlemlerin işlenmesi verimli veri yapılarına ve algoritmalara ihtiyaç duymaktadır. Apache Hadoop kullanarak zaman açısından verimli bir Hadoop CanTree-GTree algoritması öneriyoruz. Bu algoritma, kayan pencere tekniğini kullanarak, gerçek zamanlı işlemlerden tüm sık kullanılan öğe setlerini (kalıpları) madenler. Bunlar sık sık kapalı madde kümeleri için madencilik yapmakta ve daha sonra ilişkilendirme kuralları türetilmektedir. CanTree ve GTree olmak üzere iki veri yapısından yararlanır [24]. MapReduce çerçevesini kullanarak karakter tabanlı dize benzerlik birleşiminde veri çarpıklığını ele almak için yeni bir yaklaşım sunulmuştur.ve bu görevi yerine getirmek için MR-Pass Join adlı bir algoritma önerdik. Deneyleri yaptıktan sonra aday çifti oluşturma yönteminin ne yanlış pozitif ne de yanlış negatif sonuç üretmediğini gözlemlenmiştir. Önerilen algoritmanın, benzerlik veritabanı birleştirmesi için uygun sorgu işleme süresi ile ölçeklenebilirlik sağladığını tespit edilmiştir [25]. Hadoop kullanımı son yıllarda büyük oranda artmaktadır. Hadoop'un benimsenmesi yaygındır. Yahoo, Facebook, Netflix ve Amazon gibi bazı önemli büyük kullanıcılar, Hadoop'u yapılandırılmamış ve yapılandırılmamış verilerle çok iyi çalıştığı için esas olarak yapılandırılmamış veri analizi için kullanır. Hadoop dağıtılmış dosya sistemi (HDFS) büyük dosyaları depolamak içindir, ancak çok sayıda küçük dosyanın depolanması gerektiğinde, HDFS'deki tüm dosyalar tek bir sunucu tarafından yönetildiği için HDFS'nin birkaç sorunla karşılaşması gerekir. HDFS'de küçük dosyalar sorunu ile başa çıkmak için çeşitli yöntemler önerilmiştir [26]. GOM Hadoop, veri kümelerinin sipariş özelliğinden verimli bir şekilde yararlanmak için yeni bir grup-sipariş-birleştirme mekanizması benimsenmiştir. GOM Hadoop, önerilen mekanizmayı alternatif bir karıştırma mekanizması olarak dahil etmek üzere Hadoop'u genişleterek inşa edilmiştir. GOM-Hadoop'un performansı hem resmi modelleme hem de gerçek dünya veri kümeleri üzerinde kapsamlı deneyler yoluyla değerlendirildi [27]. Lin ve diğerleri hiper bağlantılı üretim iş birliği sistemleri arasında bilgi entegrasyonu ve birlikte çalışabilirlik için bir dizi Hadoop aracında yeni bir yaklaşım önermektedir. Hadoop yaklaşımında veriler “Extracted” web kaynaklarından, “Loaded” olarak “NoSQL” Hadoop'un veri kümesi içine (HBase) tabloları ve “Transformed” kovan şema on okuma ile arzu edilen biçim modeline entegredir. Sözdizimi ve anlam bilim web veri entegrasyonu için Hadoop Özü Yük Dönüştürme (ELT) yaklaşımının küresel akıllı

telefon değer zincirinde benimsenebileceğini gösteren bir vaka çalışması yapılmıştır [28].

MapReduce, bulut bilişim ve büyük ölçekli veri paralel uygulamalarında kullanılan etkili bir programlama modelidir. Hadoop, MapReduce modelinin açık kaynaklı bir uygulamasıdır ve genellikle veri madenciliği ve web dizini oluşturma gibi veri yoğun uygulamalar için kullanılır. Geçerli Hadoop uygulaması, bir kümedeki her düğümün aynı bilgi işlem kapasitesine sahip olduğunu ve görevlerin veri yerel olduğunu varsayar; bu da ek yükü artırabilir ve MapReduce performansını düşürebilir. Bu makale, dengesiz düğüm iş yükü sorununu çözmek için bir veri yerleştirme algoritması önermektedir. Önerilen yöntem, her bir düğümde depolanan verileri, heterojen bir Hadoop kümesindeki her bir düğümün hesaplama kapasitesine göre dinamik olarak uyarlayabilir ve dengeleyebilir. Önerilen yöntem, gelişmiş Hadoop performansı elde etmek için veri aktarım süresini azaltabilir. Deneysel sonuçlar, dinamik veri yerleştirme politikasının yürütme süresini azaltabildiğini ve heterojen bir kümede Hadoop performansını artırabildiğini göstermektedir [29]. Hadoop ve eneerji kullanımı ile ilgili olarak yapılan bazı çalışmalar ise [30, 31]'dır.

Yapılan çalışmalarda Pandu Sowkuntla veri azalttım ve sıralama ihtiyacını yönelik çalışması dikkate alınmıştır. Sowkuntla veri azaltmada kaba küme tabanlı yaklaşımla çalışmıştır. Özelliklerine göre verileri değerlendirerek indirgemeye gitmiştir [32]. Verileri indirgemede maliyeti baz alarak algoritma çalıştırılmıştır. Böylece veri işlemede maliyet kazancını en üste tutmayı amaçlayarak bu noktada gelişme sağlanmıştır. Önerilen algoritmada kullanılan dikey bölümleme şeması kullanılarak veriler özellik alanı üzerinden dağıtılmıştır. Bu yapıda, veriler karıştırılır ve sıralanır. Maliyet kazancı, bu noktada en üst düzeyde ele alan bir yaklaşım geliştirilmiştir. Verileri bayt uzunluklarına göre sıralanmıştır. Sıralamada böl ve yönet yaklaşımını kullanılmıştır. Geliştirilen yapının her noktasında ön bellek kullanımını ve her türlü maliyete dikkat ederek yeni azalttım yaklaşımlarında bulunması amaçlanmıştır. İşlemlerde ve sonuç çıktılarında indeksleme yapılarak maliyet kazancı arttırılmıştır. Geliştirmede sık kullanılan verileri ile az maliyetli verileri öncelikli hale getirilmiştir. Yaklaşım ileride eklenebilecek yazılım sınıflarını maliyetini de düşürmüş olmuştur.

Yao ve arkadaşları [33] Uzamsal veri bölümleme (SDP), uzamsal veriler için dağıtılmış depolama ve paralel hesaplamada yapısını önermişlerdir. Bununla birlikte, uzamsal verilerin çarpık dağılımı ve değişen uzamsal vektör nesneleri hacmi nedeniyle hem uzaysal işlemin optimum performansını hem de kümedeki veri dengesini sağlamak için oldukça güçtür. Bu sorunu çözmek için, Hadoop'ta büyük mekânsal verileri bölümlere ayırmak için mekânsal kodlama tabanlı bir yaklaşımı geliştirmişlerdir. Bu yaklaşım ilk olarak, uzamsal kod, boyut, sayım ve diğer bilgileri içeren bir algılama bilgi kümesi (SIS) oluşturmak için tüm büyük uzamsal verileri, uzamsal kodlama matrisine dayanarak sıkıştırmışlardır. Roy ve diğerleri [34] uydu görüntü işleme, büyük ölçekli görüntülerin tek bilgisayar analizine sınırlama getirmişlerdir. Hadoop Dağıtılmış Dosya Sistemi (HDFS) ise, bu dosyaları doğal veri paralellik tekniği ile ele almak için dikkate değer bir çözüm sunmuştur. Sharma ve Bagga [35] tarafından Hadoop resim işleme ara yüzü önerilmiştir.

Herodotou ve diğerleri [8] performans sağlama ve basitlik amaçlarını faklı şekilde hedef aldık. Yeni bir yaklaşım için çalıştık. Verilerin değerleri sahip oldukları bayt alanı ve veri yığınında oluşan tekrar adedi ile doğru orantılı kabul ettik. Burada kullanıcının en az maliyetli en önemli verilere ulaşması amaçladık. Yang ve arkadaşları haritalama ve indirgeme ve birleştirme eklemişlerdir. Bu noktada bizde sıralama yaklaşımı ekleyerek ek bir yapı koymayı başardık. Gopalani ve Arora [11] K en yakın komşu kullanmıştır performansta büyük artılar sağlamışlardır. Bizde verilerimizi minimize ederek en az maliyete sahip veri kümelerini en çok kullanılan veriler ile doldurduk. Bu noktada veri maliyetini verinin alanı ile ilişkilendiren ve sıralamada bu maliyeti göz önünde tutan bir yaklaşım olmuş olduk. Sıralama performans için bağlı listeler üzerinden gidilmiş. Aynı zamanda sonuçlar indekslenerek o noktada dahi maliyet kazancı göz önünde tutulmuştur.

.

BÖLÜM 3. VERİ ANALİZİ

Veri analizi ve verilerin etkin bir biçimde işlenmesi ancak veri sıralaması ve indirgemenin büyük veri analizinde etkin kullanımı ile mümkün olabilmektedir. Bulut bilişimde de veri analizi verilerin hacmi ve etkin bir biçimde işlenebilmesi açısından oldukça önemlidir. Bulut bilişim ve veri analizi aynı zamanda veri madenciliğinin temel alarak işlemleri gerçekleştirmektedir.

Veri madenciliğinde amaç anlamlı veya anlamsız verilerin arasında kalan, değerli anlamlı verilerin kullanıma hazır hale getirilmesidir. Verilerin öncelikle işlem maliyetleri çok net bir şekilde azaltmaktadır. Veri madenciliğinde gerçekleştirilen işlem basamakları ise:

1. Veri elde etme
2. Veri güvenliğini sağlama
3. Veri temizleme
4. Veri bütünleştirme
5. Veri indirgeme
6. Veri dönüştürme
7. Veri madenciliği algoritmaları uygulama
8. Yazılım dillerinde test aşamasına ve eğitim aşamasına sokma
9. Sonuçların değerlendirilmesi ve sunulması

Bulut bilişim ve büyük verilerin analizi son yıllarda artan teknoloji ile birlikte iş dünyası ve bilim dünyası olmak üzere uygulaması kaçınılmaz bir alan olarak karşımıza çıkmaktadır. İnternetin gelişmesi ile birlikte web 2.0, sosyal medya ile kullanıcılar arasında paylaşımın gelişmesi, çevrimiçi yayın yapan gazete ve yayın organlarında, dergi gibi kaynaklar ile birlikte bilgi miktarının artması ile birlikte yayınlanan bilgi

miktarı, daha önce görülmedik derecede büyük boyutlara çıkmıştır. Ancak veri miktarındaki bu artış verilerin anlamlandırılabilmesi, gerekli sonuç ve kuralların çıkartılması hususunda mevcut bilgilerin değerlendirilerek, kullanışlı veriler haline dönüştürülmesi gerekliliğini de beraberinde getirmektedir. Verilerin ait oldukları sınıfların belirlenmesi sürecinde geliştirilecek olan algoritma ile birlikte verilerin gruplandırılması ve ilerleyen zaman içerisinde uygun kuralların ortaya çıkartılabilmesi hedeflenmektedir. Makine öğrenmesi, veri madenciliği gibi algoritmaların kullanılması ile aynı zamanda büyük veri analiz işlemi de gerçekleştirilebilmektedir. Bu algoritmalar ancak büyük veri analizi için tasarlanmamışlardır. Dolayısıyla geliştirilecek algoritma yapısı ile birlikte verilerin dinamik yapıda sınıflandırılabilmesi, kümelendirilmesi gerekecektir.

Haritalama ve indirgeme ara bileşeni ile birlikte geliştirilecek program ve algoritma ile aynı zamanda verilerin analiz işlemleri de gerçekleştirilebilecektir. Bir algoritmanın işlerliği, veri miktarını analiz edebilme kapasitesine bağlı olarak değişebilmektedir.

Büyük verilerin analizleri için aynı zamanda büyük kapasiteli bilgisayarlar ve işlemcilere de ihtiyaç vardır. Bulut bilişim mimarisi ile birlikte, veri haritalama ve azatlımı teknikleri ayrıca veri madenciliği algoritmaları ile birlikte bağlantılı olarak çalıştırılacaktır. Bu çalışmada Hadoop haritalama ve indirgeme uygulaması ele alınmış ve bu uygulamanın işlem ve algoritma yapısı ele alınarak incelenmiştir. Gerçekleştirilen uygulama ile bu yapının çalışma biçimi gösterilmiştir.

3.1. Büyük Veri Yapısı ve Özellikleri

Büyük veri, sürekli artan boyutsal miktarda, hacimde ve çeşitlilikle birlikte her geçen gün artan bir yapıya sahiptir. Bu karmaşık ve sürekli büyüyen veri kümelerini çözümlemek, işlemek ve anlamlı sonuçlar çıkartmak günümüzün en büyük problemlerinin başında gelmektedir. Bu tarz problemleri çözmek ve büyük verilerin analizi sürecinde geliştirilen yazılımlardan birisi olan Hadoop haritalama ve indirgemedir. Bu çalışmada Hadoop haritalama ve indirgeme ve performans analizi

uygulamalı olarak incelenmiştir. Bir veri kümesinin büyük veri olarak sınıflandırılması için, Demchenko ve arkadaşları tarafından belirlenen 5 V kavramı; çeşitlilik, hız, hacim, doğrulama ve değer özelliklerini sağlaması gerekmektedir [1]. Bu özellikler ise;

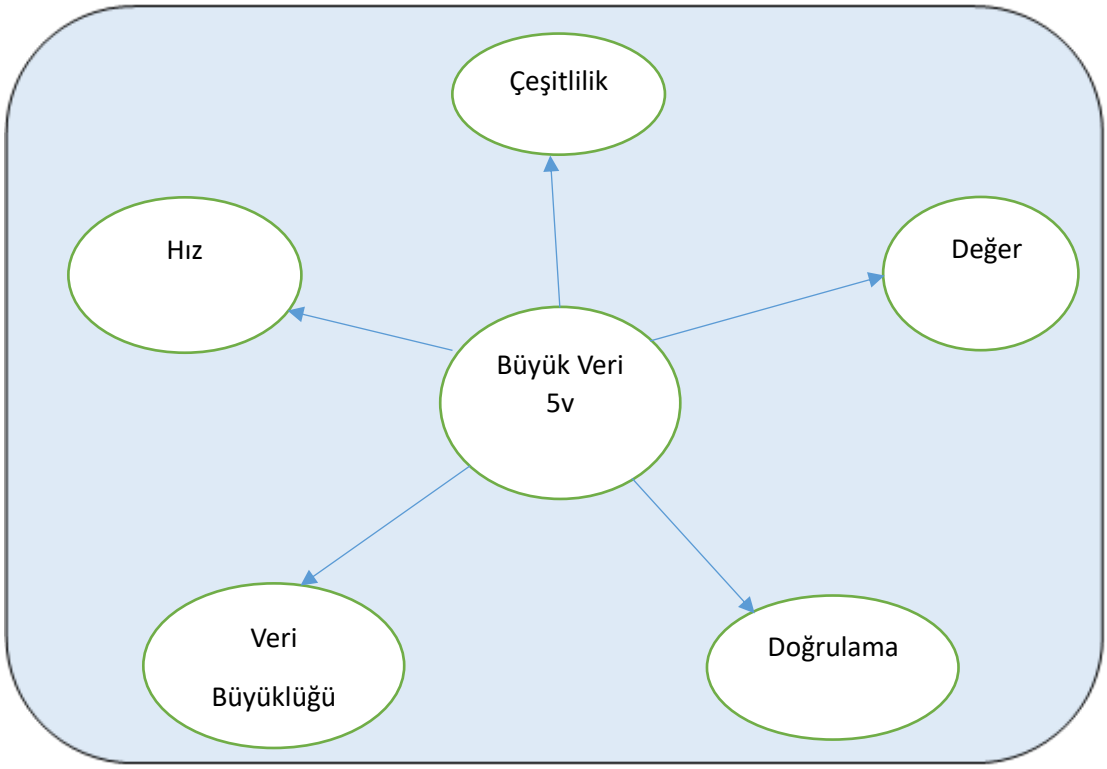
- Çeşitlilik: Verilerin yüzde 80'i yapısal değildir ve teknolojiler, farklı formatlarda veri üretebilmektedirler. Birçok teknoloji bunların yazılım sonuçlarını bir noktada birleşmek zorunda kalabilmektedir. Aynı zamanda veriler birleşebilir ve formatları uyarlanabilir olmalıdır.

- Hız: Veri hızı, her geçen gün artmakla birlikte, verilerin işlenme hızı ve kullanım hızında ihtiyaca göre artış göstermekte ve beraberinde teknolojik ilerlemeyi de zorunlu hale getirmektedir. Bu sebepten dolayı verilerin doğru haritalandırması süreci öncesi verilerin sistem içerisine dahil edilmesi önem kazanmaktadır.

- Veri Büyüklüğü: IDC istatistiklerine göre 2020'de yeni ulaşılacak veri boyutu, 2009'un 44 katı kadar olacaktır. 2010'lu yıllarda ise dünyadaki toplam bilişim harcamaları yılda %5 artmakta, fakat üretilen veri miktarı %40 olarak artmaktadır. Durum bu şekilde devam ederken verilerin nasıl yapılandırılacağı ve çözümleneceği problemini şimdiden planlamalıyız.

- Doğrulama: Bilgi yoğunluğu içerisinde veri akışı, verinin depolanması ve işlenmesi sürecinde doğrulama sürecinin de gerçekleştirilmesi gerekmektedir. Güvenli bir ortamda verinin analiz edilmesi oldukça önemlidir. Akış sırasında, doğru noktadan kullanıcılar tarafından talep edilen güvenlik kurallarına bağlı olarak alınması gerekmektedir. Bu süreç sonrasında aynı zamanda verinin doğruluğundan emin olunması gerekmektedir.

- Değer: Değer yaratması, yani verilerden belirli bir sonuç ve kural yapısının elde edilebilmesi sürecidir. Eğer veriler değer içermez ise bunları yorumlamak, işlemek veya karşılaştırmak kuşkusuz imkânsız olacaktır.



Şekil 3.1. Büyük veri kavramına ait temel bileşenler.

3.2. Büyük Veri ve Büyük Verinin Analizi

Büyük veri analizini, herhangi bir şey veya bir durum hakkında ne kadar çok şey bildiğinizle hakkında, daha güvenilir bir şekilde yeni yaklaşımlar geliştirebileceğimiz ve gelecekte neler olacağı konusunda tahminlerde bulunabileceğinizi ilkelere dayanır. Çağımız, sürekli artan oranda bir veri yoğunluğu ile karşı olduğumuzu göstermektedir. Sadece bir günlük kısıtlarda daha önceki günden daha fazla veriye sahip oluğumuz kuşku götürmez bir gerçektir. Oluşan bu veri miktarı sadece artmakla kalmayacak kullanılan teknolojileri ileriye taşıyacaktır. Daha hızlı veri analiz eden ve işleyen yapılar oluşturmamız bu konuda gereklidir. Ayrıca veri yoğunluğunda tüm verinin işlenmesi gerekliliği sorusunu kendimize sormalıyız. Örneğin analiz edilecek değerler belli özellikleri içeriyor ise haritalandırma aşamasında diğer verilerin dışarıda bırakılmasını sağlamak zorunda olduğumuz gerçeğinden hareket etmeliyiz. Ayrıca bir büyük veri havuzunun tamamını analiz etmek zorunda mıyız? Bu soruda şartlara bağlı olarak değişecektir. Verileri sınıflandırmamız ve sınırlandırmamız analiz konusunda ihtiyaç duyacağımız kaynak miktarını azaltacaktır. Ayrıca bu alınan veriler ile

karşılaştırma yapılması için oldukça fazla miktarda ve az boyutta olmaları very analizde kaynakların doğru kullanılmasını sağlayacaktır.

Verilerin analiz öncesi ve analiz esnasında dikkate alınması gereken işlemler;

1. Verileri sınıflandırmalı ve sınıf dışında kalan verileri işleme almamalı,
2. Verilerin içinde örneklem alınacağı zaman, çok miktarda olan ve az yoğunluktaki verilere öncelik verilmeli,
3. Veri işlemeye başlamadan önce bu işlemler yapılmalı ve haritalandırılmış veriler ile çalışmalar yapılmalıdır.

3.3. Hadoop Hdfs (Hadoop Distributed File System)

HadoopCommon: Büyük veri işlemesinde kullanılan tüm Hadoop modüllerini kullanımında olan ortak modülleri kapsamaktadır (https://hadoop.apache.org) [38].

Hadoop Distributed File System (HDFS): Büyük oranda olan verilere yüksek iş/zaman oranı işlem sağlayan dağıtık bir dosya yönetim sistemidir. Burada amaç bir çok düğüm üzerinde olan dosya sistemlerini birbiriyle bağlayarak tek bir dosya sistemi gibi düğümlerin çalışmasını sağlamaktadır. HDFS, düğüm noktalarının her zaman çalışmayacağını ve belli zamanlı kesintiler olabileceğini göze ardı etmez. Bu yüzden veri güvenliğini, verinin birçok ta paylaşılarak tutması sağlanmalıdır.

HadoopYarn: İş zaman ve kaynak paylaşımı yapan bir dizi kütüphaneden oluşur.

HadoopMapReduce: Yarntemelli, çok miktarda veriyi paralel olarak işlemeye yarayan bir sistemdir. İş yükünü belli parçalara ayırarak işlemektedir, burada haritalama ve işleme isimden anlaşılacağı gibi iki ayrı parçadır.

Cassandra: Ölçeklenebilir ve çok parçadan oluşan iki parçalı, kümelenmiş veri tabanıdır.

Chukwa: Büyük ve parçalı sistemleri yönetmek veriveri toplama yapabilen sistemidir.

Hive: Veri özetleme ve “ad hoc” sorgu yazma ortamı olan bir veri ambarı yapısıdır.

Pig: Paralel işlemlerin tasarlanması için hazırlanmış, veri akışlarını ve çalıştırmasının planlanması için oluşturulmuş bir dildir.

Veriler sürekli arttığı bir ortamda verilerin iş bölümü ile işlenmesi sağlanmalıdır. Tek bir makine yerine birden fazla makinede işleme ile oluşan çözümler ortaya atılmıştır. Hadoop yazılımı bulut bilişim olanakları ile birlikte, istenilen düzeyde verilerin direk olarak haritalaması, azatlımı konularını da ele almaktadır. Ancak veri azaltıcı sınıflandırma kümeleri standart bir işlem uygulamaktadır. Önermiş olduğumuz çalışma ile veri madenciliği algoritması sisteme ilave edilerek sınıflandırma işleminin gerçekleştirilmesini sağlayan algoritma oluşturulmuştur. Hadoop yazılımında, veriler birden fazla makinede saklanmakta ve HDFS kütüphanesi kullanılmaktadır. Haritalandırılan verilerin paralel işlemlere tabi tutulması ile gerekli analiz işlemleri gerçekleştirilmektedir(https://hadoop.apache.org/)[39]. HDFS tarafından gelişilen haritalama ve indirgeme yöntemi kullanılmış ve iki parçalı işlem sağlanmıştır.

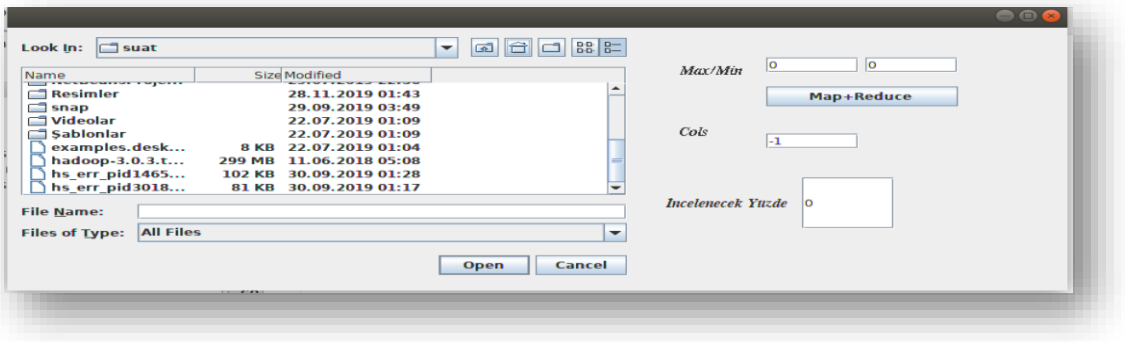
1. Devasa oranda büyük verileri saklayabilir ve işleyebilir.
2. Kaynak kullanımında çok karlı bir minimize kullanılmaktadır. Yatırım maliyetini düşürmektedir.
3. HDFS içerisine veriler aldıktan sonra birçok kaynak üzerinden aynı anda çalışmayapabilir.
4. Verilere hızlı bir erişim sunar.
5. Veriler parçalandığı için kolay ve ufak boyutlarda saklanabilir.

3.4. Veri Sınırlarının Tespiti

Veri analizi günümüzde sınırsız verilerin içinde boğulmamızı engelleyerek gerekli analizlerin yapılması ve istenilen sonuç değerlerinin elde edilmesini sağlayan bir

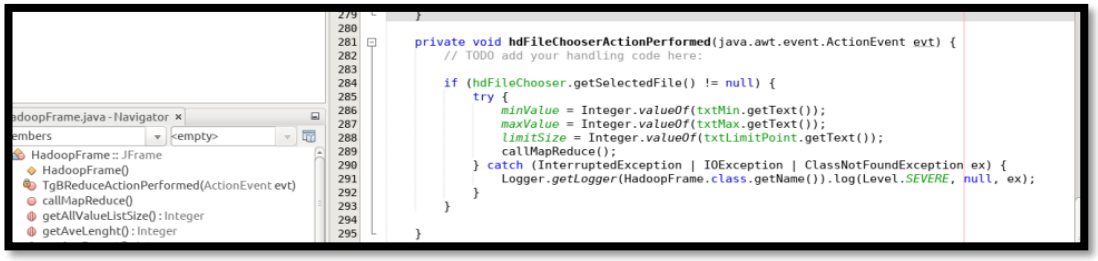
yaklaşımdır. Aranan verilerin belli sınırları tespit edilebiliyor ise, bu alt ve üst sınırlara uymayan verilerin elenmesi ve haritalandırmadan önce sistem dışına itilmesi ve haritalama aşamasında, hatta özellikle indirgeme aşamasında bize çok sağlıklı bir veri yığını oluşturabilmektedir.

Burada kullanıcı verilerin belli bir yüzdesini incelemek isteyebilir. Sorun en az ve en öncelikli olan verilerin analizi için kullanıcıya verilmesi gerekecektir. Burada en az bayt sahip ve en çok tekrara sahip veriler en önce gelecektir. Ayrıca kullanıcı verilerin bayt olarak aralığını bilebilir ve bu durumda gene kendi tercihi olan bayt aralığında olan verileri öncelikli olarak getiren bir haritalama yapısı bizi uygulamada daha tutarlı ve daha az zaman kaybına dönüşecektir. Şekil 3.2 de ise program içinde seçim, yüzde değer isteme ekran çıktısı verilmiştir.



Şekil 3.2. Program içinde bulunan seçim, yüzde ve sınır değer isteme ekranı.

Kullanıcı kolay kullanımı için grafiksel bir ara yüz tasarlanmıştır. Kullanıcı inceleceği dosya seçebilir. Dosya seçiminden önce maksimin ve minimim değerleri yazması gerekmektedir. Değerler değiştirilmez yapıda, sıfır olarak kalırsa listedeki tüm elemanlar kabul edilir. Eğer değerler değişmiş ise her eleman bayt olarak bu sayıların arasında olma şartı ile kabul edilir. Şekil 3.3'de ise sınır değişkenlere yer verilmiştir.



Şekil 3.3. Sınır değişkenleri.

Tarafımızdan geliştirilen yapıda verilerin sınırlanması bayt oranlarına bağlanmıştır. Bu bilgileri Şekil 3.3. gösterildiği gibi kullanıcıdan alınmıştır. Sınırlama için incelenecek en yüksek ve en alçak veri sınırı ile birlikte örneklem seçilmiş ve bu örneklemin toplam verinin yüzde kaçı olduğuna bakılmıştır. Örneklem seçilirken en değerli ve az maliyetli veriler, örneklemin içine yerleştirecek bir yaklaşım bu tez çalışması ile birlikte geliştirilmiştir. Bundan dolayı sınırlama esnasında en yüksek bayt oranına sahip verileri dışarıda tutulmuştur. Bu işlem ile belli bir sıralamaya şu an için sahip olmayan fakat en değerli verilerden oluşan bir örneklem elde edilmiş olacaktır. Çalışma kod parçasında öncelikle dosya seçimi olması şartı dikkate alınarak kontrol edilmiştir. Daha sonra, ekranda kullanıcı tarafından girilmiş olan ve yazılımda istenmeyen verilerin engellenmesi ile kullanılacak verilerin maksimum ve minimum uzunluğu alınarak işleme devam edilir. Son olarak ise veri yığınından oluşturulması istenen örneklem yüzdesi ele alınmaktadır. Bu yüzde bazında sıfır ile yüz arasından alınan değer oranında örneklem alınacaktır. Bu yüzdesel ifade bize seçeceğimiz örneklem boyutu hakkında bilgi verecektir. Program kod parçası ile hata oluşması durumuna karşı bir tedbir alınarak, blok içine alınarak hataya karşı korunmuştur.

3.5. Örneklem ve Önemi

Tüm veri kitlesi ile çalışmak büyük veri analizinde, büyük verilerden sonuç elde etme sürecinde bazen oldukça zor, zaman alıcı ve maliyetli olabilir. Bu durumlarda, ana kütleyi en iyi yansıtacak bir örneklemin seçilmesi oldukça önemlidir.

Örneklem seçiminde, seçilen örneklemin hacmi ne kadar sağlıklı olursa, elde edilecek sonuçlar yani ana kütleyi temsil etme yeteneği de o denli kuvvetli olur. Dolayısıyla

ana kütlenin tamamının analiz edilmesine gerek kalmadan, örneklem analizi ile ana kütle hakkında tutarlı bir biçimde tahmin edebilme yeteneğine sahip olmuş oluruz.

Amaç tüm veriyi analiz etmek yerine tüm veriyi en iyi temsil eden örneklemi seçerek, maliyet ve zaman açısından tasarruf sağlamaktır Şekil 3.4 ve Şekil 3.5 örnek filtre kodunu göstermektedir.

```
wordList.stream().filter(item->item.length()<20==true&&
item.length()>0==true).map(item->item).collect(Collectors.toList());
```

Şekil 3.4. Örnek Fitre Kodu 1.

Bu kodlardan da görüldüğü gibi, sınır değerlerin uygulanmasından sonra, uygun örneklemin seçimi esnasında, veri örneklemlerinin seçim işleminde yardımcı olmakla birlikte aynı zamanda maliyet açısından da avantaj sağlamaktadır.

```
wordList.stream().map(item->item).filter(item->item.length()<20==true
&&item.length()>0==true).collect(Collectors.toList());
```

Şekil 3.5. Örnek Fitre Kodu 2.

Kod yapısında veri işlemede uygun işlemler sıra ile yapılmalıdır. Tabi bu durumlara göre değişebilir. Öncelikli işlem Şekil 3.4.'de işlemde öncelikle analiz yapılmıştır ve ardından ise haritalama işlemi sonucu aktarılmıştır. Ardından gelen işlem Şekil 3.5.'de işlemler tam tersi bir sırada yapılmıştır. Öncelikle haritalama işlemi gerçekleştirilmiş ve daha sonra analiz işlemi gerçekleştirilmiştir. İkici kod parçası da maliyet açısında büyük farklar oluşturarak avantajlar sağlamıştır.

Geliştirilen bilgisayar program kodu ile Java'da yazılıp geliştirilen örnek kod yazılımları lambda programı ile yazılmıştır. İki kod parçasında da çalıştırılma sonucu elde edilen çıktı aynı sonuç olacaktır. İlk işlemde, önce filtreme yapılmış ardından haritalama işlemi gerçekleştirilmiştir. İkinci işleminde haritalama sonrası filtreme uygulanmıştır. Çalışma süreleri farklıdır. Sonuç aynı olsa da buradan açıkça anlaşılacağı gibi eğer liste haritalama aşamasında uygun bir filtreleme ile gerekli örneklemlerin arasında oluşturulur ise indirgeme aşamasında işlenecek veri de o denli

azalacaktır. Bu noktada geliştirilen yapıda, bu sebepten dolayı, öncelikle maliyeti yüksek verileri sistem dışına çıkartacağız ve ardından maliyeti ile değeri fazla olan veriler ön alana çekilmiştir. Bununla birlikte maliyetleri daha dip bir noktaya çekmek için verileri daha az saklamaya ve maliyetine uygun bir indeksleme yapılması gerekmektedir. Filtreleme ve sınırlama veri analizinde bu işlem gerekmektedir. Böylece en fazla verim ve az maliyet ile gerekli olan çalışma ortamı sağlanmıştır.

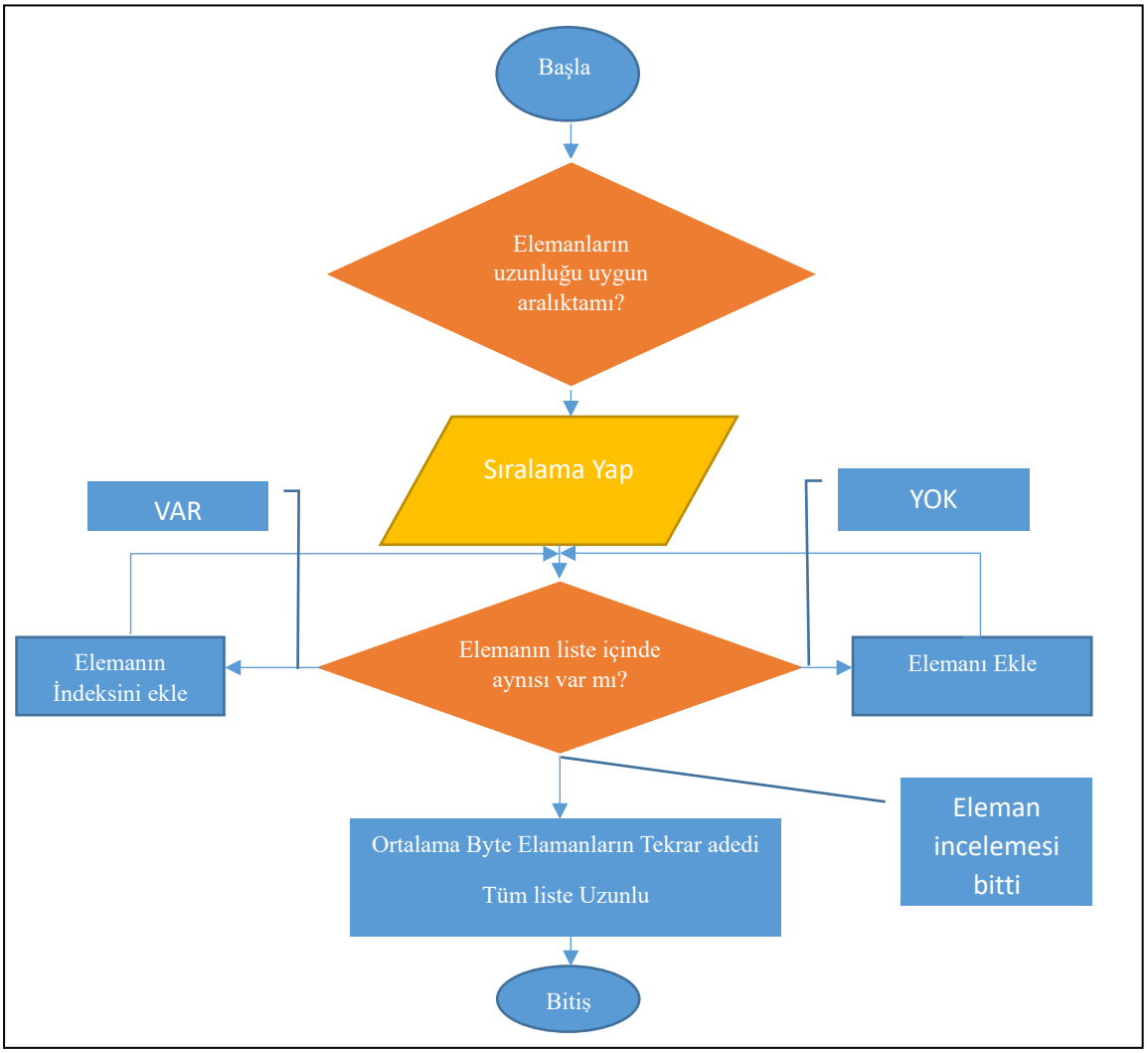
Verilerin önünde ve arkasında olan boşluklar Şekil 3.6.'da görüldüğü gibi temizlenmiş ve her verinin bayt uzunluğu dikkate alınarak işlenmiştir. Bu program parçasında verilerin bayt uzunluğu dikkate alınmıştır. Kullanıcıdan alınan en büyük ve en küçük bayt sınırları ile karşılaştırarak kontrol edilmiştir. Bu verilerin bayt uzunlukları kullanıcı tarafından istenen aralıkta olan veri uzunluğuna içerisinde olmadığında dışlanır yani kullanmak üzere bu veriler tercih edilmez. Eğer giriş yapılan için veri kabul edilen aralıkta ise program içinde olan global bir harita içine eklenir. Eğer burada listede ayni verilerden tekrar eden başka verilerde var ise bu veriler herhangi bir karışıklığa sebep vermemesi için indeks numarası ile kayıt altında tutulur. İndeks adreslemesi ile verinin işlenmesi ve sonuç çıktılarında en az maliyet sağlanması amaçlanmıştır. Ayrıca bu işlemler haritalama işlemi ile birlikte yapılacak sıralamada işlem maliyetini de azaltmıştır.

Şekil 3.6. İstenilen verilerin örneklem içinde süzülmesi.

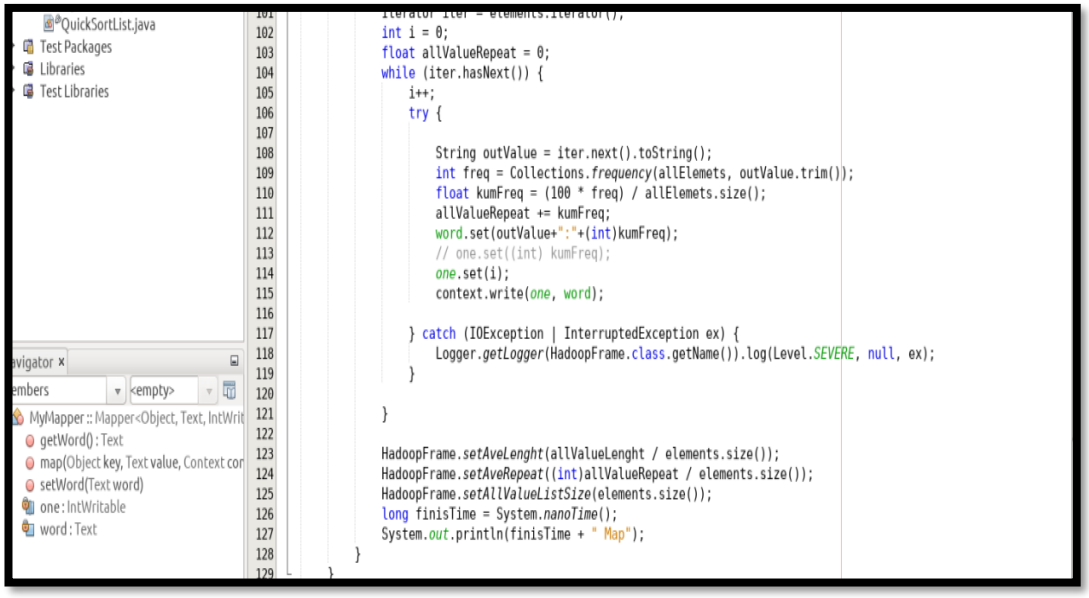
Aynı veriler, sıralamaya dahil olmayacak böylece işlem daha az yapılacak fakat aynı değerlerin yerleri tutulan indeks adreslerinden kolaylıkla bulunabilecektir. Bir veri dizisinde istenilen değerlerin tekrar sayısının değerin önemi ile orantılıdır. Bu sebepten elemanların işlem frekansı bize örneklem seçiminde kolaylık sağlamıştır. İstenilen şey en önemli, en az maliyetli ve en çok veriyi analiz etmektir. Bu sebepten dolayı

ortalama tekrar adedi ve ortalama bayt uzunluğu baz alınarak sıralama sonuçlanmalıdır. Ortalama bulma işleminde aritmetik ortalama uygun olmuştur.

Bu noktada verilerin analizinde ilgili birçok çalışmada verilerin değerleri tespit edilmeye çalışılmıştır. Şekil 3.7'de hızlı sıralama algoritmasına yer verilmiştir. Veri sıralamaya tabi tutulduktan sonra verinin tekrar adedi ile tüm yığın içinde olan değeri tespit edilmiştir. Bu işlem Şekil 3.8.'de gösterildiği gibi liste içinde olan elemanların hepsini incelenmiş ve ortalama bayt uzunluğu alınmıştır. Her eleman için bir frekans değeri ile puanlama yapılmıştır. Bu puanlama elemanın tekrar sayısının yüzdelik dilimde karşılığıdır. Ardında bu bulunan sonuçlar, kendi geliştirdiğimiz sınıflarımızın yapısında çıktıyazdırılır. Bulunan değerler ile birlikte tüm elemanların uzunluğu sonuç verisi olarak tutulmuş ve son olarak derleyici çıktısına sistem zamanı yazdırılmıştır.



Şekil 3.7. Hızlı sırama algoritması.

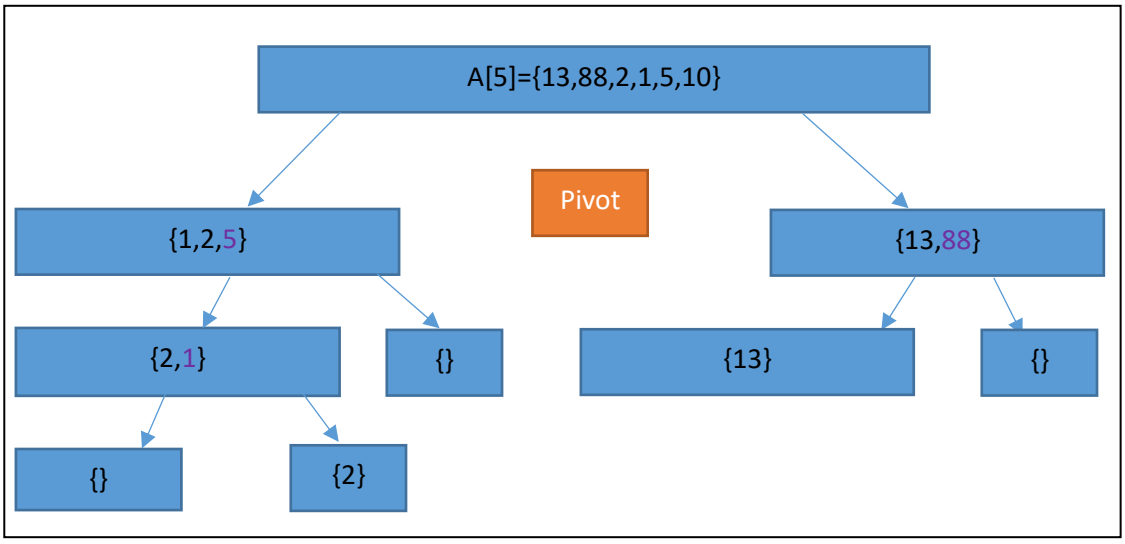


Şekil 3.8. Verilerin ortalama değerlerinin hesaplanması.

Bu çalışmada, bu verilerin analizinde verinin işlem maliyeti ile birlikte değerinin hesaplanması amaçlanmıştır. Bu işlemler esnasında maliyet ve değerin birbirleri ile ters orantılı olduğunu ve birbirleri ile bağlantılı olduğunu gösterilmiştir. Verilerin daha hızlı bir algoritma ile işlenerek sonuçlanması aynı zamanda daha az maliyet sağlayacağı da gösterilmiştir. Geliştirdiğimiz yapıda indeksleme yaklaşımını bu noktada bize katkı sağlamıştır. Zaten indeksleme yapısı ile işlemlere devam etmemiz işlem yapacak diğer sınıfların yük maliyetini ve sonuç çıktılarının oransal boyutunu düşürecektir.

BÖLÜM 4. VERİLERİN HIZLI SIRALAMASI

Verilerin hızlı sıralanması özellikle büyük veri analizinde verilerin işlenmesi ve verilerden anlamlı sonuçlar çıkartılması sürecinde oldukça önemlidir. Böl ve Fethet yapısı ile birlikte algoritma veri kümesini aşamalı olarak bölmekte ve her böldüğü parça üzerine algoritma uygulanarak anlamlı sonuçlar çıkartılmaya çalışılır. Kısaca Böl ve Fethet sıralama sorunun çözmeye yönelik yapılan bir algoritma yaklaşımıdır. Ortaya çıkan problemi benzer problem yapılarına göre parçalıyoruz. Daha sonra bu parçaları kendi içerisinde ayrı ayrı anı işlemlerle çözümledikten sonra bu parçaları birleştirerek gerçek çözüme ulaşıyoruz. Klasik yapıda veriler parçalanır karşılaştırılır ve sıralanır. Bu karşılaştırma örneği Şekil 4.1.'de verilmiştir.



Şekil 4.1. Hızlı sıralama çalışma görseli.

Hızlı sıralama (böl ve yönet) sürecinde veriler seçilen bir anahtar eleman ile işlem gerçekleştirilmekte ve sıralama işlemi bölümlendirme ile özyinelemeli olarak oluşturulmaktadır. Burada değişken durumu anahtar seçimi ile yakından ilişkilidir.

Pivot durumu tamamen verinin yapısına bağlı olarak başta veya sonda bir değer alabilmektedir. Seçilen pivot değerine bağlı olarak veriler parçalanır ve her parça birbiri ile karşılaştırılarak sıralanır. Bu bölme işleminde dolayı hızlı sıralamanın bulundu alana böl ve yönet algoritmaları denir. Hızlı sıralamalı işlemlerde tamamen verinin dinamik yapısı dikkate alınarak veri sıralaması işlemi gerçekleştirilir. Hızlı sıralama, daha kolay uygulanabilmesi için çeşitli veri tipleriyle çalıştırılabilmesi ayrıca da genel olarak diğer algoritmalardan hızlı çalışması sebebiyle oldukça sık kullanılan sıralama algoritmasıdır. Ortalama olarak O(NlogN) yapısı ele alınmıştır. Birleştirme algoritması da hız olarak iyi olmasına rağmen alansal tasarrufta hızlı sıralama kadar başarılı değildir. Bu sebepten sıralamada en uygun olan yapı hızlı sıralama algoritması olacaktır. Tablom 4.1'de hızlı sıralamanın hızlı, birleştirme, eklemeli ve seçmeli tipleri ile birlikte karşılaştırma sonucu verilmiştir. Tablo 4.2'de ise sıralama algoritmalarının özellikleri verilmiştir.

Tablo 4.1. Hızlı sıralamamın diğer sıralamalar ile karşılaştırılması.

	En İyi Zaman	Ortalama Zaman	EnKötü Zaman	Alan
Hızlı	O(nlogn)	O(nlogn)	O(n^2)	nlogn
Birleştirme	O(nlogn)	O(nlogn)	O(nlogn)	1
Eklemeli	O(n)	O(n^2)	O(n^2)	1
Seçmeli	O(n^2)	O(n^2)	O(n^2)	1

Tablo 4.2. Sıralama algoritmalarının özellikleri.

İsimlendirme	Özellikler
Sabit	O(1) tekrarsız arama
Loglog	O(loglogN) sezgisel arama
Logaritmik	O(logN) İyi azaltılmış bir argoritma bulur ve problemleri parçalıdır
Doğrusal	N logN o(NlogN) Hızlı bir algoritmadır. N tane veriyi küçük verilere böler.
Karesel	O(N2) Veri miktarı az olduğu zamanlarda kullanılmalıdır
Kübik	O(N2) Veri miktarı az olduğu zamanlarda kullanılmalıdır
Üssel	O(N2) Veri miktarı çok olduğu zamanlarda kullanılmalıdır

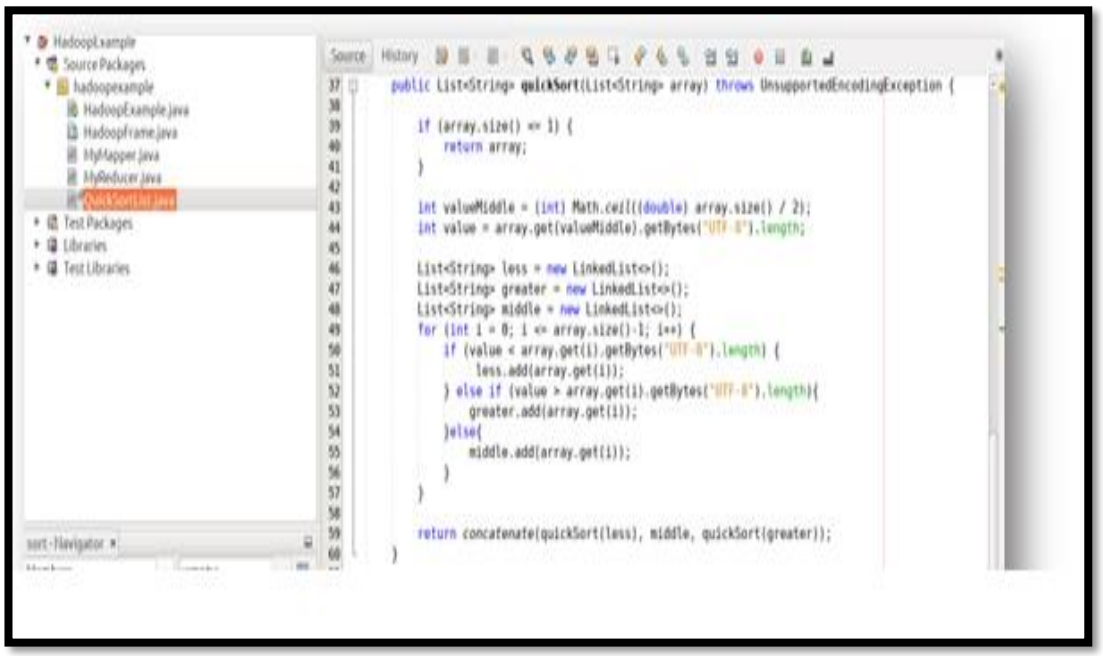
Hızlı sıralamada bayt uzunluğu önemlidir ve bu yapıda pivot ile aynı olan değerlerin sistemden çıkarılarak tekrar sıralamaya gönderilmemesi zaman ve maliyet kazancına yol açacaktır. Bu sebepten sıralamada üç adet veri tutucu geçici diziler oluşturulmuş ve bu dizilerden eşit olanlar sıralamaya gönderilmezken diğer iki dizi sıralanmak için tekrar işleme alınmıştır. İşlemler tekrarlanır ve üç adet yeni dizi oluşurulur, bu döngü sıralama bitene kadar devam etmiştir.

Bu dizi parçalama işlemleri ve sıralama işlemleri tamamlanmasının ardından sırada bu yapıları birleştirmek yer almaktadır. Veride bayt olarak daha az olan değerler başa yazılarak sıralamada daha az maliyete sebep olacak verilerin ön tarafa alınması, incelemeleri kolaylaştıracağı gibi ön görülen zamanda daha çok veriyi değerlendirmemizi sağlayacaktır.

Bu algoritmada nlogn veri alanı kullanılmaktadır. Burada veri kullanımını, minimum süre değerlerine indirgeme amaçlanmaktadır. Bu nokta çözüm yöntemi ile adres gösterme olacaktır. Kısaca aktarmak gerekir ise bir değeri listeye eklemek yerine ilgili değerin hafıza üzerinde tutulduğu veri yolu adresinin tutulmasıdır. Bu sebepten Java ile geliştirilen bu yapı içinde dizi listelerinden bağlı listeler kullanılmıştır. Liste içinde her bir öğesi, listede her iki komşu öğeye işaret eden iki işaretçi (adres) içerir. Bu nedenle kaldırma işlemi, yalnızca düğümün iki komşu düğümünde (öğe) işaretçi konumunda değişiklik yapılmasını gerektirir. Kısaca verileri parçalar iken sadece düğümlerin içerdiği adresler bağlanıp koparılır. Bu adresleme ve düğümleme ile çalışan yapı bize bellek tüketimi yüksek yapsa da sürekli olarak değişen sıralamada çok yüksek hızlı çözüm avantajı sağlamıştır.

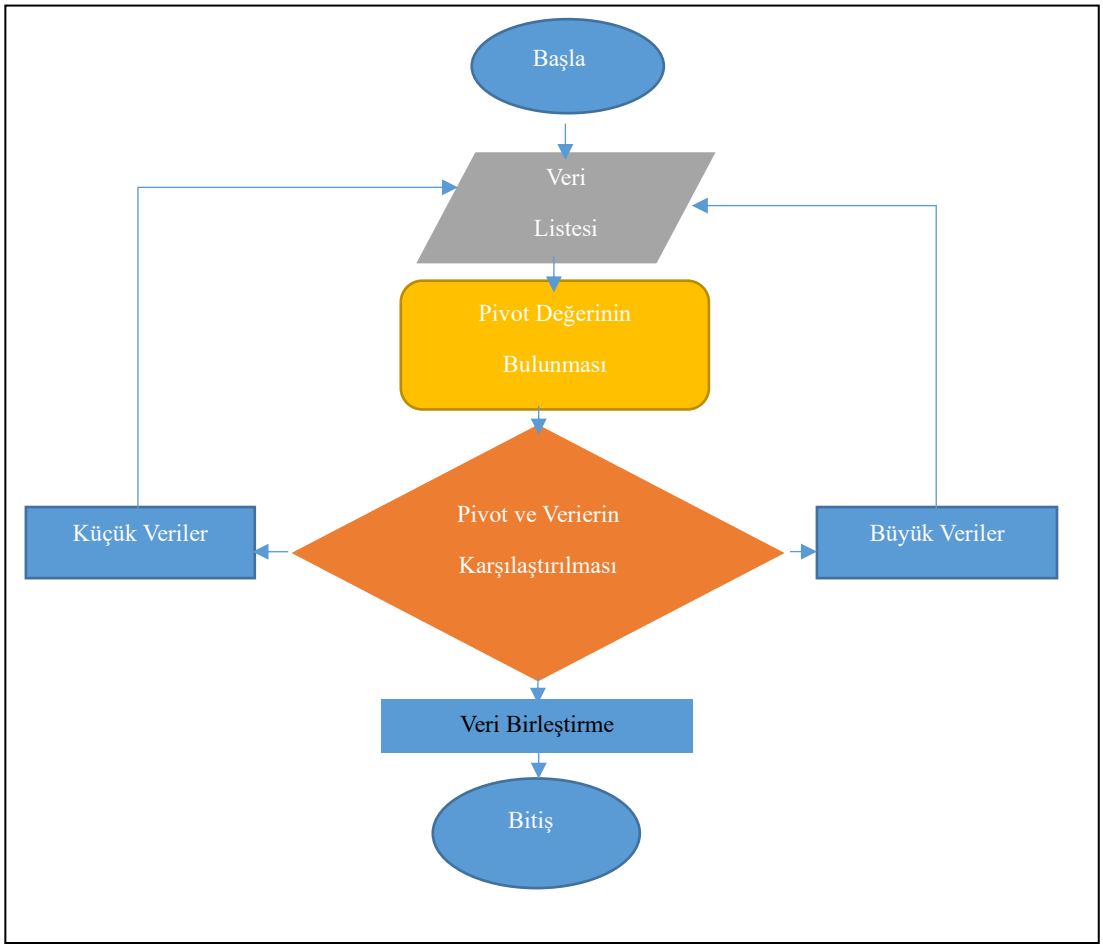
Geliştirdiğimiz bu yaklaşımda veri işleme zaman kazanımına ihtiyaç duyulmaktadır. Bu sebepten dolayı, sıralama esnasında işlem yükünü azaltmamız gerekmektedir. Bu sebep ile işlemde bölme işlemini üç parçalı gerçekleştirilmiştir. Bu parçalardan ikisi tekrar işlem içine girer iken bir parçada sürekli dışarı çıkmaktadır. Bu parçalar verilerin bayt olarak karşılaştırılası ile oluşur. Kısaca pivot değerinden büyük olanlar ile küçük olanlar iki ayrı liste olarak tekrar metot içine alınır. Eşit bayt olarak eşit olan

veriler ise tekrar metot içine alınmayan üçüncü bir listede tutulur. Şekil 4.2'de ise hızlı sıralama kod yazılımı yer almaktadır.



Şekil 4.2.Hızlı sıralama kod parçacığı.

Hızlı sıralamada bağlı listeler yaklaşımı geliştirilmiştir. Bunun sebebi, bize gereken veri çözümleme hızıdır. Eğer bağlı listeler kullanamaz isek sistem daha az bellek kullanmasına karşın daha çok program çalışacaktır. Sebebi ise, bağlı listelerde ön ve arka düğüm bilgilerinin işlem düğümünde saklanmasıdır. Şekil 4.2. Kod yazılımında, öz yenilemeli bir metot uygulanmıştır. Öncelikle liste içinde olan her elemanın adet toplamlarını birden büyük olması şartı kontrol edilmektedir. Ardından veri listesinde olan elemanların ortanca elemanı seçilmekte ve bu elemanın bayt karşılığı pivot değer olarak tutulmaktadır. Kullanılması amacı ile üç ayrı bağlı liste oluşturulmuştur. Listenin her bir elemanı sırayla pivot değer ile karşılaştırılmıştır. Bu karşılaştırılan her bir değer sonucuna göre ilgili listeye eklenmiştir. Bundan sonra büyük ve küçük bayt uzunluğuna elemanların oluştuğu listeler ele alınır ve tekrardan sıralanması için işleme gönderilmiştir. Her parça sıralaması tamamlısı ile birlikte birleştirme için yeni bir metot içine listeler gönderilmiştir. Şekil 4.3'de hızlı sıralama algoritması yer almaktadır. Şekil 4.4'de ise birleştirme metoduna ait kod yazılımı yer almaktadır.

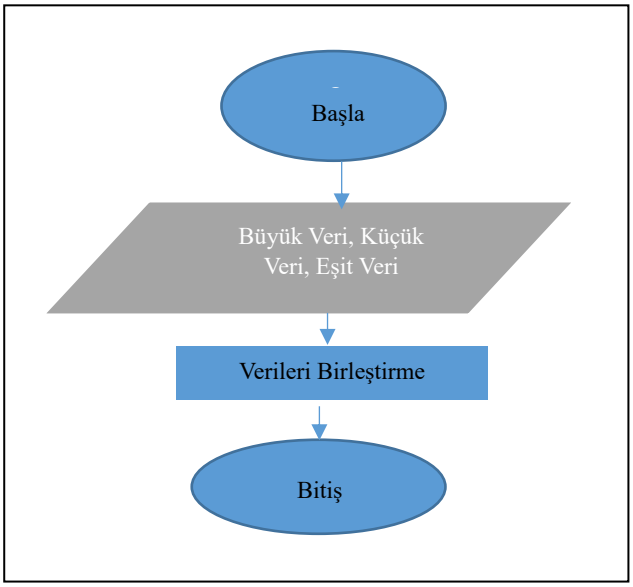


Şekil 4.3. Hızlı sıralama algoritması.

Şekil 4.4. Birleştirme metodu kod parçacığı.

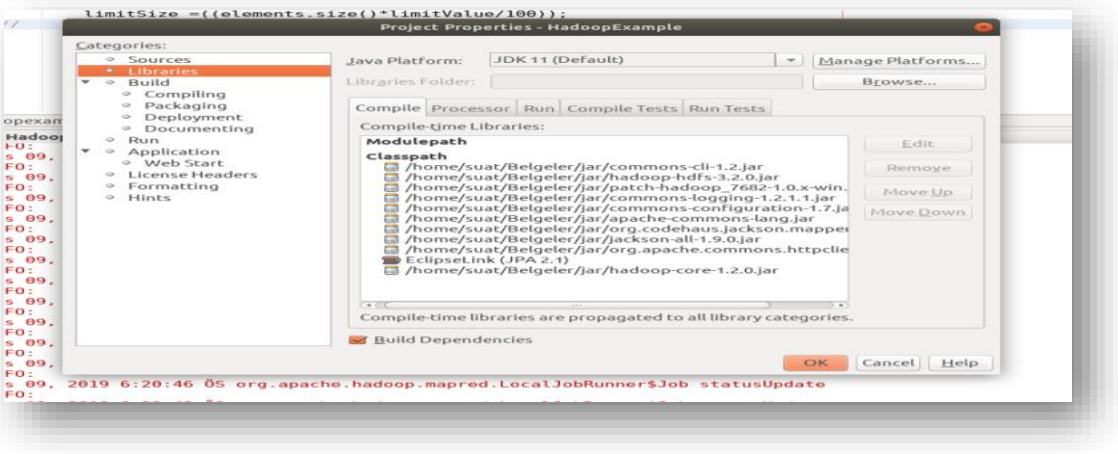
Birleştirme işlememi Şekil 4.4.'te görüldüğü gibi ayrı bir metot içinde yapılmıştır. Bu metot içine gelen listede ise küçük bayt sahibi veriler dizinin en önüne alınmaktadır. Sıralı bu listede en küçükten başlayarak en büyük verilerin sona alınması ile oluşmaktadır.

Java kütüphaneleri Hadoop yazılımıgeliştirmek için bize kolaylık sağlamaktadır. Bu kütüphaneleri sisteme ekler ve gerekli noktalarda bu yapılar kullanılmaktadır. Geliştirilen yapıda Hadoop sistemine ait kütüphaneler dışında veri incelemek için kullandığımız başka ek kütüphaneler de vardır.



Şekil 4.5. Birleştirme Algoritması.

Haritalama ve indirgemeye ekleyeceğimiz yeni yaklaşım, Hadoop'a ait sınıflar dönüştürülerek gerçekleştirilmiştir. Bu sebepten dolayı, Hadoop'a ait kütüphaneleri kendi yazlımıza ekledik. Şekil 4.6. Geliştirilen yazılım içine eklenen kütüphaneler gözükmektedir. Bu kütüphanede hazır metotlar yer almaktadır. Geliştirilen yazılımda bu metotlardan haritalama ve indirgeme metotları yeniden gözden geçirilmiş ve önermiş olduğumuz algoritmaya göre ezilerek, yeniden yazılmıştır.



Şekil 4.6. Kullanılan Kütüphaneler.

Öncelikle Hadoop için sonuç ve veri alımı yapacağımız dosya yollarını belirtmemiz gerekmektedir. Kendi geliştirdiğimiz haritalama ve indirgeme sınıflarını işyapısı içine aktarmaktayız. İndirgeme işlemi, her durumda çalışması gerekli değildir. Bu sebepten dolayı, indirgeme olmaması için sıfır atanması imkanlı hale getirilmiştir. İş yapısına göre anahtar değerleri çıktı tipler olarak atanmıştır. Hadoop benzer tipten verilerin işlenmesi esnasında çok hassas davranmaktadır. Buralarda gerekli dikkati göstermemiz sağlıklı işleyen bir modül için gereklidir.

Haritalama sıralama daha öncede birçok çalışmada denenmiştir. Biz bu denemelerin bayt olarak sıralanması yaklaşımın verimli olacağını göstermeyi amaçladık. Bu sıralamadan oluşan ek zaman farkını azaltmak için bahsettiğimiz üç yapılı liste üzerinde bağlı listeler kullanılmıştır. Böylece daha fazla hızlı işlem yaparak maliyet kazancımızı arttırdık. Sıralamada öncelikle maliyeti az olan verileri ön tarafa taşıyarak ileride yapılacak işlemlerimize de maliyet kazancı sağlamış olmaktayız.

Hadoop kısaca yönlendirebileceğimiz ve geliştirebileceğimiz bie yazılımdır. Kendi geliştirdiğimiz sınıflarımızı Hadoop sitemine aktararak geliştirdiğimiz yeni yaklaşımı uygulama fırsatı bulmaktayız. Şekil 4.7. üzerinde gösterilen kod sayfasında öncelikle Hadoop içine belgelin alınacağı ve yazılacağı klasörlerin adresleri verilmiştir. Hadoop'a ait iş yapısına kendi yazdığımız MyMapper sınıfını yerleştirdik ardından

çıkış verilerinin türlerini belirttik. Haritalamada anahtar değer InWritablesonuç değerleri ise Text olacak şekilde ayarlar yapılmıştır. İndirgeme işlemi için MyReduce sınıfını geliştirdik ve ardından Hadoop iş yapısı içine ekledik. Son olarak indirgeme işlemi her defasında yapılmak zorunda değildir. Burada kullanıcı kendi ekranından haritalama sonrasında indirgeme işlemine izin vermez ise Hadoop indirgeme iş yapısını sıfır değeri atanır. İndirgemeye izin verilmiş ise zaten otomatik olarak sistem burada bir sayısını kullanır.

```
try {
    FileInputFormat.setInputPaths(job, new Path("/home/suat/Belgeler/inputDir"));
} catch (IOException ex) {
    Logger.getLogger(HadoopFrame.class.getName()).log(Level.SEVERE, null, ex);
}
FileOutputFormat.setOutputPath(job, new Path("/home/suat/Belgeler/outputDir"));

job.setMapperClass(MyMapper.class);

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

job.setReducerClass(MyReducer.class);
if (!TgBReduce.isSelected()) {
    onlyMap = false;
    job.setNumReduceTasks(0);
}

System.exit(job.waitForCompletion(true) ? 0 : 1);

if (job.isSuccessful()) {
    System.out.println("Job was successful");
} else if (!job.isSuccessful()) {
    System.out.println("Job was not successful");
}
```

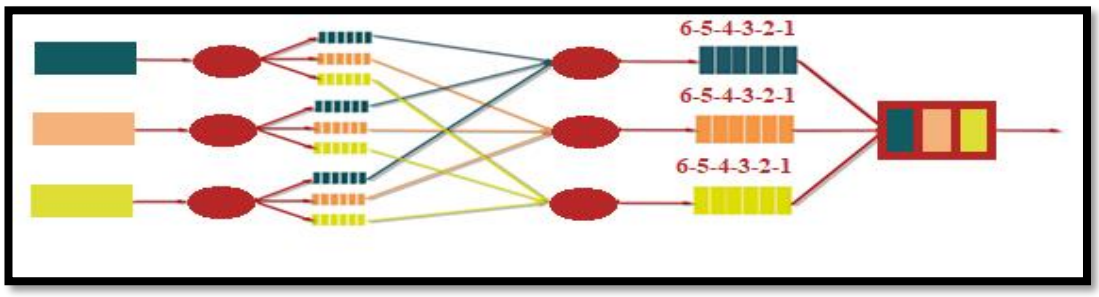
Şekil 4.7. Hadoop Üzerinde İş Parçaları Tanımlanması.

4.1. Veri Haritalama ve İndirgeme

Haritalama ve indirgeme dağıtık mimari yapıda olan ve büyük verilerin kolay bir şekilde analizini sağlayan bir haritalama ve işleme yapan sistemdir. 2004 yılında Google tarafından kullanıma sunulan bu sistem gerçekte 1960'lı yıllarda geliştirilmiş bir yapıdır. Fonksiyonel programlamadaki haritalama ve indirgeme yapısından esinlenerek oluşturulmuştur.

Büyük veri haritalama aşamasında ana düğüm tarafından alınıp daha ufak parçacıklara ayırarak işçi düğümlerine dağıtır. Bu işçi düğümler bu işleri tamamlar ve sonucunu ana düğüme tekrar geri yollarlar. İndirgeme aşamasında ise tamamlanmak istenilen işler tekrar birleştirilerek sonuçlar elde edilir. Şekil 4.8'de haritalama ve indirgeme

için örnek gösterim şeması yer alırken, Şekil 4.9'da ise haritalama ve indirgeme işlemi arasındaki sıralama işlemi verilmiştir.



Şekil 4.8. Haritalama ve indirgeme için örnek gösterim şeması.

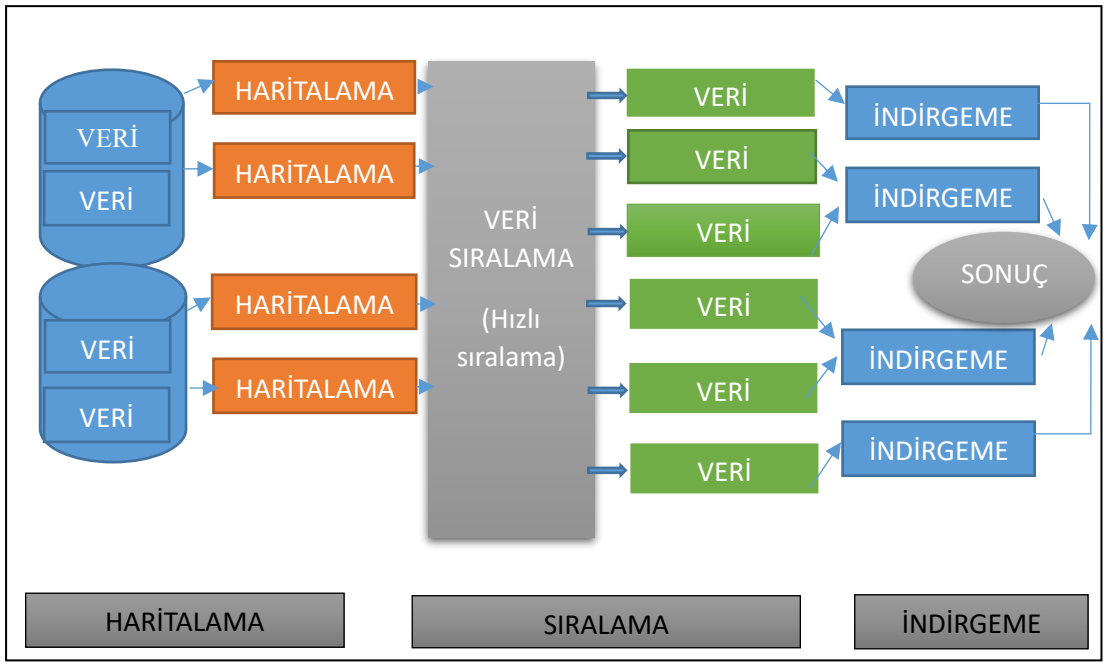
Haritalama İndirgeme, büyük veriyi devasa kümelerde haritalandırmak ve işlemek için oluşturulmuş bir dağıtık programlama modelidir. Bu yapı iki ana yapıdan oluşur. Birincisi haritalama; bir yığının tüm üyelerini sahip olduğu fonksiyonları ile çalışarak bir sonuç haritasına döndürür. İkincisi indirgeme; paralel bir şekilde çalışan iki veya daha fazla haritalama fonksiyonundan dönen sonuçları veya sonuç listelerini harmanlar ve haritalandırmaya bağlı çözümler sunar. Hem haritalama hem de indirgeme paralel bir şekilde çalışırlar, aynı anda aynı sistem üzerinde olmaları beklenmez. Burada yapılacak haritalama ne kadar sağlıklı ve düzgün sağlanırsa indirgeme fonksiyonundan da bir o kadar sağlıklı veri alınır.

Haritalama işlemi verileri parçalar ve paketler halinde etiketlemektedir, kısaca veri kümelerine dönüştürür. Her işlem için iş parçacığı oluşturulur, işlemci veya sistem, her kaydı inceleyerek ara değer içeren haritalar oluşturulur.

1. Tüm veriler dağıtılabilir olmalıdır.
2. Küresel kalıp (Nihai çıktı) hepsinden elde edilmelidir.
3. Problem harita indirgenebilir olmalıdır.

İndirgeme fonksiyonu, hafızadan, diskten veya ağ transferi ile elde edilen ara sonuç listelerini işler ve sonucu döner. Bu işlemlerin çalışması haritalama işlemine bağlı

olarak değişir. Haritalama yapısında aktarılan veriler indirgemede paralel olarak listelenmiştir.



Şekil 4.9. Haritalama indirgeme arasında olan sıralama işlemi.

Haritalama işlemi, dağınık halde bulunan ağda, sunucuda veya disklerdeki verilerin her birinin nerelerde olduğunu önemsemeksizin belli bir metodu o verilere uygular ve kendi diski üzerinde tutmaktadır. Daha sonra indirgeme işlemi her bir veriyi alır ve belli bir kurala göre analiz eder ve analiz edilmiş veriyi çıktı olarak verir.

Bizim amacımız, haritalama ile tutulan verileri belli bir hizaya kavuşturmaktır. Harita düzeni veri boyutlarına göre yapılırsa istendiği taktirde daha az boyutta ve daha çok veri ile işlem gerçekleştirilebilir. İndirgeme işlemini kullanıcı istediği taktirde tüm verilere uygulamayabilecektir. Bu işlemlerin sonrasında örneklem seçmek gerekmektedir. Veri içindeki doğal sınırlar, veri analizi vb. işlemya da algoritmaları da dikkate alınarak, az kaynak ve az maliyetle çok fazla veri analizi yapabilmektedir.

Verilerin sürekli artığı bu çağda kaynaklar aynı oranda artmadığı, maliyetlerin yükseldiği kuşku götürmez bir gerçektir. Örneklem seçimi ve analizinde haritalama sonrası yapılacak bu ara işlemler ile uygun çözüm sağlanacaktır.

Ayarlar oluşturulur ve Apache Hadoop üzerinden alınan ayarlar bir işleme dönüştürülür. Kütüphane oluşturacak sınıfı çağırılır ve sonrasında haritalamaindirgeme işleme eklenir. Bundan sonra çağırılan sınıflar sonuçları yazan sınıflardır. İşlemin planlamasını haritalama, gerçekleşmesini indirgeme sağlamaktadırr. Şekil 4.10'da ise indirgeme alt program çıktısı yer almaktadır.

```
)verride
ublic void reduce(IntWritable key, Iterable<Text> value, Context context) throws IOException, InterruptedException {


   List<String> newList = Arrays.asList(context.getCurrentValue().toString().substring(1, context.getCurrentValue().toString().length()-

   String tempValue = newList.get(2).trim();
   Double tempFreq = Double.valueOf(newList.get(1).trim());

   if (tempValue.trim().getBytes("UTF-8").length >= HadoopFrame.getAveLenght() && tempFreq >= HadoopFrame.getAveRepeat()) {
       HadoopFrame.truetrue.add(" lenght = "+tempValue.length()+" Value = "+ tempValue);
   } else if (tempValue.trim().getBytes("UTF-8").length <= HadoopFrame.getAveLenght() && tempFreq <= HadoopFrame.getAveRepeat()) {
       HadoopFrame.falsefalse.add(" lenght = "+tempValue.length()+" Value = "+ tempValue);
   } else if (tempValue.trim().getBytes("UTF-8").length <= HadoopFrame.getAveLenght() && tempFreq >= HadoopFrame.getAveRepeat()) {
       HadoopFrame.falsetrue.add(" lenght = "+tempValue.length()+" Value = "+ tempValue);
   } else if (tempValue.trim().getBytes("UTF-8").length >= HadoopFrame.getAveLenght() && tempFreq <= HadoopFrame.getAveRepeat()) {
       HadoopFrame.truefalse.add(" lenght = "+tempValue.length()+" Value = "+ tempValue);
   }

   if(Objects.equals(Integer.valueOf(key.toString()), HadoopFrame.getAllValueListSize())){
       int newIndex = 0;
       for (String values : HadoopFrame.truetrue) {
           newIndex++;
           context.write(new IntWritable(newIndex), new Text(values));

       }

       for (String values : HadoopFrame.falsetrue) {
           newIndex++;
           context.write(new IntWritable(newIndex), new Text(values));

       }

       for (String values : HadoopFrame.truefalse) {
           newIndex++;
           context.write(new IntWritable(newIndex), new Text(values));

       }


       for (String values : HadoopFrame.falsefalse) {
           newIndex++;
           context.write(new IntWritable(newIndex), new Text(values));

       }
   }
```

Şekil 4.10. İndirgeme metot görseli.

İndirgeme işleminiHadoop'a ait sınıfın ezilmesi ile oluşturduk. Burada verilerin önem değerlerine göre bir sıralamaya sahip olmasını sağladık. İşlenen değerlerini ortalama bayt ve ortalama tekrar sayısı ile oluşurduk. Bu değerlere göre dört parçalı bir birleşim listesi de oluşturduk ve çıktı olarak aktardık. Şekil 4.10.'de oluşturduğumuz bu çıktı indirgeme metodu sonucudur. Öncelikle global değişkenler tutulan ortalama baytuzunlu ve ortalama veri tekrar sayısı metot içerisine alınmıştır. İndirgeme işlemine gelen her değerin tekrar sayısı ve sahip olduğu bayt uzunluğu hesaplanır. Öncelik ile sıralama işleminde ortalama tekrar adedini sağlayanlar ve bayt uzunluğu sağlayanlar sonrasında sırası ile, ortalama tekrar adedini sağlayanlar ve bayt uzunluğu sağlamayanlar, ortalama tekrar adedini sağlamayanlar ve bayt uzunluğu sağlayanlar son olarak iki şartı sağlamayanlardan oluşturulmuştur.

BÖLÜM 5. UYGULAMA

Bu çalışmada ana yapı Hadoop üzerine kurulduğu için en verimli alt yapının Linux üzerinde sağlanacağı düşüncesi ile öncelikle Linux üzerinde kurulumlar yapılmıştır. Bu noktada Linux versiyonlarından Ubuntu tercih edilmiştir. Ubuntu https://ubuntu.com/ adresinden elde edilebilir ve ayrıca Hadoop ise https://hadoop.apache.org/ üzerinden indirilebilir. Bu iki ana kurulum sonrasında Java sanal makinesi ve Hadoop çalışmasında sorun yaşamamak için ssl kurulumu da yapıldı. Hadoop'a ait kütüphane dosyalarına ihtiyacımız vardır. Bunun sebebi Hadoop sisteminde olan metotları ezip kendi metotlarımız sisteme yerleştirecek olmamızdır. Bu jar dosyaları https://jar-download.com/ elde edilebilir. Üzerinde çalışma yapılan hdfs-jar dosyası 3.2.0'dır. Bunun dışında birçok yerden bu gerekli yazılım kurulum dosyalarını ve kütüphane dosyalarını elde edebiliriz. Java versiyonlarından JDK 8 kullanılmıştır.

Çalışmada kullanılan örneklem dosyası 2012 yılında agresif bir büyüme planı uygulayan bir sigorta firmasının 2012 yılına ait 36634 kayıttan oluşan veri kümesi Sample insurance portfolio (Örnek sigorta portföyü) isimli dosyadır. Bu veri yığınında on dokuz kolon vardır. Özellikle bu dosya coğrafi kodlamanın seçilebileceği adres bilgisine sahiptir ve dosyadaki mevcut enlem / boylam kullanılabilmektedir. Dosya boyutu 4.124.255 bayt. Dosya formatı cvs'dir. Dosya cvs formatı verilerin virgülle ayrılmasıyla oluşur. Bu ilgili cvs dosyayı https://ckan.coegss.hlrs.de/ [41] üzerinden elde edebiliriz. Bu dosya numara ve metinlerden oluşmaktadır. Bu dosyada iki uygulama yapılmıştır. Bunlardan birincisi, verileri azaltacağız sadece veri azalttım ile birlikte işlem esnasında yeni değerler elde edilecektir. İkinci denemede ise aynı şekilde veri azaltımı amaçlanmaktadır ve aynı yöntemleri uygulayarak fakat ek olarak daha

fazla çıktı dosyasında olan verilerin indekslemesi veri sonucunda oluşan dosyanın küçültülmesi ile maliyet kazınıcıda arttırılacaktır.

Bu bölümde önerilen hızlı sıralama algoritmasının Sample insurance portfolio üzerindeki uygulamasına yer verilmiştir. Özellikle Sample insurance portfolio tercih edilmesinin nedeni büyük veri yapısını en iyi bir biçimde temsil ettiği ve hızlı bir şekilde gelişen bir bölgeye ait emlak bilgilerinin yer alması sebebiyle aynı zamanda dinamik yapısı da büyük veri için iyi bir örnek oluşturmuş olmasından dolayı seçilmiştir.

5.1. Giriş

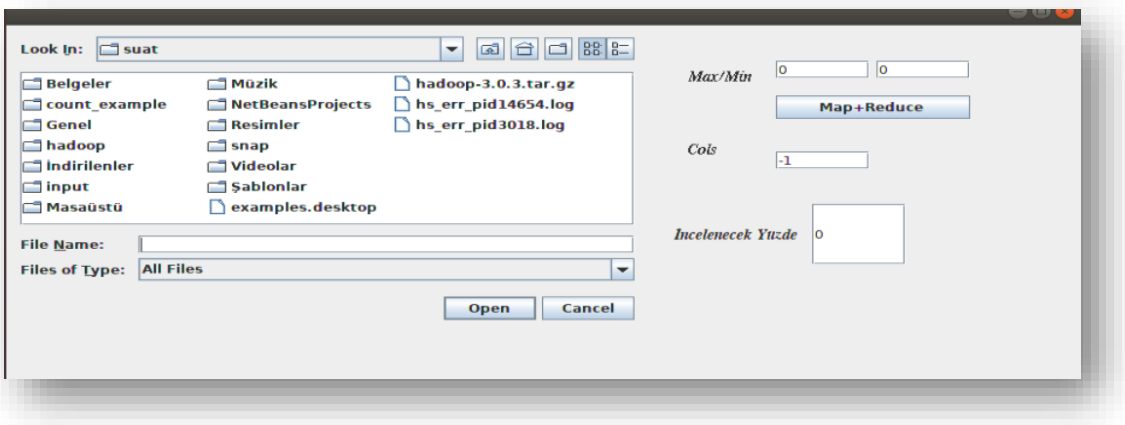
Java dilinde grafiksel kullanıcı arayüzü geliştirilmiştir. Kullanım esnasında Java ait Swing kütüphanesi kullanılarak Hadoop'a ait kütüphaneler ve dosya yönetim kütüphaneleri oluşturulmuştur. Önerilen işlemler için geliştirilen kullanıcı grafik arayüzü Şekil 5.1.'de gösterilmiştir. Bu geliştirmelerde geliştirme Java olarak yapılmıştır. Şekil 5.2'de programın çalışması esnasındaki ekran yer almaktadır. Ön yüzde olan alanların açıklamasını maddeler olarak halinde belirtilmiştir.

5.2. Uygulama

Uygulama çalışma boyunca öne sürülen önerilerin soyut bir noktadan somut bir noktaya taşımıştır. Bu noktada verilerin haritala öncesi sınırlanması ve sonrasında sıralanması incelenebilecektir. Geliştirilen uygulama özelliklerinin kullanıldığı ve kullanılmadığı durumlarda gözlenmesi için iki ayrı deneme yapılmıştır. Hadoop yapısında çok yararlı bir yapı olarak geliştirilen haritalama ve indirgeme yeni yaklaşımımız ile daha verimlim ve az maliyetli bir yapıya kavuşulaacaktır.

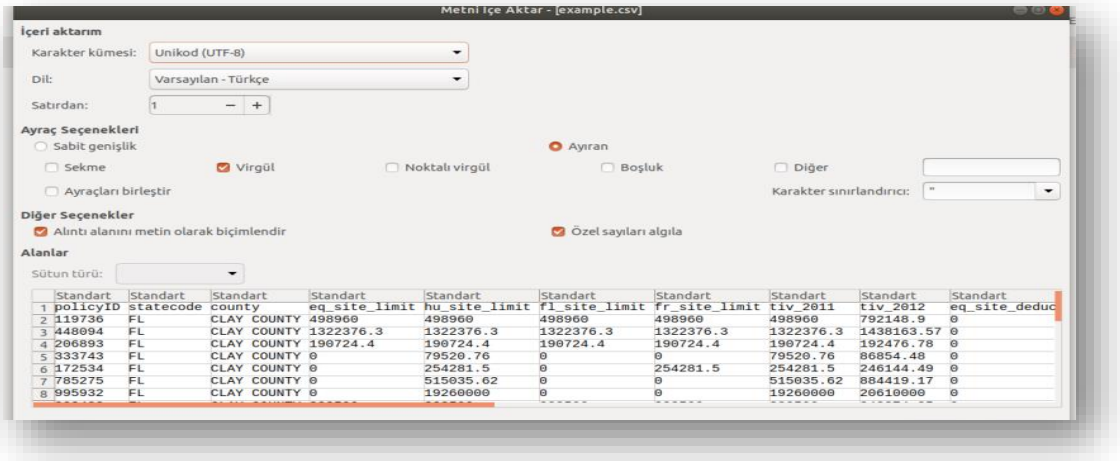
1. Open: Dosya seçimini onaylar.
2. Max/Min: İncelenecek verilerin bayt sınırlarını tespit eder.

3. Map+Reduce: Basılı durumda ise haritalama-indirgeme basılı değil ise sadece haritalama işlemi yapar.
4. Cols: Cvs ve benzeri dosya uzantılarında open'a basıldığında dosya içinde ilgili kolonu seçmek ve buna uygun bir indeksleme oluşturmak için kullanılır. Bu seçimde amaç verilerin homojenliğini arttırır. İstenilen kolon girilir veya pasif halde tutulmak için -1 değeri atanır
5. İncelenecek Yüzde: Girilen veri setinde verilen oran kadar en az ve en yoğun olan veri miktarını almak için belirtilir.



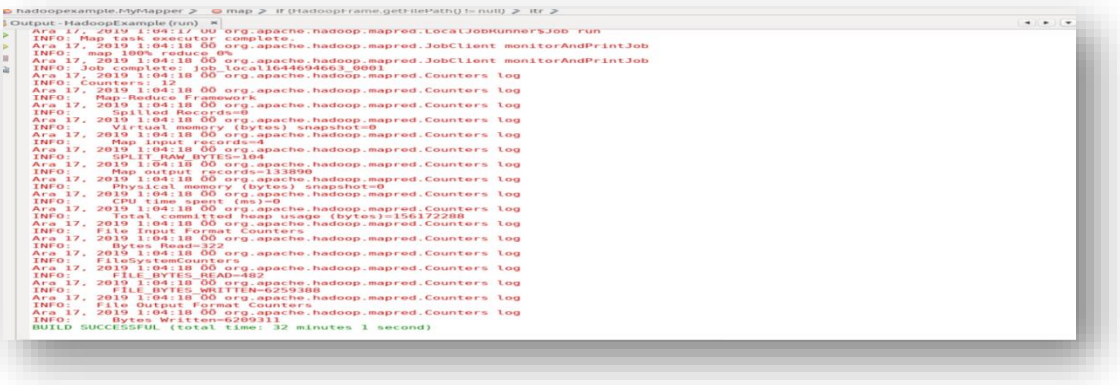
Şekil 5.1. Uygulama Ekranı.

CSV (Virgülle Ayrılan Değerlerden Oluşur) dosyadır. Bu dosya verilerini sütunlarda yerleşik olarak sunmak yerine virgülle ayrılmış olarak depolar. İçindeki Metin ve sayılar bir CSV dosyası olarak kaydedildiğinde kolaylıkla parçalanarak analiz edilebilir.



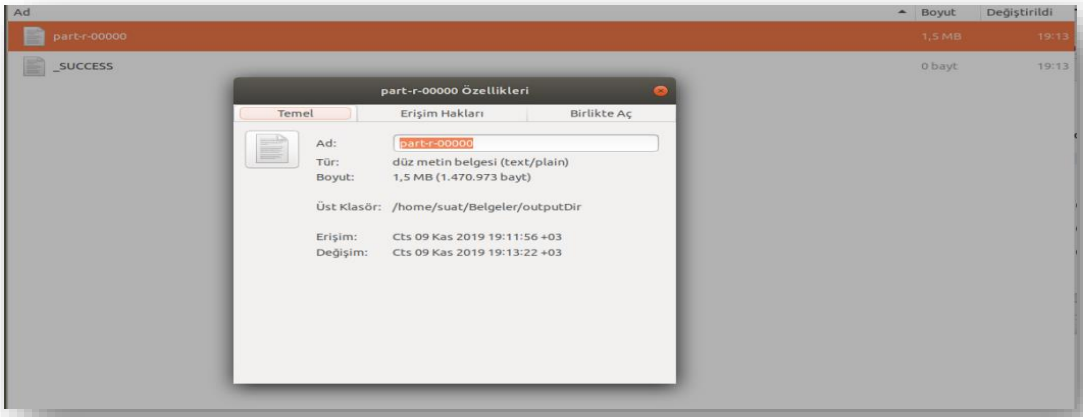
Şekil 5.2. Örnek kullanılan cvs dosya.

Örnek çalışmasında haritalandırılan değerler virgül ile ayrılmıştır veri değerlerinin toplamı kısaca hücre adedi 659.430 olmasına rağmen sınır değerimiz alınmamış harita boyutu kısa tutulmamıştır. Ayrıca indirgeme aşamasında çok ve doğru sonuç hızlıca elde edilebilmesi amaçlandığı için sıralama azdan çoğa doğru olmuştur (Şekil 5.3). Sıralama sonucunda oluşan sistemsel zaman farkı içine sıralama algoritması dahildir. Pivot değeri ortanca değerden alınarak seçilmiştir, ortanca değeri analiz etmeye başlayarak sırama yapılacaktır. Burada kullanılan yapıda pivot seçimi ve kullanılan donanımsal özellikler süre durumu değiştirebilir. Amaç yazılım olarak iyileştirme sağlamak olduğu için bu konu göz ardı edilmiştir.



Şekil 5.3. Çalışma ekranı.

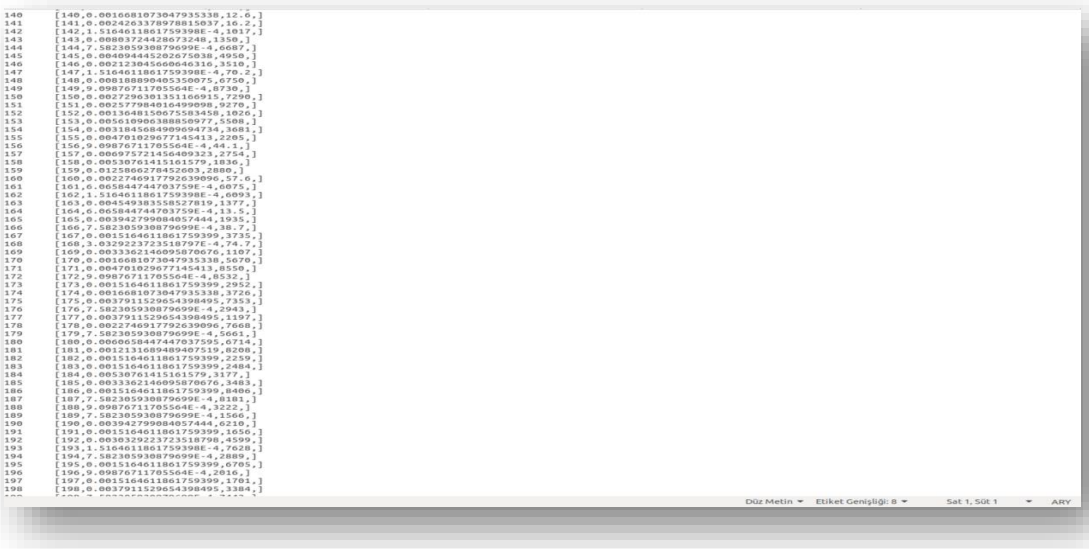
Uygulama çalıştıktan sonra part ismi ile başlayan bir sonuç dosyası oluşturur (Şekil 5.4). Çalıştırılan haritalama ise part-m veya haritalama ardından indirgeme çalıştırılmış ise part-r olarak başlayan çıktı sonuç dosyası alınmaktadır. Sonuçdosyası sistemde belirtilen çıktı yolu üzerinde oluşturulur. Bu uygulama, verilen çıktı yolunda outputDir klasörü tanımlanmıştır. Sonuç dosyaları burada otomatik olarak oluşmaktadır. Bu sonuç dosyaları bize üzerinde çalışacağım verileri düzenli bir şekilde vermektedir.



Şekil 5.4. Sonuç dosyası.

5.3. Birinci Deneme

1. Veri yığını veri sınırlama yapılmamıştır.
2. Verilerde ilk işleme girdiği indeks değerleri tutulmuş. Haritalamada gösterilmiştir
3. Veri Boyutu bayt bazında sınırlanmamıştır.
4. Veri sınırlama ve indeksleme olmadığı durumun sonucunu göstermeyi amaçlamıştır.



Şekil 5.5. Sadece haritalama birinci deneme.

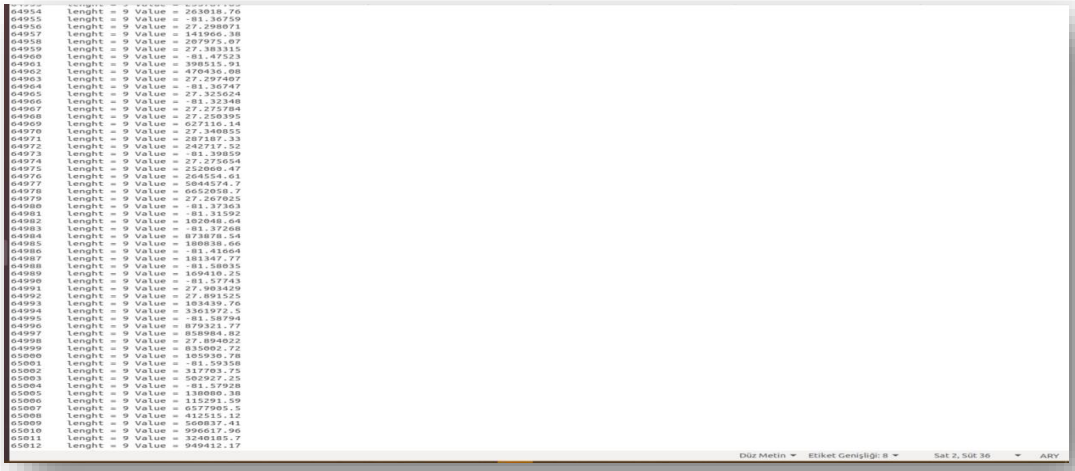
Haritalama işlemi çıktısında sıra numarası, verinin yığın içinde olan tekrar yüzdesi ile puanlamış ve işlenen verinin bilgisi köşeli parantezler içinde gösterilmiştir. Şekil 5.5. haritala işlemi sonrası elde edilmiştir. Bu işlemde 659.430 veri ele alınmış ve haritalama aşaması sonrası 113.389 olarak azaltılmış bu veriler tekrar eden değerlerdir. İlk noktaları ve toplam adetlerine göre frekans değerleri hesaplanmıştır. İşlem sonrası uzunluğa bağlı elimizde veri yığını oluştu. Haritalama işlemi 11 dakika 25 saniye sürmüş ve bir harita dosyası oluşturulmuştur. 6.161.167 bayt büyüklüğünde ve part-m-00000 isimli dosya oluşmuştur. Haritalama işlemi sonunda sistem nano time olarak 6577001052929 olarak kayda alınmıştır (Şekil 5.6). İndeks listesi ve listedeki her elemana ait frekans puanı ve haritalamada analizi edilmiştir.

Şekil 5.6. Haritalama çalışma grafikleri birinci deneme.

1. (GC) uygulamanın hafıza kullanımını ile (CPU) İşlemci kullanımı gösterir
2. (Heap Size) uygulamanın hafıza konuşlandığı alan ile UsedHeap kullandığı alanı gösterir
3. GarbageCollector üzerinden çöplerin toplanması işlemi grafiğidir. Hafızanın geri alınması görevini GarbageCollector yapar. (GC) aralığı iki çöp toplama arasını işlemleri ve (SurvivingGeneration) en az bir çöp koleksiyonundan kurtulanları objeleri gösterir.
4. (Threads) İş parçacıkları ile Yüklenmiş (LoadedClasses) Yüklenmiş sınıfları gösterir

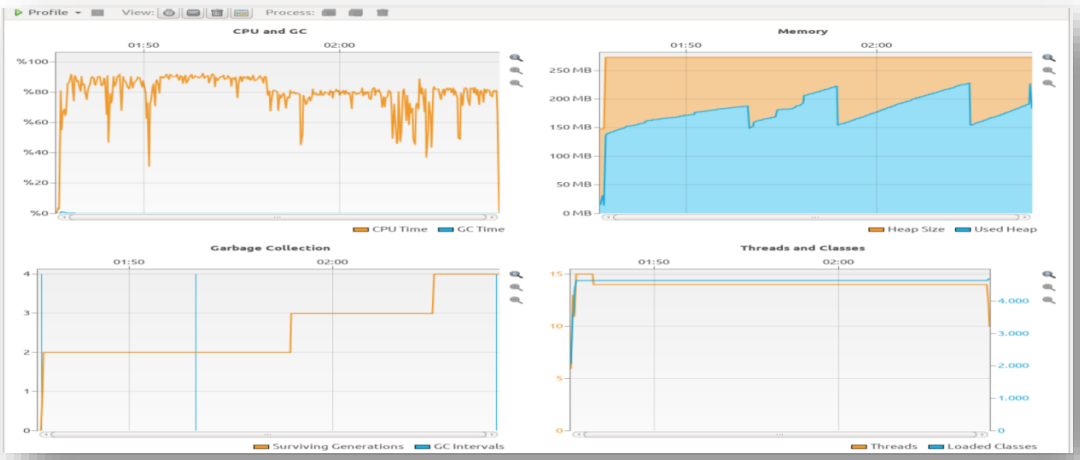
İndirgeme işlemi yapılması için kolon seçimi yapılmamıştır. Bundan dolayı tekrarda öncelikli ve ortalama tekrar sayılarını verenler başta olmak üzere tüm veriler puanlanmış ve sıralı bir sonuç çıktısı vermiştir. Sonuç çıktısı Şekil 5.7.'de gösterildiği gibi indirgeme işleminde sıralı olarak 113.389 veriden oluşan 4 ayrı listenin birleşim sonucunu vermiştir. Sıralama ortalama uzunluk ve ortalama frekans göz önünde tutularak elde edilmiştir. 6.707.094 bayt büyüklüğünde ve part-r-00000 isimli dosya oluşmuştur (Şekil 5.8). Liste ortalama uzunluk ve ortalama frekans şartını

sağlayanlardan başlar ve sıra ortalama uzunluk sağlayan fakat ortalama frekans şartını sağlamayanlar, ortalama uzunluk sağlamayanlar fakat ortalama frekans şartını sağlayanlar son olarak da her iki şartında sağlamadığı değerlerin listesi olarak sonuçlanır.



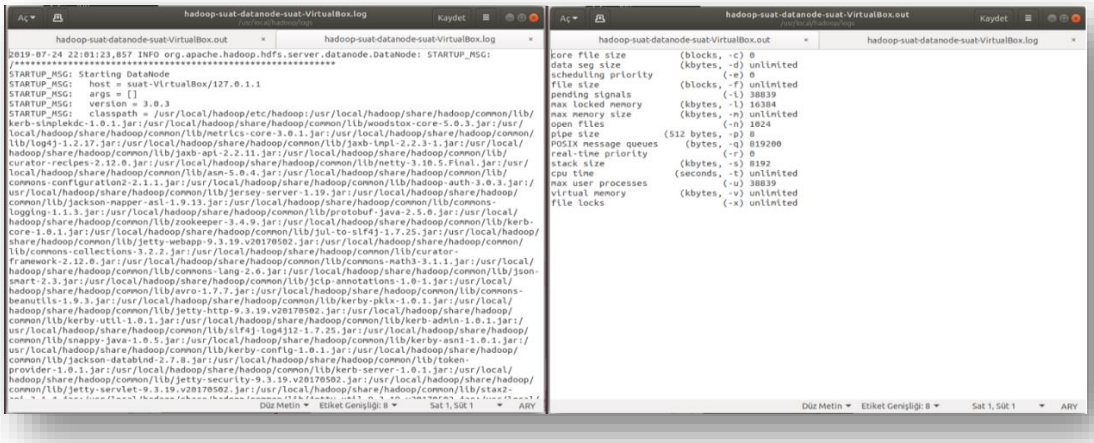
Şekil 5.7. Sonuç haritalama indirgeme birinci deneme.

Tutuğumuz veri miktarı azalmasına karşın dosya boyutu artmıştır. Bunun sebepleri işlenen değerler dışında işlemde oluşan değerlerin çıktı olarak yazdırılmasıdır. Ayrıca Hadoop üzerinde çıkan ve yazdırılan sonuç dosyasının dosya boyutunda artış sağlamıştır. Buradan ortaya çıkan sonuç. Maliyetleri azaltmak için sadece veri adedini düşürmemiz yetmez. Her veriler için uygun sınırlama ve indeksleme kullanmalıdır. İkinci yapacağımız denemede bu noktalarında geliştirdiğim yazılımda sonucunu olumlu olduğunu gösterilmiştir.



Şekil 5.8. Haritalama indirgeme çalışma grafikleri birinci deneme.

Hadoop yazılımında işlenen verilerin ve oluşan işlemlerin takip edildiği log çıktıları oluşturulmuştur. İşlemler sonrasında Hadoop işlem dosyalarında olan log dosyasını inceleyebiliriz. Bu log dosyasına ait çıktı Şekil 5.9. gösterilmiştir.Bu noktadan anlaşıldığı gibi işlemler devam ederken Hadoop'da arka taraftan log işlemine devam edilmiştir. İşlemin toplam zamanı 23 dakika 35 saniye olarak log düşürülmüştür. Okunan dosya 4.911.589 bayt yazılan dosya ise 1.482.473 bayt olarak kayda geçmiştir. Okunan dosya cvs formatındadır.



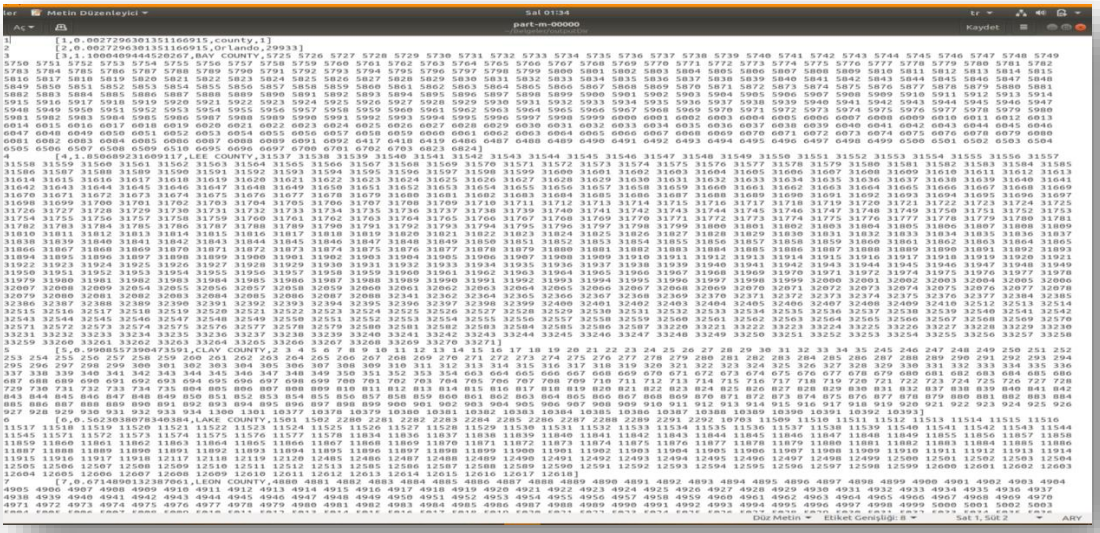
Şekil 5.9. Hadoop log.

5.4. İkinci Deneme

İkinci deneme için gerçekleştirilen işlem basamakları aşağıda verilmiştir.

1. Veri yığınında bir kolon seçilmiş böylece veri sınırlama yapılmıştır.
2. Verilerde tüm işleme girdiği indekslerin değerleri tutulmuş.
3. Veri Boyutu bayt bazında sınırlanmamıştır.
4. Veri sınırlama ve indeksleme olduğu durumun sonucunu göstermeyi amaçlamıştır.

Sıralama işlemi sonrası indirgeme işlemi yapılırsa sonuç çıktısında veriler sıralanmıştır. Burada köşeli parantez içinde verinin sahip olduğu değer puanı ikinci sırada sonrasında verinin kendisi ardından bulunduğu indeks adresleri sıralı bir şekilde verilmiştir. Yapılan deneme sonucu Şekil 5.10'da bize gösterilen sonuçta çok büyük kazanım elde edilmiştir. İşlemde 659.430 veri ele alınmış ve haritalama aşaması sonrasında sonuç 68 adet veriden oluşmuş ayrıca bu veriler tekrar eden değerlerdir. Toplam adetlerine göre frekans değerleri hesaplanmıştır. Bunun dışında ortalama verilerin bayt uzunluğu hesaplamamıştır.İşlem sonrası uzunluğa bağlı elimizde veri yığını oluştu. Haritalama işlemi 28 saniye sürdü ve bir harita dosyası oluşturdu. 211.54 bayt büyüklüğünde ve part-m-00000 isimli dosya oluşmuştur. Haritalama işlemi sonunda sistem nano time olarak 46106917359782 olarak kayda alınmıştır.İncelenen değerlerin hepsine ait indeks listesi değerin yanında kayda alınmıştır.

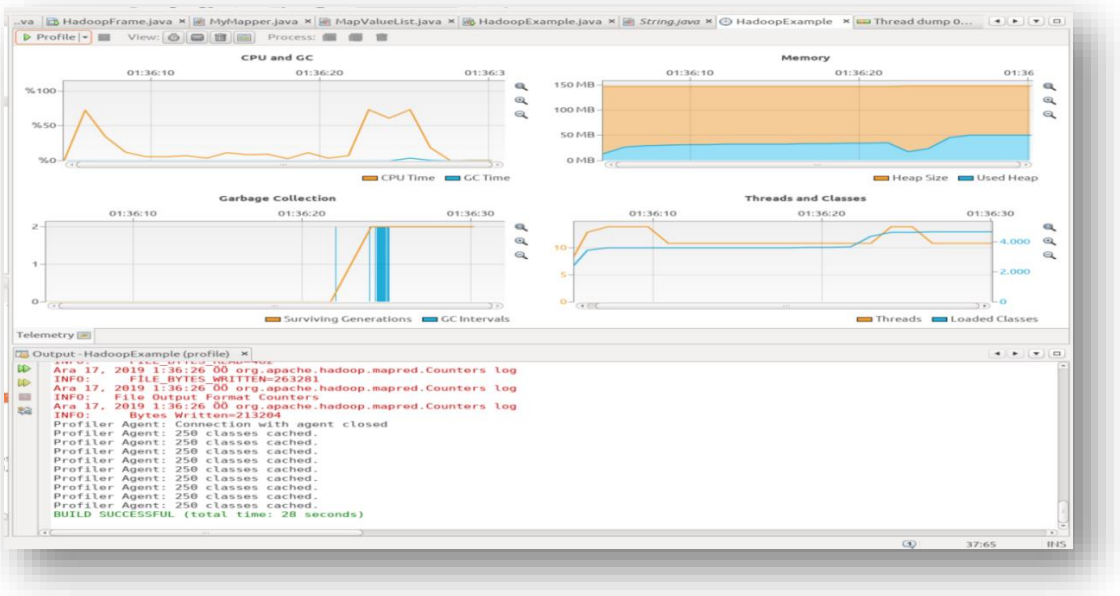


Şekil 5.10. Sadece haritalama ikinci deneme.

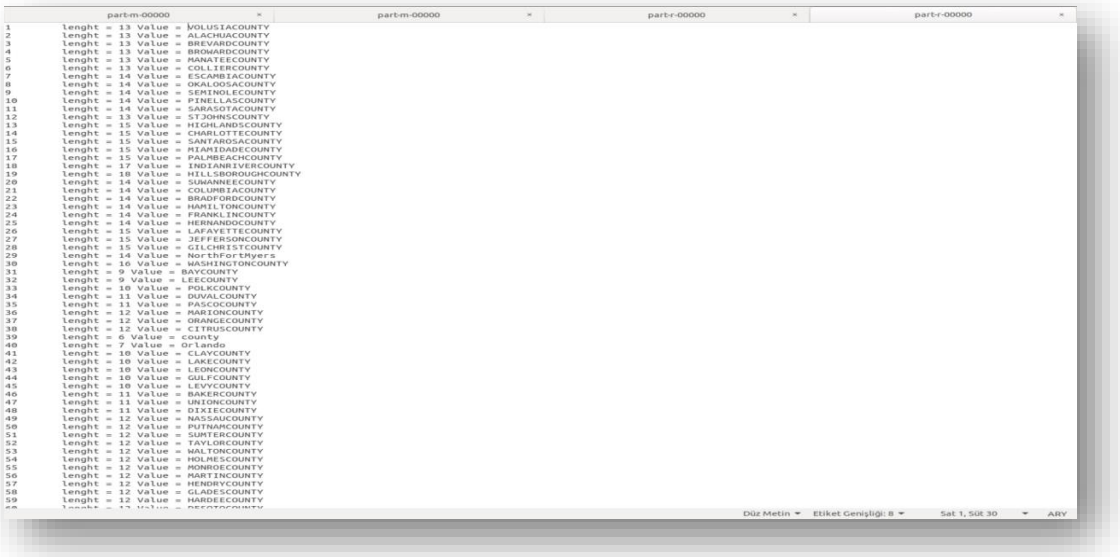
İki denemde sadece haritama çalışma grafikleri sonuçlarının (Şekil 5.11 ve Şekil 5.6) görselleri bize çözüm konusunda yardımcı olmaktadır. İkinci denemede veri sınırlama ve indeksleme kullanımı yoğun tam anlamı ile sağladığı için istatistikler olumlu yöne geçmiştir. Kullanılan geçici hafıza büyük oranda düşmüştür. Uygulamanın hafıza ve işlemci kullanımı çok azalmıştır ve ayrıca oluşan artık, kullanılmayan çöpl verilerinde azalığını görebiliriz. İki deneme için oluşan bu grafik farkları haritalama sonrası indirgeme yapılan işlemlerin grafiklerinde de oldukça kolay bir şekilde aynı olumlu yönde gözlenebilir.

Haritalama işlemi sonrasında eğer indirgeme yapılmaz ise oluşan sonuç listesinin bir bölümü Şekil 5.12'de gösterilmiştir. Burada sonuç olarak indirgemeişleminde sıralı olarak 68 veriden oluşan 4 ayrı listenin birleşim sonucunu vermiştir. 2.547 bayt büyüklüğünde ve part-r-00000 isimli dosya oluşmuştur. Sıralama ortalama uzunluk ve ortalama frekans göz önünde tutularak elde edilmiştir. Liste ortalama uzunluk ve ortalama frekans şartını sağlayanlardan başlar ve sıra ortalama uzunluk sağlayan fakat ortalama frekans şartını sağlamayanlar, ortalama uzunluk sağlamayanlar fakat

ortalama frekans şartını sağlayanlar son olaraktan her iki şartında sağlamadığı değerlerin listesi olarak sonuçlanır.



Şekil 5.11. Haritalama çalışma grafikleri ikinci deneme.

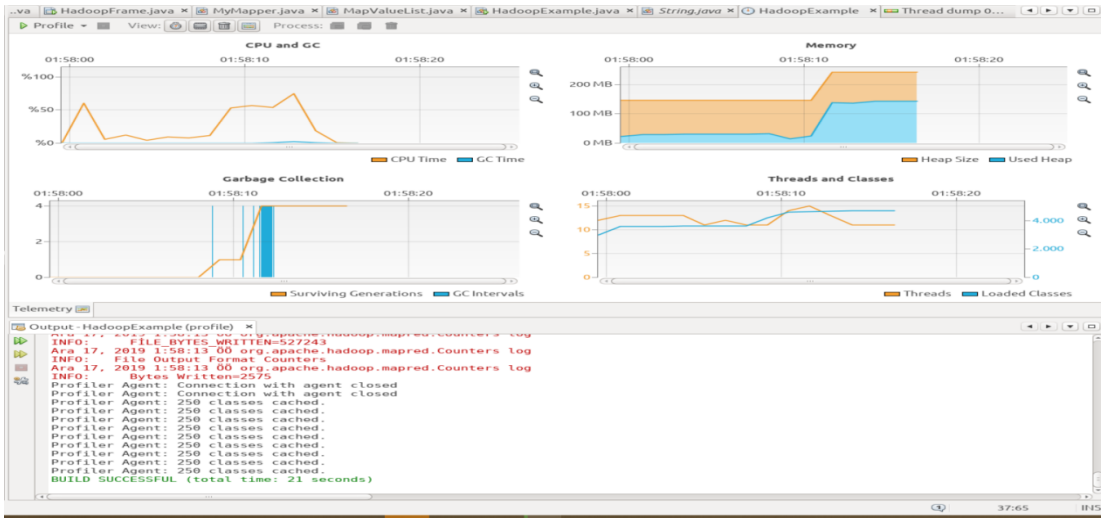


Şekil 5.12. Sonuç Haritalama indirgeme ikinci deneme.

İki deneme arasında oluşan farkın sebebi verilerin homojen olmasıdır. Analizde frekans bazında puanlamayı çok daha verimli hale getirmiştir. İkinci denemede sadece

bir kolondaki veriler puanlamıştır. Bu Veri değerlendirmede sınırlar belirtildikçe değerleme verimi artıracağının kanıtıdır (Şekil 5.13 ve Şekil 5.14).

2012'de Florida'da oluşan bu agresif büyüme analiz etmemek istendiği taktirde yapılan bu denemeler bazında VolusiaCounty analizinde en az maliyet ile en değerli verilerin olduğu yerdir. Bu veri yığını üzerinde yapılacak en az maliyetli ve en değerli veri çalışmasında VolusiaCounty'e ait veriler öncelikle kullanılmalıdır.

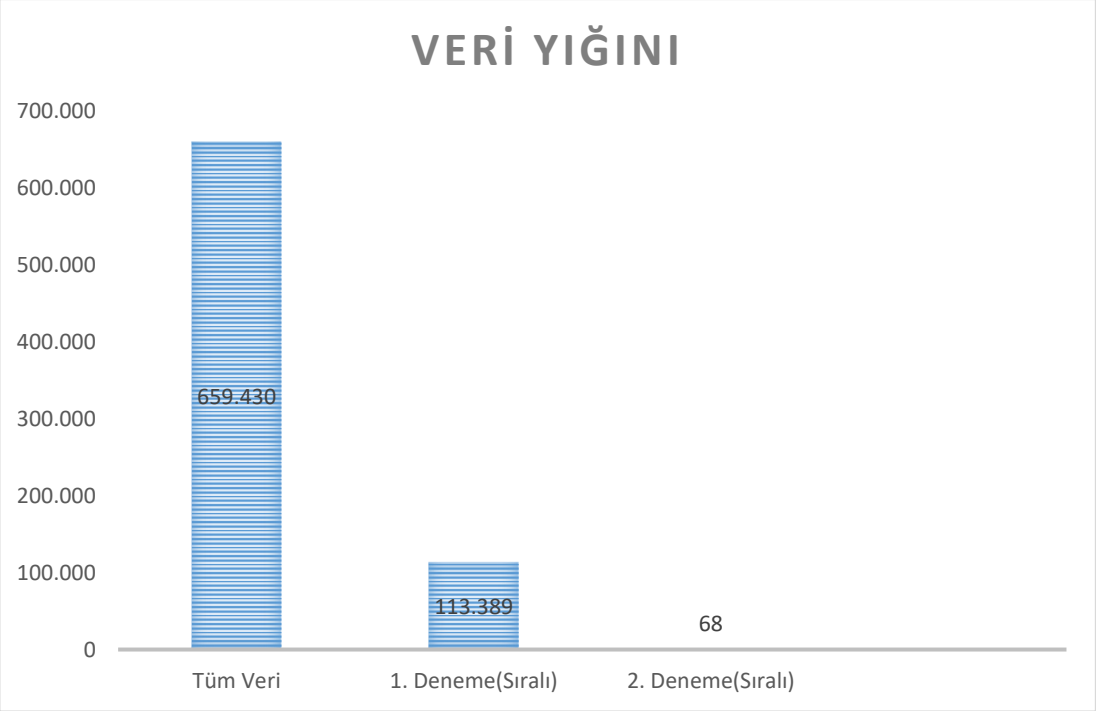


Şekil 5.13. Haritalama indirgeme çalışma grafikleri ikinci deneme.

Şekil 5.14. En Yüksek Önceliğe Sahip Değer.

BÖLÜM 6. SONUÇ

Sürekli artan veri yoğunluğu verilerin analizini çok önemli hale getirmiştir. Giderek artan maliyet ve zaman kısıtlılığı, haritalama-indirgeme aşamalarında oluşan maliyetlere çözümler bulmayı gerektirmiştir. Burada analiz yapılırken kullanılacak işlemlerin daha az maliyetli olması amaçlanmıştır. Bu amaçtan dolayı homojen veri yığınları filtrelenmesi ve az maliyetli verilere öncelik verilmiştir. Bu geliştirme ve üzerinde yapılan çalışmada analiz işlemi esnasında, maliyetin minimize edilmesi hedeflenmiştir. Uygulamada gerçekleştirilecek işlemlerin daha az maliyetli olması için homojen veri yığınlarının filtrelenmesi amaçlanmıştır. Şekil 6.1 fitreme sonuçlarının farkları görsel olarak sunulmuştur.



Şekil 6.1. Veri azalttım grafiği görseli.

Geliştirilen algoritma ve program uygulaması sonucunda ortaya çıkan verileri birleşiminde büyük veri analizinde haritalamaindirgeme işlevsel bir uygulamadır analizine varırız. Haritalama aşaması indirgeme aşamasını etkilemektedir. Bu sebepten dolayı haritalama aşaması sırasında indirgeme planlanmalıdır. Haritalama aşamasına başlamada sınır değerler var ise sınıra uymayan değerler dışlanmalıdır ve buradan alınacak örneklem sonuçlanma için yeterli ise örneklem sınırları belirlenmelidir. Haritalama aşaması sonrası hızlı sıralama ile verilerin içinden az kaynağa ve daha çok veriye ulaşmamızı sağlar. Bu sınırlara verilerin en uygun kapladığı alan miktarının sonucu bize getirir. Hızlı sıralama hem az alana ihtiyaç duyduğu hemde sıralamalarda veriyi n adet parçaya bölerek daha az zaman tüketimi, bellek tüketimi, işlemci tüketimi ve disk tüketimi gibi konularda maliyet azaltıcı etki gösterecektir. Büyük veride yaklaşım **sınırlama->haritalama->sıralama->değerlendirme->indirgeme** olarak seçilmeli ve geliştirilmelidir. Her çalışmada sıralama ve sınırlama gerekmeyebilir, bu gibi durumlarda çalışmaların verimliği için metotların uygun olarak dönüştürülmesi (Ezilmesi) kullanıcı verimini arttıracaktır. Verilerin tekrar etme sıklıkları önem düzeyini çıkarmada bir sınır önemi sağlayabilir ve bu sebepten dolayı verilerin işlenmesinde, tekrar sınırlaması dikkate alınarak düzenlenmelidir.

Haritalama öncesi aynı değerler ile analiz içinden çıkarılmış maliyet kazanımı elde edilmiş ve aynı zamanda homojenlik arttırılmıştır. Bu değerlerin indekslenerek maliyet azaltılmıştır. Tüm veri yığınında belli bir alan seçimine girilmesi veya girilmemesi durumuna bakılmaksızın az maliyetli ve tekrar frekansı yüksek olan veriler öncelikli hale gelmiştir. Böylece işlemler için indeksleme ile maliyeti oluşmuş fakat verilerin işleme esnasında oluşacak maliyet minimuma indirilmiştir. Sıralama algoritmalarında hızlı sıralama (böl yönet) algoritmaları verilerin sıralamasında zaman maliyetini azaltmıştır ve ayrıca hızlı sıralamada üç parçalı yapı kullanılarak daha fazla maliyet azatlımı sağlanmıştır. Maliyet kazanımı için ortalama tekrar frekansı ve bayt uzunluğu belirlenmiş bu değerler sıralamada kullanılmıştır. Bu analiz ile veriler yüzdelik puanlama ile değerlendirilmiştir. Değerli veriler listelemede öne alınmış ve böylece haritalama ve indirgeme çalışması ile veri analizinde gerekli kaynak ve

çalışma süresi azaltılmıştır. Bu çalışma ile uygulamada büyük verinin artan maliyeti düşürülmüştür.

Yapılan bu çalışma sonucunda elde edilen tecrübelere dayalı olarak, harita indirgeme öncesinde veriler en sağlıklı puanlama için homojen olmalıdır. Veriler en sağlıklı puanlama içinde homojen olmalıdır. Bu kullanıcının belli oranda kontrolünde bir seçenektir ve bu sebepten homojenliği artıracak yöntemler üzerinde çalışılmalıdır. Ayrıca farklı verilerin birbirlerini ait değerleri etkileyebileceği durumlar için. Verilerin çapraz puanlaması sağlanmalıdır. Bayt uzunluğu alanı içinde değersiz veri varsa azaltılan listeye eklenmeden önce değerlendirme dışına itilmelidir. Bu duruma örnek olarak, boşluk veya virgülden sonraki sayı basamak değerleri verilebilir. Son olarak sıralama algoritması çalışırken yoğun ram tüketimini azaltıcı donanımsal ve yazılımsal çözümler geliştirilmelidir.

KAYNAKLAR

[1] Dean J., Ghemawat S., Simplified Data Processing On Large Clusters, Communications of the ACM January, 2008.

[2] Yang H.C., Parker D.S., Simplified Indexing on Large Map-Reduce-Merge Clusters, DASFAA 2009: Database Systems for Advanced Applications pp 308-322, 2009.

[3] Zaharia M., Konwinski A., Joseph A.D., Katz R.H., Stoica I., Improving Map Reduce Performance in Heterogeneous Environments, Stoica - Osdi, 2008 - static.usenix.org

[4] Yang H., Dasdan A., Hsiao R.L., Parker D.S., Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters, SIGMOD '07: Proceedings of the 2007 ACM SIGMOD, International Conference on Management of Data June 2007 Pages 1029–104, 2007.

[5] Kolb L., Thor A., Rahm E., Multi-Pass Sorted Neighborhood Blocking with Map Reduce, Computer Science – Research and Development, Vol. 27, pp. 45–63, 2012.

[6] Verma A., Cho B., Zea N., Gupta I., Campbell R.H., Breaking the Map Reduce stage barrier, Published: 10 September 2011 Cluster Computing Vol. 16, pp.191–206, 2013.

[7] Goodrich M.T., Sitchinava N., Zhang Q., Sorting, Searching, and Simulation in the Map Reduce Framework, ISAAC 2011: Algorithms and Computation, pp 374-383, 2011.

[8] Herodotou H., Babu S., Profiling What-if Analysis, and Cost-based Optimization of Map Reduce Programs, in Proceedings of the VLDB Endowment 4(11):1111-1122 · August 2011, DOI: 10.14778/3402707.3402746

[9] Holmes A., Hadoop in Practice, ISBN 9781617290237; 536 pages, October 2012.

[10] Slagter K., Hsu C.H., Chung Y.C., Zhang D., An improved partitioning mechanism for optimizing massive data analysis using Map Reduce, The Journal of Supercomputing, Vol. 66, pages 539–555, 2013.

[11] Gopalani S., Arora R., Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means, International Journal of Computer Applications (0975 – 8887) Vol. 113 – No. 1, March 2015

[12] McCreadie R., Macdonald C., Ounis I., Map Reduce Indexing Strategies: Studying Scalability and Efficiency, Information Processing & Management Vol. 48, Issue 5, Pages 873-888, September 2012.

[13] Bakratsas, M., Basaras, P., Katsaros, D.,Tassiulas,L., Hadoop MapReduce Performance on SSDS for Analyzing Social Networks, Big Data Research, Vol. 11, pp.1-10, March 2018.

[14] Auradkar, P., Prashanth, T., Aralihailli, S., Kumar, S.P., Sitaram, D., Performance Tuning Analysis of Spatial Operations on Spatial Hadoop Cluster with SSD, Procedia Computer Science, V. 167, pp. 2253-2266, 2020.

[15] Lu, L., Feng, Q.Y., An Optimal Solution of Storing and Processing Small Image Files on Hadoop,Procedia Computer Science, Vol. 154, pp. 581-587, 2019.

[16] Babar, M., Arif, F., Jan, M.A., Tan, Z., Khan, F., Urban Data Management System: Towards Big Data Analytics for Internet of Things Based Smart Urban Environment Using Costimized Hadoop, Future Generation Computer Systems, Vol. 96, pp. 398-409, July 2019.

[17] Tallada, P., Carretero, J., Casals, J.,Acosta-Silva, C., Tonello, N., COSMOHUB: Interactive Exploration and Distribution of Astronomical Data on Hadoop, Astronomy and Computing, Vol. 32, Article 100391, July 2020.

[18] Ibrahim, S., Phan, T.D., Carpen-Amarie, A., Chhoub, H.E., Antoniu, G., Governing Energy Consumption in Hadoop Through CPU Frequency Scaling: An Analysis, Future Generation Computer Systems, Vol. 54, pp. 219-232, Jan. 2016.

[19] Rasooli, A., Down, D.G., COSHH: A Classification and Optimization Based Scheduler for Heterogeneous Hadoop Systems, Future Generation Computer Systems, Vol. 36, pp. 1-15, July 2014.

[20] Wu, W.T., Lin, W.W., Hsu,Ching-Hsien, H., LiGang, O., Energy-efficient Hadoop for Big Data Analytics and Computing: A Systematic Review and Research Insights, Future Generation Computer Systems, Vol. 86, pp. 1351-1367, September 2018.

[21] Pradeep, D., Sundar, C., QAOC: Novel Query Analysis and Ontology Based Clustering for Data Management in Hadoop, Future Generation Computer Systems, Vol. 108, pp. 849-860, July 2020.

[22] Mazumdar, S., Scionti, A., Chapter One: Fast Execution of RFD Queries Using Apache Hadoop, Advances in Computers, Vol. 119, pp. 1-33, 2020

[23] Usama, M., Liu, M., Chen, M., Hadoop environment: testing real-life schedulers using benchmark programs, Digital Communications and Networks, Vol. 3, Issue 4, pp. 260-273, November 2017

[24] Kusumakumari, V., Sherigar, D., Chandran,R., Patil,N.,Frequent Pattern Mining on Stream Data Using Hadoop Tree-GTree, Procedia Computer Science, Vol. 115, pp.226-273, 2017.

[25] Meena, K., Tayal, D.K., Castillo, O., Jain, A., Handling Data-skewness in Character Based String Simişlarity Join Using Hadoop, Applied Computing and Informatics, in pres, 16 Nov. 2018

[26] Bende, S., Shedge, R., Dealing with Small Files Problem in Hadoop tributed File System, Procedia Computer Science, Vol. 79, pp. 1001-1012, 2016.

[27] GOM-Hadoop: A Distributed Framework for Efficient Analytics on Ordered Datasets, Journal of Parallel and Distributed Computing, Vol. 83, pp. 58-69, Sept. 2015.

[28] Lin, H.K., Harding, J.A., Chen, C.I., A Hyperconnected Manufacturing Collaboration System Using The Semantic Web and Hadoop Ecosystem System, Procedia CIRP, Vol. 52, pp. 18-23, 2016.

[29] Lee, C.W., Hsieh, K.Y., Hsieh, S.Y., Hsiao, H.C., A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments , Big Data Research, Vol. 1, p. 14-22, Aug. 2014.

[30] Zhou, Y., Taneja, S., Dudeja, G., Qin, X., Alghamdi, M.I., Towards thermal-aware Hadoop clusters, Future Generation Computer Systems, Vol. 88, pp. 40-54, November 2018.

[31] Zhou, Y., Taneja, S., Dudeja, G., Qin, X., Zhang, J., Jiang, M., Alghamdi, M.I., Towards Thermal-aware Hadoop Clusters, Future Generation Computer Systems, Vol. 88, pp. 40-54, November 2018.

[32] Sowkuntla P., Prasad P.S.V.S, Mapreduce Based Improved Quick Reduct Algorithm With Granular Refinement Using Vertical Partitioning Scheme, Knowledge-Based Systems, Vol. 18915, Article 105104, February 2020

[33] Yao, X., Mokbel, M.F., Alarabi, L., Eldawy, A., Zhu, D., Spatial Coding-based Approach for Partitioning Big Spatial Data in Hadoop, Computers & Geosciences, Vol. 106, pp. 60-67, Sep. 2017.

[34] Roy, S., Gupta, S., Omkar, S.N., Case study on: Scalability of Preprocessing Procedure of Remote Sensing in Hadoop, Procedia Computer Science, Vol. 108, pp. 1672-1681, 2017.

[35] Sharma, N., Bagga, S., Girdhar, A., Novel Approach for Denoising Using Hadoop Image Processing Interface, Procedia Computer Science, Vol. 132, pp.1327-1350, 2018.

[36] Demchenko Y., Cees de L., Membrey P., Definingar Chitecture Components Of The Big Data Eco System. In International Conference On Collaboration Technologies And Systems (CTS) Pp. 104-112, 2014.

[37] Uzunkaya, C., Ensari T., Hadoop Ecosystem and its Analysis on Tweets, Procedia-Social and Behavioral Sciences, Vol. 1953, pp.1890-1897, July 2015.

[38] https://hadoop.apache.org.

[39] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, Erişim Tarihi: 15.12.2019.

[40] https://jar-download.com/, Erişim Tarihi: 15.12.2019.

[41] https://ckan.coegss.hlrs.de/, Erişim Tarihi: 15.12.2019.

ÖZGEÇMİŞ

Suat ERDOĞAN, 07.05.1987'de İstanbul Eyüp'te doğdu. İlk, orta ve lise eğitimini İstanbul'da tamamladı. 2005 yılında Profilo Anadolu Meslek Lisesi'nden mezun oldu. 2007 yılında başladığı İstanbul Kültür Üniversitesi Bilgisayar Teknolojisi ve Programlamadan mezun oldu. Sonrasında 2010 yılında başladığı Anadolu Üniversitesi İşletme Fakültesi'nden mezun oldu, ayrıca 2013 yılında başladığı Karabük Üniversitesi Bilgisayar Mühendisliği Bölümü'nü 2016 yılında bitirerek lisans eğitimini tamamladı. 2018 yılında başladığı yüksek lisans eğitimine Sakarya Üniversitesi Mühendislik Yönetimi Bölümü'nde devam etmektedir.