ÖHÜ Müh. Bilim. Derg. / OHU J. Eng. Sci.

ISSN: 2564-6605

Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi, Cilt 6, Sayı 2, (2017), 316-338
Omer Halisdemir University Journal of Engineering Sciences, Volume 6, Issue 2, (2017), 316-338

Derleme / Review

BULUT BİLİŞİM SİSTEMLERİNDE VERİNİN FARKLI BOYUTLARI ÜZERİNE DERLEME

Ahmet Şakir DOKUZ (ORCID ID: 0000-0002-1775-0954)[1*]
Mete ÇELİK (ORCID ID: 0000-0002-1488-1502)[2]

[1]*Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Niğde Ömer Halisdemir Üniversitesi, Niğde, Türkiye*
[2]*Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Erciyes Üniversitesi, Kayseri, Türkiye*

Geliş / Received: *10.06.2016*
Düzeltmelerin gelişi / Received in revised form: *20.03.2017*
Kabul / Accepted: *23.03.2017*

ÖZ

Bulut bilişim, son yıllarda gelişmekte olan ve güncelliğini koruyan bir teknolojidir. Bulut bilişim aracılığıyla kişisel bilgisayarlar üzerinde yapılan tüm işlemler, uzak sunucular aracılığıyla gerçekleştirilebilir hale gelmiştir. Bulut bilişim altyapıları kullanılarak sunucu sistemleri kurmadan büyük boyutlu depolama sistemleri kurulabilir ve yüksek seviyeli hesaplamalar yaptırılabilir. Bulut bilişim altyapıları üzerinde verinin farklı boyutlarda incelenmesi, bulut sistemleri üzerindeki veriye dayalı çalışmalarda fayda sağlayacaktır. Bu çalışmada bulut bilişim altyapıları üzerinde verinin farklı boyutları incelenerek bu boyutlarda ne tür stratejiler izlendiği incelenmiştir. Geçmiş çalışmalardan yola çıkılarak bulut bilişimde verinin ne tür gelişmeler izleyeceği öngörülmeye çalışılmıştır.

Anahtar Kelimeler: Bulut bilişim, bulut verisi, bulutta verinin boyutları

A SURVEY ON DIFFERENT ASPECTS OF DATA IN CLOUD COMPUTING SYSTEMS

ABSTRACT

Cloud computing is an emerging technology which is gaining popularity in recent years. With cloud computing, all computer-oriented operations can be done using remote cloud servers. Large scale storage systems can be established and high level computations can be made on cloud computing infrastructures without setting up server systems in-the-house. Lots of data centers store data for applications to process. Data play a main role for cloud computing applications which are generally developed to process large scale data. The research of different aspects of data in cloud computing infrastructures will provide information for future data-centric studies on cloud computing. In this survey, different aspects of data were analyzed and the strategies on every data aspect were researched. Also evolution of data in cloud computing was estimated based on recent developments.

Keywords: Cloud computing, cloud data, data aspects in cloud

1. GİRİŞ

Bulut bilişim, son yıllarda popülerlik kazanmış ve günümüzde gelişmekte olan bir teknolojidir [1, 2]. Yüksek hesaplama gücü veya depolama alanı gerektiren uygulamalar için uzak sunucularda bu alanları kullandığın-

[*]Corresponding author / Sorumlu yazar. Tel.:+90 388 225 2482; e-mail / e-posta: adokuz@ohu.edu.tr

kadar-öde sistemine dayalı olarak sağlamaktadır. Bulut bilişimin en önemli özelliği, sağladığı altyapıların ihtiyaca göre çok kısa süreler içerisinde ölçeklenebilir olmasıdır.

Günümüzdeki teknolojik gelişmeler sayesinde verinin her çeşidine rahatlıkla erişebiliyor ve bu verilerden anlam çıkarabiliyoruz. Klasik yöntemlerle kişisel altyapılar aracılığıyla verinin işlenmesi yaklaşımı bulut bilişim altyapılarının kullanılmasına doğru bir kayma göstermektedir. Bulut bilişim sistemleri tarafından sağlanan ucuz ve yüksek performanslı donanım kaynakları, klasik veri depolama ve işleme sistemlerinin terkedilmesi ve bulut bilişim sistemlerinin tercih edilmesini sağlamaktadır.

Bulut bilişim üzerinde verinin depolanması ve işlenmesi, veri sahipleri açısından büyük önem taşımaktadır. Depolanmaya değer bir veri var ise bu verinin işlenmesi ve anlamlandırılması da fayda sağlayacak demektir. Bulut bilişim üzerindeki verinin her yönüyle incelenmesi bulut bilişim altyapılarını kullanmayı planlayan kişiler için önem taşımaktadır. Bu çalışma, bulut bilişim sistemleri üzerinde verinin bulunduğu bütün katmanları sırasıyla incelemektedir.

Öncelikli olarak bulut bilişimin temelleri anlatılarak bir giriş yapılmış, ardından bulut üzerinde depolanan verilerin türleri ve çeşitleri araştırılmış, daha sonra bulut sistemleri içerisinde kullanılan dosya sistemlerine yer verilmiş ve ardından verinin çeşitli boyutları – depolama, sorgu ve veri yapısı – incelenerek verinin bu boyutlarında ne tür yaklaşımlar sergilendiği incelenmiştir.

Bulut bilişim sistemlerinde verinin bütün boyutlarıyla ele alınması, büyük veri uygulamaları başta olmak üzere veriyle ilişkili çalışmalara temel oluşturacaktır. Verinin hangi boyutlarıyla bulut bilişim sistemlerine dahil edilebildiği, veri üzerindeki manipülasyon stratejileri, veri yapılarının bulut sistemlerindeki uygulamaları gibi pek çok veri ilişkili konular, bu çalışma aracılığıyla ortaya çıkarılmıştır.

Bu çalışmada ikinci bölümde bulut bilişim temelleri anlatılmıştır. Üçüncü bölümde, bulut üzerinde tutulan veri türleri ve bulut verilerinin özellikleri incelenmiştir. Dördüncü bölüm, verilerin üzerinde bulunduğu dosya sistemleri ve özelliklerini içermektedir. Beşinci bölümde, bulut altyapıları üzerinde verinin depolanma stratejileri araştırılmıştır. Altıncı bölümde, bulut verisinin sorgulanmasındaki gereksinimleri ve sorgu stratejileri incelenmiştir. Yedinci bölümde, bulut üzerinde depolanan veriler için kullanılan veri yapıları sunulmuştur.

2. BULUT BİLİŞİM TEMELLERİ

Bulut bilişim için henüz fikir birliğine varılmış bir tanımlama yoktur. Geniş perspektiften bakıldığında, servis sağlayıcı veri merkezindeki donanım ve yazılım kaynaklarıdır [1]. Bulut bilişim, minimum yönetim eforları veya servis sağlayıcı etkileşimi ile hızlıca ücretlendirilebilen ve serbest bırakılabilen, ayarlanabilir hesaplama kaynaklarının (örneğin, ağlar, sunucular, depolama, uygulamalar ve servisler) paylaşımlı havuzuna yaygın, pratik ve anlık ağ erişimi sağlayan bir modeldir [2]. Bulut bilişim, servis sağlayıcı ve müşterisi arasında servis antlaşması kurulmuş olan, dinamik olarak ücretlendirilebilen ve bir veya birden fazla birleştirilmiş hesaplama kaynaklarına sahip birbirine bağlı ve sanallaştırılmış bilgisayarlar içeren paralel ve dağıtık hesaplama sistemidir [3].

Daha önce dağıtık ve paralel programlama modelleri ile aynı amaca hizmet etmekte ancak işlevsel olarak bazı farklılıkları bünyesinde barındırmaktadır. Gereken hesaplama gücünün uzak sunucular aracılığıyla sağlanması açısından aynı ortak paydaları bulunmasına karşın bu hesaplama gücünün temini konusunda bulut bilişim diğer modellerden ayrılmaktadır. Bulut bilişimin en önemli özelliği istendiğinde eklenip çıkarılabilen depolama, altyapı ve yazılımın sağlandığı esnek sunucu modeli olmasıdır.

Bu bölümde bulut bilişim konusunda temel bir tanıtım yapılmış, özellikleri, servis modelleri, bulut türleri verilmiş, bulut bilişimin avantaj ve dezavantajları tartışılmıştır.

2.1. Bulut Bilişim Özellikleri

Bulut bilişimin temel bazı karakteristik özellikleri bulunmaktadır. Hesaplama ve depolama için klasik sunucu sistemlerinden farklı yaklaşım sergilemektedir. Bu farklılıklar ve özellikleri [4];

- Sanallaştırılabilirlik
- İstendiğinde, kendi kendine kaynak ayırma
- Kullandığın kadar öde modeli
- Sanal depolama.
- Genel bulut ortamı arayüzleri.
- Sanal ağlar.
- Dinamik kaynak sağlama
- Sanal kümelemeler

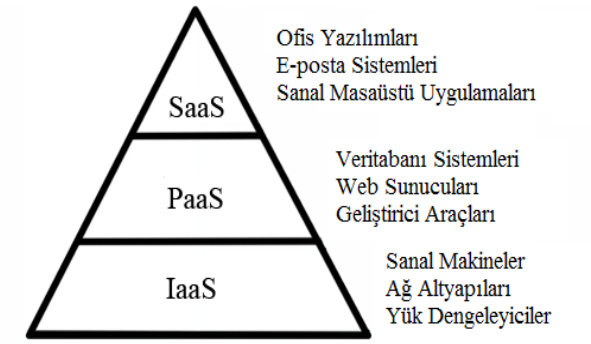
- Yüksek seviyede erişilebilirlik ve veri kurtarma

2.2. Bulut Bilişim Servis Modelleri

Bulut bilişim servis modelleri [2], bulut sağlayıcısının donanım ve yazılım kaynaklarından kullanıcının ihtiyacına göre yetkilendirilmesi ve erişiminin sağlanması için oluşturulmuş modellerdir. Hangi seviyede bir modele ihtiyaç duyulduğu tespit edildikten sonra bulut sağlayıcıları aracılığıyla amaca en uygun şekilde hizmet edecek olan servis modeli kiralanabilmektedir. Yazılımdan altyapıya geçişte yetki ve özgürlükler arttıkça ücretlendirme de buna paralel olarak artacaktır.

Sayısı konusunda farklı görüşler bulunsa da temel olarak üç tür servis modelinden bahsedilebilir [2]:

- **Servis olarak Yazılım (Software as a Service, SaaS):** Kullanıcıların uygulamalara erişmek için kendi sistemlerine herhangi bir kurulum yapmalarına gerek kalmadan, internet tarayıcıları aracılığıyla bulut bilişim sistemleri üzerindeki uygulamalara erişerek çalışma yapabilmeleridir. Müşteriler alt yapıdaki ağ, sunucu, işletim sistemi ve depolama aygıtları gibi bileşenleri yönetmez veya denetlemez. Sadece kullandıkları uygulamalara özgü ayarlamalar yapabilirler.
- **Servis olarak Platform (Platform as a Service, PaaS):** Servis sağlayıcı, müşteriye kendi uygulamasını geliştirip, çalıştırabileceği bir platform sunar. Bu platform uygulamanın geliştirileceği, çalıştırılacağı ortamla birlikte, tamamlayıcı servisleri ve gerekli teknolojik altyapıyı da kapsar. Kullanıcının kendi kurduğu uygulama dışında, platform altyapısını oluşturan bileşenler üzerinde herhangi bir kontrolü ve yönetim imkânı yoktur.
- **Servis olarak Altyapı (Infrastructure as a Service, IaaS):** Altyapının bir bulut servisi olarak sunulması modelinde müşteri ihtiyacı olan işlemci, depolama, ağ kaynağı ve diğer temel bilişim kaynaklarını kendisi yapılandırabilmekte ve bunların üzerine ihtiyacı olan işletim sistemi ve uygulamaları kurabilmektedir. Müşterinin ağ yapısı üzerinde yönetimi ve tam bir kontrolü olmamasına rağmen, işletim sistemi seviyesinde sisteme tam bir hâkimiyeti bulunmakta ve bazı ağ bileşenlerini (Firewall benzeri) yönetebilmektedir.



Şekil 1. Bulut bilişim servis modelleri

Şekil 1'de bulut bilişim servis modellerinin birbirlerine göre konumları ve örnekleri verilmiştir. Örneğin bir servis olarak platform modelinde, veritabanı sistemleri, web servisleri ve geliştirme araçları bulunurken bir servis olarak yazılım modelinde, ofis yazılımları, eposta sistemleri ve sanal masaüstü sistemleri bulunmaktadır. Piramidin en altındaki servis olarak altyapı, kendisinden daha yukarıdaki elemanlar için temel altyapıları sağlamaktadır. Piramidin tabanından tepesine doğru yaklaştıkça ilgilenilmesi gereken teknik detaylar azalmakta ancak bununla beraber özgürlükler de kısıtlanmaktadır. Servis olarak yazılım modeli tarafından sağlanan eposta sistemlerini kullanabilmek için servis olarak altyapı modelindeki ağ altyapısının sağlanması ve ardından servis olarak platform modelindeki web sunucularının ilgili sunucuya kurulmuş olması gerekmektedir. Ardından eposta yazılımına servis olarak yazılım aracılığıyla ulaşılabilir.

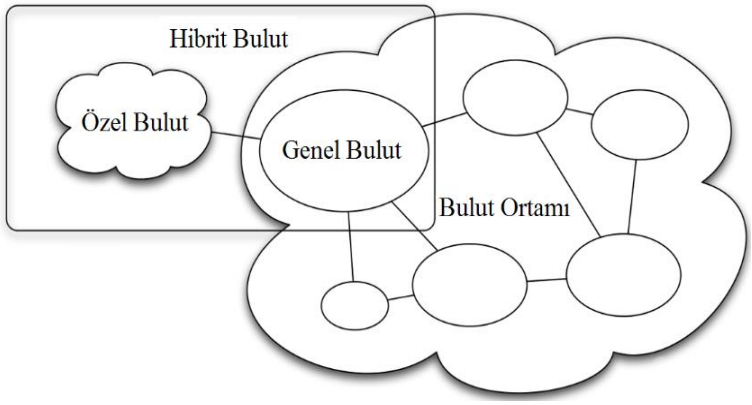
2.3. Bulut Bilişim Türleri

Bulut bilişim genel, özel ve hibrit bulut olmak üzere üç farklı tür altında incelenebilir [1, 2].

- **Genel Bulut (Public Cloud):** Genel bulutta servis sağlayıcı, uygulamalar ve depolama alanlarını internet üzerinden açık bir şekilde sağlar. Genel buluta en iyi örnek kullanıcılar tarafından yüklenen Youtube videolarıdır. Video kaynakları ve bilgileri bulut sağlayıcısı tarafından organize edilir ve internet erişimi olan herkes tarafından erişilebilirdir.

- **Özel Bulut (Private Cloud):** Özel bulutta servis sağlayıcı tarafından sağlanan depolama ve uygulamalar üzerine kullanıcının kendi güvenliğini sağlama yetkisi vardır. Bu sayede özel bilgiler ve veriler, bulut sağlayıcısı dâhil üçüncü şahıslar tarafından erişilemez. Özel buluta örnek Google Drive hizmeti olarak verilebilir. Kişisel dosyalar Google sunucuları tarafından depolanır ve sadece ilgili kullanıcı tarafından erişilebilir ve düzenlenebilir.
- **Hibrit Bulut (Hybrid Cloud):** Hibrit bulutta bulut sağlayıcısından kiralanan alanın tamamının güvenliği sağlanmaz, sadece kullanıcıya ait özel bilgilerin güvenliği sağlanır. Herkes tarafından erişilebilir alanlarda ise özel bilgiler saklanmaz. Örnek olarak Google Drive üzerinden paylaşıma açılan dosyalar verilebilir. Gizli kalması istenen dosyalar paylaşıma açılmayabilir ancak belirli dosyalar Google Drive üzerinden paylaşıma açılabilir.

Bulut bilişim türlerini daha iyi anlayabilmek bir web sitesi örnek olarak verilebilir. Bir web sitesinin giriş sayfası herkes tarafından görülebilir olmakla birlikte kullanıcılara ait bilgiler, giriş yapıldıktan sonra görülebilmektedir. Herkes tarafından görülebilen kısımları genel bulut, kullanıcı girişi yapıldıktan sonra görülebilen kısımları ise özel bulut olarak görülebilir. Eğer web sitesi bu iki türün birleşiminden oluşuyorsa da hibrit bulut yapısına bir örnek olur.



Şekil 2. Özel, genel ve hibrit bulut ortamları

Şekil 2'de özel, genel ve hibrit bulut ortamları gösterilmiştir. Özel bulut ortamı, bulut sistemlerine entegre olmayan ve kendi içerisinde hesaplama ve depolama altyapılarını barındıran bir bulut türüdür. Genel bulut ise hesaplama ve depolama altyapılarını bir bulut sağlayıcısından sağlayan bir bulut türüdür. Bazı özel durumlarda, hem özel hem genel bulut ortamları birlikte kullanılmak istenebilir. Bu yaklaşıma da hibrit bulut adı verilmektedir. Hibrit bulut, bir yandan özel bulut altyapısına sahip olurken, diğer yandan bazı hesaplama ve depolama ihtiyaçlarını genel bulut sağlayıcılarından temin edebilen bir ortam sunmaktadır.

2.4. Bulut Bilişim Avantajları

Bulut bilişim sistemleri, klasik sunucu sistemlerinden çeşitli noktalarda ayrılmakta ve daha büyük faydalar sağlamaktadır. Bulut bilişim sistemlerinin avantajları:

- Düşük Yazılım ve Donanım Maliyeti
- Gelişmiş Performans
- Anlık Güncelleme
- Sonsuz Depolama Kapasitesi
- Artırılmış Veri Güvenliği
- İşletim Sistemleri Arasında Geliştirilmiş Uyum
- Uyumlu Dosya Formatları

Bulut altyapısı tarafından donanım ve yazılımlar sağlandığı için bu sistemler için ayrıca herhangi bir ücret ödenmesi söz konusu olmamaktadır. Ayrıca profesyonel ekipler tarafından sağlanan yazılım ve donanım desteği ile daha performanslı sistemler elde etmek mümkün olmaktadır. Bunlara ek olarak, sunuculardaki veri güvenliğini profesyonel ekipler sağladığı için, kişisel sunuculardan daha güvenli bir ortam sunulmaktadır.

2.5. Bulut Bilişim Dezavantajları

Bulut bilişim sistemleri, uzak sunucularda çalıştıkları için çeşitli noktalarda dezavantajları bulunmaktadır. Bulut bilişim sistemlerinin dezavantajları:

- Sabit İnternet Bağlantısı
- Düşük Bağlantı Hızında Yavaş Çalışma
- Güvenlik Açıkları
- Sistem Güncellemeleri
- Deneyimsiz Bulut Operatörü

Bulut bilişim sistemlerinde internet üzerinden erişim sağlandığı için sürekli internet bağlantısı gerekmektedir. Bu nedenle düşük internet hızı ile erişim sağlamak uygulamaların yavaş çalışmasına sebep olur. Her ne kadar profesyonel ekipler tarafından güvenlik sağlansa da, küçük bir hata büyük zararlara yol açabilmektedir. Bulut altyapısı güncellendiğinde, bu altyapı üzerindeki uygulama düzgün çalışmayabilir.

3. BULUT BİLİŞİMDE VERİ TÜRLERİ

Günümüzde gelişen teknolojiler – uydu ve sensör teknolojisi, internet teknolojileri, mobil cihazlardaki gelişmeler – depolanan veri miktarını ve türünü çok büyük ölçüde artırmaktadır. Daha önce sadece metin olarak depolanabilen veri formatları günümüzde fotoğraf, ses veya video olarak da depolanmayı desteklemektedir. Ayrıca çeşitli alanlarda bilgisayar teknolojilerinin etkin kullanılmaya başlanması da veri boyutunu artıran bir etmendir. Örneğin coğrafya verisi harita mühendisliği için önemlidir. Belli bir bölgedeki tüm mekânsal parametrelerin depolanması ve erişiminin maliyeti fazladır. Yeni verilerin depolanması ihtiyacı ortaya çıktığı zaman, eğer eldeki alan yetersiz ise yeni donanım satın almak ve bu donanımın sunucuya entegre edilmesi gerekmektedir. Ancak bulut sistemlerinin sağladığı esnek depolama modeli ile dakikalar içerisinde yeni depolama alanları satın alınarak yeni verilerin kaydı kolayca gerçekleştirilebilir.

Bulut bilişim altyapıları, üzerinde her türden veriyi depolama ve işleme yeteneğine sahiptir. Klasik sunucularda, veriyi saklamak için sabit bir depolama alanı ayrılır ve eğer yeni veriler depolamak gerekiyor ise fazladan donanım satın almaya ihtiyaç duyulurdu. Bulut bilişim sistemleri sayesinde, yeni depolama alanları dakikalar içerisinde ayrılabilmekte ve kullanılabilmektedir.

Dijital olarak depolanan verilerin bazı karakteristik özellikleri bulunmaktadır. Bu özelliklerine göre dijital veri yapısal (structured), yapısal olmayan (unstructured) ve yarı-yapısal (semi-structured) veri olmak üzere üç kısma ayrılabilir. Bu bölümde yapısal, yapısal olmayan ve yarı-yapısal veri türleri anlatılmış, bulut verisinin karakteristik özellikleri verilmiş, bulut veri türleri ve mahrem ve gizli verilerin güvenliğinin sağlanması tartışılmıştır.

3.1. Yapısal Veri

Yapısal veri genellikle öntanımlı bir tür, boyut ve formata sahip olan veriyi ifade etmektedir [5]. Genellikle veri tabanında tutulan, belirli bir yapıya sahip olan veridir. Yapısal veri öncelikli olarak veri modeline dayanır. Yapısal verinin okunması, işlenmesi ve anlaşılması kolaydır.

Yapısal veri, bilinçli olarak verinin toplanması temeline dayanır. İki tür yapısal veri kaynağı bulunmaktadır; bunlar bilgisayar- veya makine-kaynaklı ve insan-kaynaklı olarak isimlendirilebilir. Makine-kaynaklı veri genellikle, insan müdahalesi olmadan makine veya bilgisayar tarafından üretilen veriyi ifade etmektedir. İnsan-kaynaklı veri ise bilgisayar aracılığıyla insanlar tarafından üretilen veriyi ifade eder.

Makine-kaynaklı yapısal veri türlerinin bazıları aşağıda listelenmiştir:

- Sensör verisi
- Web log verisi
- Alışveriş verisi
- Finans verisi
- …

İnsan-kaynaklı yapısal veri türlerinin bazıları aşağıda listelenmiştir.

- Girdi verisi
- Tıklama verisi
- Oyun-bağlantılı veri
- …

3.2. Yapısal Olmayan Veri

Özel bir tür, boyut veya formata sahip olmayan veriye yapısal olmayan veri denir [5]. Günümüz dünyasında saklanan verilerin büyük çoğunluğu yapısal olmayan verilerden oluşmaktadır. Bu nedenle yapısal olmayan verinin işlenmesi, içerdiği bilgi potansiyeli nedeniyle önem taşımaktadır. Son yıllara kadar yapısal olmayan veri, üzerinde durulmayan ve analizi için efor sarf edilmeyen bir veri türü olmuştu. Günümüzde büyük boyutlardaki verinin analizi ihtiyacı, yapısal veriden taviz verilmesini ve hızlı bir şekilde verinin analizini gerçekleştirebilmek için verinin yapısal olmayan bir şekilde depolanmasını gerektirmiştir. İçerdiği potansiyel bilgi kapasitesi ile analiz edilme sebebi olduğu için depolanması ve işlenmesi önem kazanmıştır.

Yapısal olmayan veride, yapısal veriyle aynı veri kaynakları bulunmaktadır; makine-kaynaklı ve insan kaynaklı.

Makine-kaynaklı yapısal olmayan veri türleri:

- Uydu resimleri
- Sismik, atmosferik ve yüksek enerjili fizik verisi
- Fotoğraf ve video verisi
- Radar veya sonar verisi
- …

İnsan-kaynaklı yapısal olmayan veri türleri:

- Özel metin verileri
- Sosyal medya verisi
- Mobil veri
- Web sitesi içerikleri
- …

Yapısal olmayan veri, yapısal verinin çok üstünde bir yaygınlığa sahiptir. Pek çok uzman, günümüzde depolanan verinin yaklaşık %80'ini yapısal olmayan verinin oluşturduğu ve yapısal veriden çok daha hızlı artış gösterdiği konusunda fikir birliğine varmıştır [5]. İçerdiği potansiyel bilgi zenginliğiyle yapısal olmayan verinin depolanması ve analiz edilmesi yapısal olmayan veriyi cazip hale getirmektedir.

3.2.1. Yarı-Yapısal Veri

Yarı-yapısal veri (semi-structured data), yapısal ve yapısal olmayan veri türlerinin aynı kayıt içerisinde birlikte kullanılmasıyla ortaya çıkan bir veri türüdür. Yarı-yapısal veride, belli parametreler yapısal veri özelliğinde, belli parametreler ise yapısal olmayan veri özelliğindedir. Kimlik numarası, IP adresi gibi zorunlu alanlar yapısal veri olarak depolanmakta; iş geçmişi, kişisel bilgiler gibi zorunlu olmayan alanlar yapısal olmayan veri olarak depolanmaktadır. Yarı-yapısal veriler, yapısal olmayan veriler içerisinde sayılabileceği gibi ayrıca da sınıflandırılabilir.

3.3. Bulut Verisinin Karakteristik Özellikleri

Bulut altyapısı üzerinde tutulan verinin genel bazı karakteristik özellikleri vardır. Karakteristik özelliklerine göre bulut verisi, geleneksel veriden verinin depolanması, boyutu, erişimi gibi konularda farklılıklar göstermektedir. Aşağıda bulut verisinin karakteristik özellikleri incelenmiştir.

1. Bulut üzerinde tutulan veri genellikle yapısal olmayan veriden oluşur.
2. Büyük boyutlu veya hızla artan eğilimdedir.
3. İnternet üzerinden erişilir.
4. Erişimi maliyet gerektirir.
5. Genellikle tek merkezde tutulmaz, farklı sunucularda dağıtılmış haldedir.
6. Erişimi esnasında güvenlik kontrolü gerekir.
7. Veri erişim hızı, internet bant genişliğine bağlıdır.
8. Verinin güvenliği bulut sağlayıcıları aracılığıyla sağlanır.
9. Sürekli olarak bulut sağlayıcısına, depolanan verinin boyutuyla orantılı olarak ödeme yapılması gerekir.
10. Bulut türüne bağlı olarak, veri gizli veya açık tutulur.

3.4. Mahrem ve Gizli Verilerin Güvenliğinin Sağlanması

Kullanıcılarının mahrem verilerinin güvenliğinin sağlanıp sağlanmayacağı endişesi, bulut bilişim sistemlerinin kullanılmasındaki çekincelerden biridir. Mahrem ve gizli verilerin güvenliğini sağlamak için çeşitli yaklaşımlar önerilmiştir. Bu yaklaşımlardan biri de verileri şifrelemektir. Ancak büyük boyutlardaki veriler söz konusu olduğu için tüm parametrelerin şifrelenmesi çok büyük yük getireceğinden dolayı, genel eğilim, sadece mahrem ve gizli verilerin şifrelenmesi yönündedir.

Şifrelenen verilerin analizi iki şekilde yapılabilmektedir. Birincisi; şifreli veri kullanıcı sisteminde çözülür ve analizi kullanıcı sisteminde yapılır. İkincisi ise veri şifreli halde iken bulut üzerinde analizinin yapılmasıdır. Birinci yaklaşım kullanıcı açısından hesaplama maliyeti getirmektedir. İkinci yaklaşım kullanıcı açısından daha avantajlı olarak görülebilir ancak analiz aşamasında şifreli verinin orijinal veriyi ne derece temsil ettiği önem taşımaktadır. Verinin şifrelenmesi ve şifreli verinin analizi, Bölüm 7'de daha detaylı olarak incelenmiştir.

4. BULUT BİLİŞİM DOSYA SİSTEMLERİ

Bulut bilişim sistemleri üzerinde kurulacak olan dosya sistemleri bulut sağlayıcılar için oldukça büyük öneme sahiptir. Dosya sistemleri, kullanıcılar tarafından oluşturulan, düzenlenen veya silinen verileri etkin bir şekilde bulut üzerinde depolamakta ve kullanıcılar için bu dosyalara erişim sağlamaktadır. Bulut üzerinde tutulan verilerin boyutlarının her geçen gün artması da dosya sistemlerinin etkin bir şekilde kullanılması ihtiyacını ortaya çıkarmaktadır.

Klasik veri sunucuları, tüm işlemleri kendi iç sistemleri içerisinde gerçekleştirdikleri için dosya sistemleri ikinci planda kalmakta idi. Ancak bulut sistemlerinde dosyalama sistemleri birincil öneme sahiptirler. Bulut kullanıcılarının, bulut üzerindeki depolama ihtiyaçları genellikle büyük boyutludur. Örneğin, bir sigorta şirketi verilerini bulut üzerine taşıdığı zaman ülke genelindeki tüm sigortalılarının bilgilerini bulut sistemine taşımış olmaktadır. Bu durumda da yüzbinlerce, belki milyonlarca, kayıt ve dosya bulut üzerine taşınmış olacaktır. Bu dosyalara erişim ve düzenlemeler etkin ve hızlı bir şekilde gerçekleştirilebilmelidir. Etkin erişim ve düzenleme imkânı için ise dağıtık işlemler yapmak gerekmektedir. Bulut sistemleri de dağıtık işlemler yapabilmek için dağıtık dosya sistemleri kullanmaktadır.

Bu bölümde bulut bilişim sistemleri için önemli olan dağıtık dosya sistemlerinin özellikleri tartışılmıştır. Aynı zamanda bulut dosya sistemlerinden üç örnek tanıtılarak kullanım şekilleri anlatılmıştır.

4.1. Dağıtık Dosya Sistemleri Özellikleri

Dağıtık dosya sistemleri, normal dosya sistemlerinden bazı konularda ayrılmaktadır. Aynı anda birden fazla kullanıcı tarafından erişilebilir olma ihtiyacı, uzak sunucularda tutulduğu için güvenlik açıklarına karşı stratejiler kullanma gerekliliği gibi bazı konular dağıtık dosya sistemlerine özgüdür. Bu başlık altında, saydamlık, dosya kopyalama, hata toleransı gibi dağıtık dosya sistemleri için geçerli olan özellikler incelenecektir [6].

Saydamlık: Dosya sistemleri, bir bilgisayar sistemi içerisinde, veri temelli çalışmalarda en yoğun olarak kullanılan sistemlerdir. Bu nedenle dosya sistemlerinin tasarımı, çeşitli saydamlık gereksinimlerini karşılar tarzda olmalıdır. Dosya sistemi, yazılım karmaşıklığı ve performansı sağlarken güvenlik, esneklik ve ölçeklenebilirliği de destekleyebilmelidir. Saydamlık çeşitleri incelenecek olunursa:

- Erişim Saydamlığı: Kullanıcı programlar, dosyaların dağıtımından haberdar olmamalıdır. Uzak ve yerel dosyalara olan erişim için temel işlemler seti sağlanır. Yerel dosyaları işlemek için çalışan programlar uzak dosyalar için de çalışabilmelidir.
- Mekân Saydamlığı: Kullanıcı programlar, aynı dosya uzayını görebilmelidir. Dosyalar veya dosya grupları dosya yolları değiştirilmeden yeniden konumlandırıldıklarında kullanıcı programlar nerede çalışırsa çalışsınlar, güncel dosya uzayını görebilmelidirler.
- Hareket Saydamlığı: Ne kullanıcı programları ne de sistem yönetici tabloları, dosyalar taşındığında değişmemelidir. Bu durum dosya hareketliliği sağlamaktadır.
- Performans Saydamlığı: Kullanıcı programları, servis yükü belli değerler arasında değişirken bile, kabul edilebilir bir şekilde işlem yapmalıdır.
- Ölçekleme Saydamlığı: Servis, artan iş yükü ve ağ boyutuna bağlı olarak artırımlı olarak genişletilebilmelidir.

Eşzamanlı Dosya Yükleme: Bir dosyaya bir kullanıcı tarafından yapılan değişiklik, aynı dosyaya başka bir kullanıcının başka bir değişiklik yapmasına engel olmamalıdır. Paylaşımlı veri erişiminde eşzamanlılık kontrolü

ihtiyacı pek çok uygulama tarafından kabul edilmiştir ancak bu işlem ek maliyet getirmektedir. Pek çok güncel dosya sistemi, zorunlu veya seçimli olarak dosya veya kayıt kilitleme mekanizması kullanmaktadır.

Dosya Kopyalama: Dosya kopyalamayı destekleyen bir dosya sisteminde, dosyanın farklı mekânlarda çeşitli kopyaları bulunur. Bu sayede i) aynı dosyaya erişim için çoklu sunucu ve yük paylaşımı sağlanır, ii) dosyanın kopyalarından birini barındıran sunucunun erişime kapanması durumunda dahi dosyayı erişime açık halde tutarak hata toleransı sağlanır. Ancak dosya kopyalamadaki problem, bir sunucudaki değişikliğin dosyayı barındıran diğer tüm sunuculara aksettirilmesi gerekliliği ve bu eşitleme işleminin ne sıklıkta neye göre yapılacağının belirlenmesidir. Eğer çok sık eşitleme yapılmak istenirse, sistem üzerindeki maliyet artacaktır. Benzer olarak eşitleme süresi artarsa, ilgili dosyaya erişimde eski sürümle karşılaşma riski ortaya çıkacaktır. Etkin bir dosya kopyalama stratejisi belirlenmesi bu nedenle önem taşımaktadır.

Heterojen Donanım ve İşletim Sistemi: Servis arayüzleri, farklı işletim sistemleri veya bilgisayarlarda problemsiz çalışabilmesi için iyi tanımlanmalıdır. Açık sistem yaklaşımı için bu kriter önem taşımaktadır.

Hata Toleransı: Dağıtık sistemlerde dosya servislerinin temel rolü kullanıcı veya sunucu hatalarında da servis çalışmasının devamını sağlamaktır. Dosya kopyalama ve ön belleğe alma yaygın olarak kullanılan hata toleransı stratejileridir.

Tutarlılık: Klasik dosya sistemleri dosyanın tek kopyası üzerinde işlem yapmaktadır. Ancak eğer dosyalar, hata toleransı veya iş yükünü azaltmak için farklı sunuculara kopyalanmışsa, kopyalar arasındaki güncelleme farkı tutarlılıktan taviz vermeyi gerektirmektedir. Etkin bir dosya kopyalama stratejisi olmaması halinde, dosya sisteminde tutarlılıktan söz edilemez.

Güvenlik: Tüm dosya sistemleri, erişim kontrol listeleri kullanarak erişim kontrol mekanizmaları sağlarlar. Dağıtık dosya sistemlerinde, sunucudaki erişim kontrolünün doğru kullanıcı kimliklerine bağlanması için ve isteğin içeriğini koruyarak ve dijital imzalarla mesajları cevaplamak için ve gizli veriyi şifrelemek için kullanıcı isteklerinin doğrulanması ihtiyacı vardır.

Verimlilik: Dağıtık dosya sistemi, en azından klasik dosya sistemlerinin sağladığı güçte ve yaygınlıkta yetenekler sağlamalıdır ve kabul edilebilir bir seviyede performans sergileyebilmelidir. Bir dağıtık dosya sistemi, klasik dosya sisteminden daha iyi performans ve verimlilik sergileyebilmelidir. Uzaktan erişim için gerekli şartlara iyi adapte olabilmelidir.

Metaveri: Önemli olan ancak mecburi olmayan bir diğer özellik ise dosyaların metaverileridir. Metaveri, dosyayı tanımlayan etiketleri barındıran bir üstbilgi olarak tanımlanabilir. Metaveri sayesinde dosyanın tamamına erişim yapılmadan dosya içeriği öğrenilebilir ve eğer gerekli görülürse dosyaya erişim sağlanabilir. Bu durum, dosya sisteminin iş yükünün azalmasını sağlayarak daha performanslı bir sistem sağlayabilir.

4.2. Bulut Dosya Sistemleri

Bu başlık altında bulut bilişim üzerinde yaygın olarak kullanılan bazı dosya sistemleri incelenerek özellikleri anlatılacaktır. Bu dosya sistemleri içerisinde en yaygın olarak kullanılan Google Dosya Sistemi (GFS) ve bu sistemin açık kaynak kodlu türdeşi Hadoop Dağıtık Dosya Sistemi (HDFS)'dir. Bunların yanında genellikle kullanılan bazı dosya sistemleri Amazon S3, Microsoft Azure, GlusterFS ve XtreemFS olarak sayılabilir. Bu çalışmada GFS, GlusterFS ve XteemFS dosya sistemleri incelenecektir.

4.2.1. Google Dosya Sistemi (GFS)

Google Dosya Sistemi (GFS) [7], Google tarafından üretilmiş olan bir dağıtık dosyalama sistemidir. Ucuz sunucularda hatalardan uzak bir şekilde çalışacak şekilde tasarlanmıştır. Temel çıkış noktası, hızla artan veri işleme ihtiyacına cevap verebilmektir. Google Dosya Sistemi, performans, ölçeklenebilirlik, güvenilirlik ve erişilebilirlik gibi daha önceki dağıtık dosya sistemlerinin özelliklerini içermektedir.

Google Dosya Sisteminde klasik dağıtık dosya sistemlerinden farklı olarak bazı ek özellikleri bulunmaktadır. Performans, ölçeklenebilirlik gibi özelliklerinin yanında, arızalara karşı sürekli görüntüleme, hata tespiti, hata toleransı ve otomatik kurtarma metotları da bulunmaktadır. İkincisi, geleneksel standartlara göre dosyalar büyük boyutlu olmaktadır. Ancak bazı durumlarda küçük boyutlu pek çok sayıda dosyanın veya büyük boyutlu bir dosyanın küçük boyutlu dosya parçalarına bölünerek depolanması ve işlenmesi ihtiyacı ortaya çıkabilir. Üçüncüsü, dosyalar için belli standart kullanım örüntüleri bulunmaktadır. Bu örüntülere göre dosya sistemi organize edilmeye çalışılmıştır. Son olarak da uygulama ve dosya sistemlerinin ortak tasarlanması verimlilik ve esnekliği artıracağından uygulama entegrasyonu sağlanmaya çalışılmıştır.

4.2.2. GlusterFS

GlusterFS [8], çok büyük ölçekli verilerin depolanması için geliştirilmiş açık kaynak kodlu dağıtık dosya sistemidir. Farklı disk ve hafıza kaynaklarını tek bir ortak isim uzayında birleştirmektedir. Dosyalarına erişmek isteyen uygulama ve kişiler bu ortak isim uzayı üzerinden erişebilmektedirler. Sunucular tuğla (brick) denilen alt parçalara bölünmüştür. Tuğlalar yardımıyla düşük seviyedeki dosya iletişimi sağlanmaktadır. Bir sunucuda birkaç tuğla bulunabilir. Sunucuların dağıtık bir şekilde bir araya gelmesiyle de Güvenilir Depolama Havuzu (Trusted Storage Pool – TSP) oluşturulur.

4.2.3. XtreemFS

XtreemFS [9], geniş alan altyapıları için kullanılan nesne tabanlı açık kaynak kodlu bir dosya sistemidir. Fiziksel olarak farklı sunuculara dosyaları kopyalayarak hata toleransı sağlamaya çalışmaktadır. XtreemFS mimarisinde, her biri bulunması gereken üç tür sunucu şekli vardır. Birincisi Dizin Servisi (Directory Service – DIR)'dir. DIR merkezi kayıt sistemini, konfigürasyon ayarlarını ve eşleşen dosyaların konumlarını içermektedir. Bu sistem Metaveri ve Kopyalama Katalogu (Metadata and Replication Catalog – MRC) tarafından uygun sunucuların belirlenmesinde kullanılır. MRC'de dizin ağacı ve metaveriler bulunmaktadır. Aynı zamanda dosya kimliklendirme ve yetkilendirmeden sorumludur. Üçüncü tür ise, Nesne Depolama Cihazı (Object Storage Device – OSD)'dır. OSD yazma ve okuma işlemleri için dosyaların nesnelerini ve arayüzlerini bulundurmaktan sorumludur.

Sistemin daha güvenilir olabilmesi için her üç sunucu türü de kopyalanabilmektedir. MRC ve DIR sunucularının kopyalanması demek dosya sisteminin daha güvenilir olması anlamına gelmektedir. Ayrıca, dosyalar üzerinde işlemler yapılmaya devam ederken dosyaların kopyalanması ve kopya dosyaların sunuculara iletilmesi de söz konusu olmaktadır. Bu durum da güvenilirliği artıran bir diğer husus olmaktadır.

5. BULUT BİLİŞİM VERİ DEPOLAMA SİSTEMLERİ

Bulut bilişim altyapıları üzerinde veri depolamada, güçlü bir sanallaştırma ve kullanıcının, verinin fiziksel konumu ve sunucusu ile ilişkisinin mümkün olduğu kadar azaltılması amaçlanmaktadır. Bu sayede teknik bilgi seviyesi düşük olan kullanıcıların hatalarına karşılık sistemin daha kararlı olması amaçlanmaktadır. Veri depolama için bulut sağlayıcısının, bulut bilişimde geçerli olan ölçeklenebilirlik, elastiklik, dinçlik ve çok-kullanıcılı erişimler için altyapı sağlaması gerekmektedir.

Etkili ve verimli bir şekilde kullanıcı verilerini bulut altyapıları üzerinde tutabilmek için çeşitli veri depolama yaklaşımlar geliştirilmiş, veri yapıları yardımı ile bu yaklaşımlar güçlendirilmeye çalışılmıştır. Bulut veri depolama alanında, klasik sunuculardan değişik sistem yaklaşımları geliştirmek zorunludur. Dağıtık bir ortamda olmasından dolayı, bulut sistemleri etkili ve doğru bir şekilde istenen sorguları işletmelerinde bazı zorluklarla karşılaşılabilmektedir.

Bu kısımda öncelikle bulutta veri depolamanın zorlukları anlatılmıştır. Ardından bulut üzerinde çalıştırılacak olan bir veri depolama sistemi için gereken şartlar incelenmiştir. Daha sonra veri depolama için geliştirilmiş olan yaklaşımlar incelenerek birbirlerine göre avantaj ve dezavantajları ortaya konulmuştur.

5.1. Bulut Veri Depolamanın Zorlukları

Bulut üzerinde depolanan verilere erişim için gerekli bazı şartlar vardır [10]:

- Verilerin tam ve doğru bir şekilde kullanıcılara iletilmesi,
- İstenen verilerin kısa süre içerisinde sağlanması,
- Kayıtlar arasındaki geçişlerin uzun zaman almaması,

en önemlileri olarak sayılabilir. Zaman yönetimi veri depolamadaki en büyük zorluklardan biridir. Zamandan tasarruf etmeye çalışırken, diğer parametrelerin de başarılı bir şekilde ele alınması başlı başına problem olmaktadır. Çünkü bulut üzerinde tutulan veriler dağıtık veritabanlarında tutulmakta ve veriler aynı fiziksel sunucuda dahi bulunmamaktadır. Bu durumda farklı sunucuların farklı işlem süreleri ve sunucuların arızalanma ihtimalleri de – ki bulut üzerinde kullanılan donanımlar göz önünde bulundurulduğu zaman arıza ihtimali her zaman yüksektir – dikkate alındığında problemin çözümü için etkili yöntemler önermek gerekmektedir.

Bulut üzerinde depolanan verinin güvenliğini sağlamak bir diğer zorluktur. Her ne kadar profesyonel bir ekip ile çalışılsa da, sisteme sızmak isteyen gruplar da profesyonel olarak saldırmaktadırlar. Bunun yanında bulut sağlayıcısının ne derece güvenilir olduğu sorusu da kullanıcıların kişisel bilgilerin paylaşmalarında bir engel

oluşturmaktadır. Bulut verilerinin güvenliğini sağlamak için etkili yöntemler geliştirmek ve bulut sağlayıcısının kullanıcı verilerine olan ilişkilerini şeffaf bir şekilde ortaya koyması zorunludur. Günümüzde bulut üzerinde paylaşılan veriler genellikle herkes tarafından erişilebilir olan verilerdir. Örneğin bir Youtube videosu bulut verisidir. Bu video kaynağına erişmek için Youtube sayfasına girmek yeterli olacağı gibi verinin gizliliği de bulunmamaktadır. Ancak kişisel bilgileri içeren bir bulut verisi, kimlik numarası, banka hesap numarası gibi, titizlikle korunması gereken sınıftadır.

Bulut üzerinde depolanan verinin sürekli bir şekilde erişimi de önem taşımaktadır. Bulut sunucusunun çok kısa sürelerle dahi olsa hizmet verememesi kritik uygulamalar için kabul edilemez olmaktadır. Bu nedenle verilerin sürekli erişilebilir olması gerekmektedir. Verilere erişim için sistemler arası kopyalama ve verilerin yedeklenmesi stratejileri önem taşımaktadır. Verilerin yedeklenmesi konusunda bazı hukuki problemler ortaya çıkabilmektedir. Bunlara örnek olarak bazı ülkeler, bankacılık işlemlerinin ülke dışına taşınmasına izin vermemektedir. Sunucuları dünya geneline dağılmış bir bulut sağlayıcısı düşünüldüğünde, yerelleştirme ve yerelde kalması gereken bilgilerin ayıklanması işlemi mecburi olmaktadır. Bu durumda hali hazırda zaten zor olan problem daha da zorlaşmaktadır.

Bulut veri depolama sistemleri için bazı özelliklerin olması mecburidir. Bunların başında ölçeklenebilirlik gelmektedir. Bulut üzerinde depolanan verilerin boyutu düşünüldüğünde, ölçeklenebilirlik daha da önemli hale gelmektedir. Ayrıca esnek bir veri depolama ve veri erişimi sistemi kurulması da gerekmektedir. Kurulan sistemin değişikliklere açık olması farklı alanlara hitap eden bulut uygulamaların gereksinimlerini sağlamak için bir gereklilik olmaktadır. Kurulan sistemlerin mümkün olduğunca kullanıcıdan soyutlanması, kullanıcının dosya sistemlerine doğrudan erişiminin engellenmesi, teknik bilgisi olmayan kişilerin oluşturacağı hataları önlemek için önem taşımaktadır. Aynı zamanda verinin tutulduğu sunucuların fiziksel konumlarının soyutlanması da farklı bölgelerdeki sunucuların farklı özellikler içerebileceği düşünüldüğünde önemlidir.

Bulut veri depolama sistemleri, yapmaları gereken dağıtık sorgular ve işlemler nedeniyle klasik veritabanı sistemlerinden farklı stratejiler geliştirmek durumundadırlar. Bu durumda bulut için geliştirilen bir veri depolama sistemi için de çeşitli ön koşullar bulunmaktadır. Bu ön koşulların sağlanması ve bulut üzerinde etkin bir şekilde sorguların işletilmesi de önemli olmaktadır. Etkin bir sorgu arayüzü ve bu arayüzün işletileceği ortamın geliştirilmesi de bulut veri depolamanın ilgilendiği alanlardan biridir.

5.2. Bulut Veri Depolama Sistemi Gereksinimleri

Bulut veri depolama sistemleri, veri gizliliği konusundaki güvensizlikleri giderebilmek için doğruluk, verimlilik, hata toleransı gibi çeşitli gereksinimleri [10] sağlamak durumundadır. Bu gereksinimlerin sağlanmaması halinde kullanıcılar bulut depolama sistemine karşı güvensiz yaklaşacak ve kullanmamayı seçeceklerdir. Bu nedenle aşağıda sayılan gereksinimler bir bulut veri depolama sisteminin göz önünde bulundurması ve çözümler üretmesi gereken etkenlerdir. Bu sayede güvenilir ve sağlam bir veri depolama sistemi kurulabilir.

Zaman Yönetimi: Zaman yönetimi bulut veri depolama sistemleri için kaçınılmaz bir gereksinimdir. Dosyaların ve kullanıcı verilerinin belli bir gecikmeyle kullanıcıya sunulması gerekmektedir. Gecikmenin artması kullanıcı memnuniyetini ve sistemin başarısını olumsuz etkileyecek bir faktördür. Bu nedenle geliştirilecek olan bir veri depolama sistemi için önde gelen unsurlardan biri kabul edilebilir bir süre içerisinde, talep edilen verilerin kullanıcının kullanımına sunulmasıdır.

Doğruluk: Zaman yönetimi kadar önemli bir diğer unsur da kullanıcı verilerinin doğru ve eksiksiz bir şekilde iletilmesidir. Eğer ki kullanıcının verileri eksik veya yanlış bir şekilde sunulursa sistem başarısız olarak etiketlenecektir. Bu nedenle bulut veri depolama sistemleri kullanıcı verilerini doğru ve eksiksiz şekilde kullanıcılara sunmak durumundadırlar. Bazı durumlarda verinin boyutu dikkate alındığında bu işlem kolay olmamaktadır. Verinin çok sayıda sunucuya dağılması durumunda doğru ve eksiksiz olarak verilerin temini zorlaşmaktadır. Özellikle hata toleransı sağlamak adına verinin kopyalarının çeşitli sunuculara dağıldığı sistemlerde eldeki verinin en güncel veri olduğunun tespiti çok zordur. Örneğin, bir kullanıcının daha önce değiştirmiş olduğu dosya diğer sunuculara kopyalanmadan kullanıcı bu sunuculardaki dosyayı açmaya çalışırsa dosyasının eski hali ile karşılaşacaktır. Eldeki verinin en güncel veri olduğunun doğrulanması için bulut depolama sistemleri çözümler üretmek durumundadırlar.

Verimlilik: Verimli bir bulut depolama sisteminden, istenen şartları sağlayacak şekilde sistemini kullanıcı için en uygun hale getirmesi beklenir. Kullanıcılar ücret ödedikleri bir sistemden maksimum fayda beklerler. Bulut sistemleri de depolama hizmeti sağlamasına karşılık olarak bu fayda garantisini verebilmelidir. Kullanıcıların faydası sürdüğü müddetçe bulut depolama sistemi verimlidir denilebilir. Bu nedenle verimlilik, üzerinde çok düşünülmesi ve optimizasyonların yapılması gereken alanlardan biridir.

Hata Toleransı: Hata toleransı da bulut depolama sistemleri için zorunlu bir özelliktir. Kullanıcılar tarafından yüklenen verilerin herhangi bir sunucu hatası nedeniyle silinmesi veya eski halinde kalması kabul edilemez. Bu

nedenle bulut sistemleri kendi yaklaşımlarını geliştirmek mecburiyetindedirler. En genel yaklaşım verinin farklı sunucularda kopyalarının tutulmasıdır. Bu sistem sayesinde verilerin güvenliği garanti altına alınmış olmaktadır. Ancak bu durumda da doğruluğun garanti edilmesi oldukça zorlaşmaktadır. Hatayı tolere etmeye çalışırken kullanıcı güvenini kaybetmek büyük risktir. Bu nedenle etkili yaklaşımlar geliştirerek hata toleransı ve zaman arasında optimizasyon yapmak, bulut veri depolama sistemlerinin gereksinimlerinden biridir.

Heterojen Çevrelerde Çalışabilme: Bulut donanım kaynakları çok büyük çeşitlilik içerebilir. Ucuz ve işlem gücü yüksek olmayan donanımlarla oluşturulmuş olan büyük bir bulut sunucu ağında her tür markadan her tür işletim sistemine sahip ve değişken cevap sürelerine sahip cihazların olması kaçınılmazdır. Bu nedenle kurulacak olan bir sistem için ideal olanı farklı donanım ve yazılım kaynakları üzerinde dahi aynı işlemi yaklaşık aynı sürelerde gerçekleştirme kapasitesine sahip olmasıdır.

Şifreli Verilerle Çalışabilme: Bazı kullanıcılar kişisel bilgilerini veya müşterilerinin mahrem verilerini bulut üzerine koymak istemezler. Bu durumda bulut veri depolama sistemlerinden beklenen bu isteğe cevap verecek şekilde çalışabilmeleridir. Şifreli verilerle çalışabilen bir bulut depolama sistemi müşterinin güveni konusunda tercih sebebi olabilmektedir. Bulut üzerinde depolanan şifreli verilerin de etkin bir şekilde erişimini sağlamak ve kullanıcının istediği işlemleri gerçekleştirmek bulut depolama sistemleri için bir gerekliliktir.

Farklı Uygulamalara Entegre Olabilme: Ticari olarak iş yapan pek çok işletme için önemli olan şey bilgilerine doğru ve hızlı bir şekilde erişebilmektir. Bunun yanında veritabanı arayüzleri gibi teknik olan sistemlerden mümkün olduğunca kaçınmak isterler. Bulut depolama sistemleri de kullanıcıların bu isteklerine cevap verecek şekilde tasarlanmalıdır. Bulut üzerinde tutulan verilerin kullanıcı dostu arayüzler aracılığıyla sorgulanması, görselleştirilmesi gibi işlemlerin gerçekleştirilmesi için altyapılarının olması bulut depolama sistemlerinden beklenen özelliklerden bir diğeridir.

Gizlilik ve Güvenlik Sağlama: Bazı bulut kullanıcıları için önemli olan bir diğer husus ise verilerinin gizliliğinin ve güvenliğinin sağlanmasıdır. Bazı önemli bilgileri tutan işletmeler bu bilgilerini bulut üzerine koymakta tereddüt yaşayabilmektedir. Bunun en önemli sebebi ise bulut üzerinde verilerinin kötü niyetli kişilerin eline geçeceği veya verilerinin sunucular üzerinden silinebileceği düşüncesidir. Kullanıcıların bu düşüncelerini ortadan kaldıracak şekilde bir sistem tasarımı geliştirmek ve gizliliğe ve güvenliğe önem veren bir sistem oluşturmak da bulut depolama sistemlerinin gerekliliklerinden biridir.

5.3. Bulut Üzerinde Kullanılan Veri Depolama Sistemleri

Günümüzde ticari olarak veri depolama hizmeti sağlayan çeşitli uygulamalar vardır. Bu başlık altında bu sistemleri ve birbirlerine göre avantajları incelenecektir. Temelde tüm veri depolama sistemleri bulut verisini depolayabilirler ancak bulut sistemlerinin dağıtık çalışması ve verinin farklı fiziksel sunucularda bulunması gibi sebeplerden dolayı klasik yöntemler pek çok bulut uygulaması için yetersiz kalmaktadır. Ek sistem yaklaşımları ile bazı depolama sistemleri diğerlerine göre daha avantajlıdır.

Bulut veritabanları ilişkisel ve NoSQL veritabanları olmak üzere iki kısma ayrılmaktadır.

5.3.1. İlişkisel Bulut Veritabanları

İlişkisel bulut veritabanları, klasik ilişkisel veritabanı modelinin bulut üzerindeki uyarlamasını kullanan sistemlerdir [11, 12]. Veriler ilişkili tablolarda tutulmaktadır ve öntanımlı bir şeması bulunmaktadır. İlişkisel veritabanı sistemlerinin en önemli dezavantajı ilişkilerin tanımlanması gerekliliğinden dolayı sorguların yavaş çalışmasıdır. Veri ile yoğun bir şekilde çalışmayan bulut sistemlerinde ilişkisel veritabanı sistemleri önerilebilir. İlişkisel bulut veritabanlarına örnek olarak; Oracle, IBM DB2, Microsoft SQL Server, PostregeSQL ve MySQL verilebilir.

İlişkisel veritabanı sistemleri, verimli bir şekilde çalışabilmek için veritabanı içerisindeki tüm alanların doğru ve eksiksiz bir şekilde doldurulmasına ihtiyaç duymaktadır. Bu nedenle yapısal veriler üzerinde etkili olduğu görülmektedir. Depoladığı verilerin format olarak doğruluğu da sistemin çalışmasını etkileyen bir faktördür. Örneğin; bir tarih bilgisinin hangi formatta girildiği ve doğru olup olmadığı önem taşımaktadır. Bunun yanında eksik veri olması veya format uyumsuzluğu olan veriler de sistemin çalışmasını negatif etkileyen faktörler olarak görülebilir. Getirdiği sınırlamaların yanında ilişkisel veritabanı sistemleri küçük ve orta yoğunlukta veri içeren uygulamalar için oldukça performanslı bir şekilde çalışabilmektedirler.

Günümüz bulut sistemlerinin ihtiyaçları göz önünde bulundurulduğu zaman, ilişkisel veritabanları hesaplama-yoğun (compute-intensive) uygulamalar için kullanılabilir olmakla birlikte, veri-yoğun (data-intensive) uygulamalar için doğru bir tercih olmamaktadır. İlişkisel tablolar ve arka plandaki indeksleme yapıları, sorguların yavaşlamasına sebep olmaktadır. Veri boyutunun artması da işlemlerin yavaşlamasının bir sebebidir. Bunların yanında sorguların kısıtlı bir şekilde paralelleştirilebilmesi de işlemlerin hızlandırılmasına yönelik bir diğer engeldir. Klasik ilişkisel veritabanı yaklaşımları ACID (atomiklik, tutarlılık, izolasyon, sağlamlık)

kurallarına [13] uymak durumundadırlar. Bu nedenle aynı sunucudaki verilerin dahi ACID kurallarına göre kontrol edilmesi gerekmektedir. Dağıtık yapıya sahip bulut sistemleri için ise bu kuralları sağlayabilmek büyük bir zorluğu da yanında getirmektedir. Bu nedenle hızlı ve etkili veri manipülasyon uygulamalarında ilişkisel bir veritabanı sistemi kullanmak işlem zamanını artırarak performansı düşürecektir.

5.3.2. NoSQL Bulut Veritabanları

NoSQL veritabanları, veri depolama işlemleri için klasik ilişkisel veritabanlarından farklı bir yaklaşım sunmaktadır. Anahtar-değer (key-value) ikilisi aracılığıyla dosya-tabanlı veri depolaması sağlamaktadır [14-16]. Bu sayede, dosya erişimi, dosya okuma ve yazma işlemleri oldukça hızlı bir şekilde gerçekleştirilebilmektedir. NoSQL veritabanları ham veriyi işleyerek kullanmak durumundadırlar. Ek işlemler fazladan maliyet getirmesine rağmen, bellek üzerinde çalışması ve yüksek derecede paralelleştirilebilmesi sebebiyle ilişkisel veritabanı yaklaşımlarından daha hızlı çalışabilmektedir.

NoSQL veritabanı sistemleri, ilişkisel veritabanı sistemlerinin getirdiği tüm sınırlamaları ortadan kaldırmaktadır. Tablo kullanmak gerekliliği, tablolar arası ilişkiler, verilerin formatlarının doğru olma zorunluluğu gibi kısıtlamaları ortadan kalkmasıyla daha sade ve daha kolay programlanabilir bir sistem ortaya çıkmaktadır. Dosyadaki her bir kayda ait anahtar bilgisinden ilgili kayda erişim sağlamaktadır. Her kayıtta aynı alanların bulunması zorunluluğu bulunmamaktadır. Farklı bilgilerin aynı dosyada tutulabilmesine de imkân tanınmaktadır.

Sayılan bu özellikleri yanında NoSQL sistemleri, veriler üzerinde analizler yapabilmek için ara işlemlere ihtiyaç duymaktadır. Bu durum da işlem maliyetini artırmakla birlikte, paralelleştirilebilmesi sayesinde bu eksikliğini giderebilmektedir. Verilerin temel dosya sistemleri üzerinde depolanması da ayrıca bir avantaj olarak görülebilir.

Veri-yoğun çalışan bulut sistemleri dikkate alındığında ve sosyal medya ağları gibi yapısal verinin bulunmasının çok zor olduğu uygulama alanları için NoSQL veritabanı sistemleri uygun bir çözüm haline gelmektedir. İndeksleme olmaması ve arama işlemlerinin anahtar değer üzerinden gerçekleştirilmesi yavaşlık ve dezavantaj olarak sayılabilir ancak paralel ve dağıtık işlem yapabilmesi sayesinde bu eksikliğini kapatabilmektedir.

NoSQL veritabanı sistemleri temelde anahtar-değer depolama yapısını kullanmaktadırlar. Gerek arama işlemleri için, gerekse de kayıtlar arası ilişkilerin belirlenmesinde anahtar-değer ikilisi yaygın bir şekilde kullanılmaktadır. Ancak ilişkilerin tutulabilmesi ve yönetilebilmesi için graf depolama yapısı gibi bazı farklılıklar içeren değişik yaklaşımlar geliştirilmiştir. NoSQL veritabanları, anahtar-değer, geniş sütun, doküman ve graf depolama olmak üzere dört farklı türde incelenebilir [16].

Anahtar-Değer Depolama: Anahtar-değer depolama yapısında, her bir kayda ait bir benzersiz anahtar bulunmaktadır ve ilgili kayda anahtar üzerinden erişilmektedir. Anahtar, benzersiz olmak zorundadır ancak değer için böyle bir zorunluluk bulunmamaktadır. Değer, basit metinlerden oluşabileceği gibi karmaşık listeler ve setlerden de oluşabilmektedir. Arama işlemi sadece anahtar üzerinden yapılabilir. Kullanıcı profillerini düzenleme, oturumları yönetme veya ürünleri sıralama işlemleri için idealdir. Anahtar-değer yapısının bir dezavantajı, değer kısmındaki herhangi bir içerik ile aramanın oldukça maliyetli olmasıdır. Anahtar-değer depolama yapısını kullanan veritabanlarına örnek olarak Dynamo [17, 18], Voldemort [19, 20] ve Redis [21] verilebilir.

Geniş Sütun Depolama: Bu depolama yapısında her bir kayda ait anahtar değerine değişken sayıda özelliklerin eklenebildiği dağıtık, sütun tabanlı bir veri yapısı bulunmaktadır. Her kayıt bir anahtar-değer ikilisi olarak görülebilir. Geniş sütun depolamanın avantajı, içerik sağlanamayan sütunların ilgili kayıtta hiç bulunmamasıdır. Değer girilmeyen bir alan için yer ayırma gereksinimi de bu sayede ortadan kalkmaktadır. Ayrıca geniş sütun depolamada süper sütunlar oluşturulabilmektedir. *Ad* ve *Soyad* bilgilerini tutan iki sütun, bir *İsim* süper sütununda birleştirilebilir. Bu birleştirme işlemi sadece mantıksal olarak gerçekleştirilmektedir. Bu sayede ortak ve benzer sütunlar bir araya getirilerek ortak bilgilerin analiz edilmesine destek verilmektedir. Geniş sütun depolama yapısını kullanan veritabanlarına örnek olarak Bigtable [22], Hypertable [23], Cassandra [24, 25] ve SimpleDB [26, 27] verilebilir.

Döküman Depolama: Döküman depolama yapısı dökümanları yönetmek ve depolamak için tasarlanmıştır. XML, JSON vb. standart veri formatlarından biriyle kodlanan verilerin her biri ayrı dökümanlarda tutulurlar. Değerler, döküman içerisinde yarı-yapısal bir şekilde özellik adı/değer ikilisi aracılığıyla tutulur. Tek bir sütun yüzlerce özelliği içerebilir ve özelliklerin sayısı ve türü kayıttan kayıta değişebilir. Doküman tabanlı depolamanın en önemli avantajı hem anahtar, hem değer aracılığıyla arama yapılabilmesidir. Döküman depolama yapısı kullanan veritabanlarına örnek olarak CouchDB [28, 29] ve MongoDB [30, 31] verilebilir.

Graf Depolama: Graf depolamada ilişkisel tablolar, bağlantılı anahtar-değer ikililerinden oluşan ilişkisel graflara dönüştürülür. Graf depolama yapısı, diğer üç yapıdan daha fazla görsellik içerir ve kullanıcı dostudur.

İlişkileri de göstermesi bakımından diğer üç NoSQL depolama yapısından ayrılır. Graf depolama yapısı, verilerin kendi iç özelliklerinden çok veriler arasındaki bağlantıların önemli olduğu uygulamalarda kullanılabilir. Graf depolama yapısını kullanan veritabanlarına örnek olarak Neo4j [32, 33], InfoGrid [34] ve InfiniteGraph [35] verilebilir.

5.4. Veri Manipülasyon Stratejileri

Eldeki büyük boyutlu bir verinin manipüle edilmesi ve düzenlenmesi, günümüz veritabanı sistemlerinin en büyük problemlerinden biridir. Klasik SQL sorguları ile büyük veri içeren problemlere çözüm üretilmeye çalışıldığında, sorgu darboğazına yol açılmaktadır. Aynı anda milyonlarca sorgu geldiğinde, tek bir veritabanının bu sorgulara cevap verebilmesi mümkün olamaz. Klasik SQL veritabanları yerine büyük veri ile daha iyi baş edebilecek, daha performanslı çalışacak yeni sorgu dilleri ve yeni veri manipülasyon stratejileri geliştirilmesi gerekmektedir.

Bulut altyapısı üzerinde veri manipülasyon stratejisinin seçimi, sistemin performansını etkileyen bir faktördür. Normal şartlarda, yapılan bir sorgunun döndüreceği kayıt sayısına bağlantılı olarak işlemin uzaması beklenir. Bulut sistemleri düşünüldüğünde, bir sorgu sonucunda milyonlarca kayıt döneceği için, uygun bir veri manipülasyon stratejisi ve sorgu dili olması bulut sistemleri için kaçınılmaz bir gerekliliktir.

Klasik ilişkisel veritabanlarında kullanılan veri manipülasyon dili SQL sorgulama dilidir. SQL sorgulama dili, yapısal ve orta boyutlu veriler üzerinde oldukça verimli bir şekilde çalışmaktadır. Temel SQL sorguları yardımıyla veritabanı içerisindeki kayıtlara erişilebilir, güncellenebilir, yeni kayıt eklenebilir ve mevcut kayıtlar silinebilir. SQL sorguları tüm alanları tanımlanmış, eksiksiz ve doğru olan veriler üzerinde performanslı bir şekilde sonuç üretebilirler.

SQL sorgularının en önemli dezavantajı kısıtlı bir paralelleştirme sunmasıdır. Sunulan bu paralelleştirmeyi kullanmak için oldukça düşük seviye bir SQL dili kullanmak gerekmektedir. Ayrıca her veritabanı sunucusunun sunduğu paralelleştirme yöntemi farklı olduğu için platformlar arası geçişler yapmak mümkün olmamaktadır.

SQL sorgularının bir diğer dezavantajı, çok fazla veri okuma-yazma-silme işlemleri yapılması durumunda, indeksleme yapısını düzenlemesi gerektiğinden dolayı yavaşlamasıdır. Bunun yanında büyük boyutlu veriler üzerinde bu işlemlerin yapılabilmesi başlı başına bir zorluk olmaktadır.

Bunların yanında SQL sorguları, yapısal olmayan veriler üzerinde verimli ve etkili bir şekilde çalışamazlar. Bir sütundaki metin içerisinden bir değerin aranması maliyeti de SQL sorgularının dezavantajıdır.

NoSQL veritabanlarında kullanılan sorgulama dili, temelde dosya üzerine kaydedilmiş olan veriye erişen ve veriyi manipüle eden bir dildir. Dosya boyutunda işlem yapıldığı için yüksek seviyede paralelleştirme imkanı sunmaktadır. Büyük boyutlu ve ilişkisel olarak zayıf olan veriler üzerinde verimli ve etkili bir şekilde kullanılabilmektedir. Sosyal medya verileri gibi çok hızlı artan boyutlardaki verilerin depolanması ve sorgulanması için en uygun seçenek NoSQL veritabanlarıdır.

NoSQL sorgu dili, okuma-yazma-silme işlemlerinin yoğun olduğu sistemlerde de efektif bir şekilde kullanılabilir. Temeldeki dosya yapısının sağladığı esneklik sayesinde kolaylıkla programlanabilir ve bakımı yapılabilir. Aynı zamanda paralelleştirilebilmesi sayesinde oldukça yüksek hızlarda veriye erişim sağlayabilmektedir. Verinin hızlı bir şekilde okunması ve yazılması gerekiyorsa ve gelen veri çok büyük boyutlarda ise, NoSQL sorgu dili ve NoSQL veritabanı kullanmak en uygun yaklaşım olacaktır.

6. BULUT BİLİŞİMDE VERİ SORGULAMA

Bulut üzerinde tutulan verilerin sorgulanması, verinin depolanmasından sonra üzerinde düşünülmesi gereken bir diğer gereksinimdir. İstenen verilere hızlı bir şekilde ulaşabilmek, veri üzerinde değişiklik yapılacaksa bu değişikliğin kolay bir şekilde yapılabilmesi gibi özelliklerin sağlanması gerekmektedir. Bir önceki bölümde bulut veritabanları incelenmiş ve NoSQL veritabanı sisteminin bulut üzerindeki veriler için daha etkili olduğu belirtilmiştir. Bu bölümde ise bulut veritabanları üzerinde veri sorgulama işleminin nasıl yapılabileceği anlatılmıştır.

İlişkisel veritabanı sistemlerinde veritabanı işlemlerinin yapılabilmesi için standartlaştırılmış bir sorgu dili (SQL) bulunmaktadır. SQL sorgu dili veritabanından bağımsız bir şekilde tüm ilişkisel veritabanları için geçerli seçme, güncelleme, silme veya birleştirme gibi komutlar sunmaktadır. SQL standart sorgu dili sayesinde kullanılan veritabanından bağımsız bir şekilde veritabanı sorguları oluşturulabilmektedir.

NoSQL veritabanı sistemlerinde herhangi bir standart sorgu dili bulunmamaktadır. NoSQL veritabanları, geliştirilen her uygulama için farklı yapılarda kullanılabildikleri ve yüksek oranda özelleştirilebilir oldukları için, tüm veritabanlarını ve tüm uygulamaları kapsayacak bir standartlaştırma oldukça zor olacaktır. İlişkisel veritabanlarındaki gibi sütun tabanlı bir yapısı olmadığı ve her bir kayda sadece benzersiz bir anahtar değeri ile

erişilebildiği için, NoSQL veritabanlarında seçme, güncelleme gibi standartlaşmış komutlar bulunmamakta ve çoklu düzenlemeler desteklenmemektedir. Bunun yerine, her uygulama için bir veri sorgusu geliştirilmesi gerekmektedir. Her uygulamanın yapısı birbirinden farklı olduğu için bu yaklaşım bir avantaj olarak görülebilir ancak diğer yandan bir NoSQL veritabanı için geliştirilen sorgu yapısının diğer bir veritabanında da geçerli olmaması dezavantaj olarak değerlendirilebilir.

Bu kısımda, öncelikle bulut üzerinde veri sorgusunun temel özelliklerine değinilmiş, ardından MapReduce yapısı ile bulut verisinin analiz edilmesi anlatılmış, Hadoop ortamı tanıtılmış ve son olarak da farklı tür bulut veritabanları için kullanılan veri sorgu dilleri incelenmiştir.

6.1. Bulut Veri Sorgusu

Bulut veritabanları için standart bir veri sorgulama dili bulunmamaktadır. Her bir veritabanı için geliştirilmiş olan sorgu dilleri ise farklı özelliklere sahiptir. Bu veritabanları üzerinde depolanan verinin verimli bir şekilde sorgulanması, bulut verilerinin erişimi, kullanımı ve düzenlenmesi için bir gerekliliktir. Standartlaşmış bir veri sorgulama dili bulunmasa da bulut üzerindeki verilerin sorgulanması için çeşitli ortak gereksinimler bulunmaktadır. Bu kısımda bulut sorgularının doğru, verimli ve hızlı bir şekilde çalışması için çeşitli gereksinimler sunularak açıklanmıştır.

Sorgu ölçeklenebilir olmalıdır. Bulut üzerinde çalıştırılan bir sorgunun, sonucuna göre ölçeklenebilir olması gerekmektedir. Veri boyutu ne kadar artarsa artsın, bulut sorgusu bu artışa karşılık iç mekanizmalar geliştirebilmelidir.

Sorgu paralelleştirilebilir olmalıdır. Bulut bilişim için önemli olan bir nokta paralelleştirmedir. Bulut üzerindeki herhangi bir işlemin paralelleştirilebilir olması, o işlemin oldukça yüksek hızlarla tamamlanabilmesine imkân sağlamaktadır. Bulut sorgularının paralelleştirilebilir olması, veriye hızlı erişim için ek özellik değil bir gereksinimdir.

Sorgu hatalara karşı toleranslı olmalıdır. Bulut bilişim altyapısı olarak kullanılan ekipmanlar, teknik özellikler bakımından gelişmiş olmaları gerekmediği için, bir iş parçacığının işlemini tamamlamadan bozulması veya servis dışı kalması olası bir durumdur. Bulut sorgusu, ekipmanların bozulması durumundaki stratejiyi iyi belirlemeli ve yedek iş parçacıkları kullanarak hataları tolere edebilmelidir.

Sorgu farklı fiziksel sunuculardaki verileri bir araya getirebilmelidir. Bulut üzerindeki veriler, genellikle farklı fiziksel sunucularda tutulmaktadır. Bu nedenle, bulut sorguları, farklı fiziksel sunucular üzerindeki verileri toparlayarak tek bir arayüzde sonuçları gösterebilmelidir.

Sorgu sonucunda verinin en son sürümü döndürülmelidir. Bulut üzerinde, güvenlik gerekçeleri ile verinin farklı sunucularda yedeklenmesi söz konusu olabilir. Sorgu işletildiği zaman veriye ait en son sürümün hangi sunucuda bulunduğu bilgisinin tutulabilmesi ve buna göre en son sürümün döndürülmesi gerekmektedir.

Sorgu özelleştirilebilir olmalıdır. Klasik sorgular gibi, bulut üzerindeki sorgular belirli anahtar kelimeler kullanılarak özelleştirilebilir olmalıdır. Bu sayede sorgu, ilişkisiz ve çok sayıda sonucu döndürmekten kurtarılmış olur.

Sorgu alt sorgulara bölünebilmelidir. Bulut sorgusunun hem paralelleştirme açısından hem de görevleri farklı iş parçacıklarına paylaştırma açısından alt sorgulara bölümlendirilebilmesi, sorgu işletim hızını artıracaktır.

Sorgu optimize edilebilmelidir. Yazılmış olan bir sorgunun en az maliyetli ve en hızlı şeklinin elde edilebilmesi için sorgunun optimizasyonu gerekmektedir. Bulut sorgularının da optimize etmeleri gereken sunucu lokasyonları, koşulların sıralanması gibi çok çeşitli parametreler bulunmaktadır.

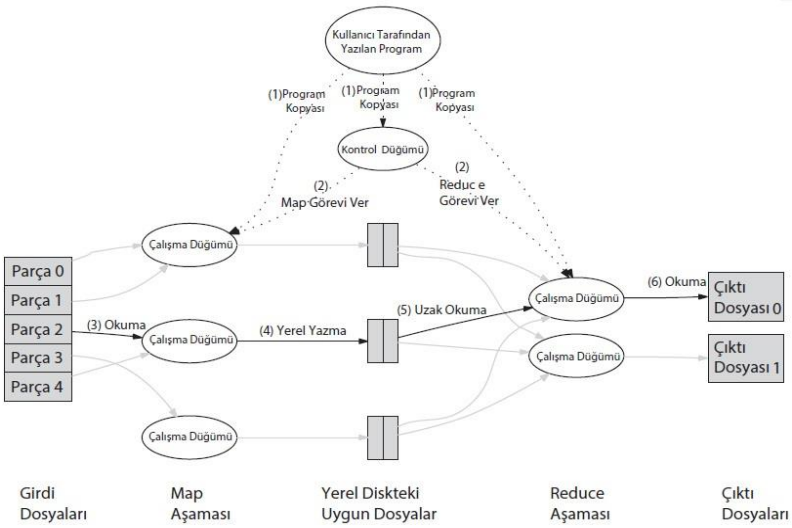
Sorgu sonuçları kabul edilebilir sürelerde döndürülmelidir. Bulut sorguları, yapacağı paralelleştirmeler ve kullanacağı optimizasyon stratejileri ile hızlı çalışabilmelidir. Kullanıcı memnuniyetini artırmak için sonuçların mümkün olduğu kadar kısa sürede döndürülmesi amaçlanmalıdır.

Sorgu bir metin veya dosya içerisinden etkili bir şekilde arama yapabilmelidir. Bulut veritabanları söz konusu olduğunda, farklı tür veri depolama stratejileri (anahtar-değer, doküman, vs.) bulunmaktadır. Bulut sorgularının, metin içerisinden belirli anahtar kelimelere göre belirli alanlarda arama yapabilmesi gerekmektedir. Aranan içeriğin elde edilmesi için de ek stratejiler geliştirilmesi zorunludur.

6.2. MapReduce Programlama Modeli

MapReduce, büyük veri kümeleri üzerinde veri-yoğun (data-intensive) görevleri yerine getirmek için geliştirilmiş olan bir programlama modelidir [36]. Başlangıçta Google tarafından web sayfalarının indekslenmesi için geliştirilmiştir. Anahtar-değer çifti (key-value pair) mantığıyla çalışmaktadır. Map ve Reduce fonksiyonları olmak üzere iki temel alt fonksiyonu bulunmaktadır. Map fonksiyonu <key, value> çiftlerinden <key, value> çift dizisi oluştururken, Reduce fonksiyonu <key, value> çift dizisinden <value> dizisi elde etmektedir.

Şekil 3'te görüldüğü üzere parçalanmış dosyanın her bir parçası Master tarafından bir Worker'a atanmakta ve **map** işlemini yapması beklenmektedir. Ardından orta katmandaki verilerin **reduce** işlemi yapılmak üzere Master tarafından yine Worker'lar görevlendirilmektedir.



Şekil 3. MapReduce çalışma mimarisi [36]

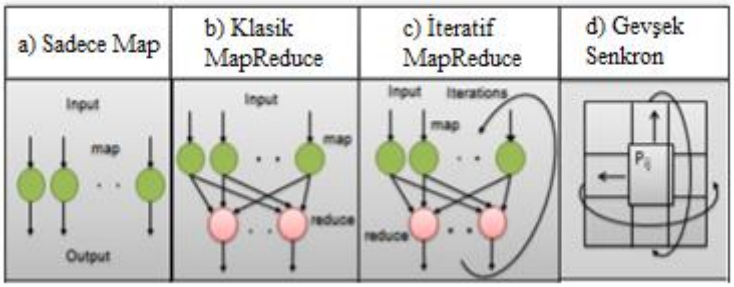
MapReduce için yapılan verimlilik hesaplamalarına göre, bu model oldukça verimli çalışmaktadır [36]. İşlem parçacıklarının pek çoğunun işlem dışı kalması durumunda dahi, nispeten daha uzun sürse de işlemi başarılı bir şekilde tamamlanabilmektedir.

MapReduce mimarisinde hata toleransı iki kısma ayrılmaktadır. Worker hatası ve Master hatası [36]. Worker'lara sürekli ping atılarak hata olup olmadığı ve Worker'ın halen sisteme dâhil olup olmadığı tespit edilmektedir. Master hatası için de görevler sürekli loglandığı için eğer Master hatası olursa sistem son kaldığı yerden çalışmasına devam edebilmektedir. Ancak MapReduce Worker için görev ataması yaptıktan sonra, sistemden kopup kopmadığını araştırmamaktadır. Ayrıca yazılımsal sorunlar nedeniyle Worker'lardan bir kısmı doğru sonuç döndürmeyebilirler. Bu durumları da araştıran bir hata toleransı yaklaşımı geliştirmek daha etkili sonuçlar alınmasını sağlayacaktır.

MapReduce iyi bir kontrol modeline sahiptir. Worker'ların bir kısmı ellerindeki görevi bitirmeye yaklaştıklarında, Master, devam eden görevler için yedek görevler oluşturur ve işlemini bitiren Worker'lara atar. Birincil veya yedek Worker, görevi bitirdiğinde ise görev tamamlanmış kabul edilir. Bu sayede, birkaç çok yavaş çalışan cihaz yüzünden tüm sistem yavaşlamamış ve işlem hızlı bir şekilde tamamlanmış olur.

MapReduce veri modelinde dışarıdan gelen girdi dosyası öncelikle parçalara ayrılmaktadır. Daha sonra parçalar <anahtar, değer> çifti haline dönüştürülerek işlemler devam ettirilmektedir. Veri modeli olarak daha etkili ve farklı yöntemler kullanılarak elde edilen sonuçların kalitesi artırılabilir.

MapReduce paralel işlem yapmakla birlikte tamamlanmamış görevleri beklemesi ve Reduce fonksiyonuna geçmek için Map fonskiyonlarının tüm Worker'larda tamamlanmasını beklemesi açısından eksiklikler içermektedir. Bu eksiklikleri giderebilmek için geliştirilen MapReduce versiyonları Şekil 4'te görülmektedir [37].



Şekil 4. MapReduce versiyonları [37]

MapReduce, bulut verisinin analizi için oldukça yaygın olarak tercih edilen bir yaklaşımdır. Sağladığı kolay programlanabilir fonksiyonları ve sade programlama çerçevesi sayesinde, hızlı bir şekilde verilerin analizi gerçekleştirilebilmektedir. Pig, Hive, JAQL, SCOPE, Sawzall ve Dryad MapReduce tabanlı veri sorgulama dillerine örnek olarak sayılabilir.

6.3. Apache Hadoop

Hadoop, sıradan sunucu kümeleri üzerinde büyük verileri güvenilir, ölçeklenebilir ve dağıtık bir şekilde işlemek için geliştirilmiş açık kaynak kodlu bir projedir [38, 39]. Proje içerisinde dört modül bulunmaktadır;

- **Hadoop Common:** Diğer modüller için ortak hizmetleri sağlar.
- **HDFS (Hadoop Distributed File System):** Uygulama verilerine hızlı erişim için kullanılan dağıtık dosyalama sistemidir.
- **Hadoop YARN:** Görev zamanlama ve kaynak yönetimi için kullanılan çerçevedir.
- **Hadoop MapReduce:** Büyük ölçekli veri kümeleri için YARN tabanlı paralel veri işleme sistemidir.

Bu modüller arasından HDFS ve MapReduce, asıl işlemleri yapan modüller olarak değerlendirilebilir. Common ve YARN diğer iki modül için altyapısal ve çevresel destek sağlamaktadır.

HDFS [40], Hadoop ortamında veri depolama işlemleri için kullanılan dağıtık dosyalama sistemidir. HDFS, çok büyük boyutlu (en az terabyte ve genellikle onlarca terabyte seviyesindeki) veriler için kullanılmaktadır. Bu büyüklükteki veri tek bir kaynakta tutulamayacağı için birbirinden uzak sunucularda paylaşımlı ve yedekli bir şekilde depolanan dağıtık dosya sistemine sahiptir.

HDFS, en önemli veri analiz örüntüsünün bir-defa-yaz, çoklu-oku (write-once, read-many-times) yapısına sahip olduğu düşüncesine göre tasarlanmıştır. Bir veri kümesi, kaynakta bir defa oluşturulur ve birçok defalar erişilerek okunur. Bu şekilde daha geniş bir gruba hitap etmek hedeflenmiştir.

HDFS, pahalı ve yüksek güvenilirlikte donanımlar gerektirmemektedir. Nod bozulmalarının ve erişimlerin kopmasının yaygın olduğu sıradan sunucular kümesi üzerinde çalışmak üzere tasarlanmıştır. HDFS dosya sisteminde hata toleransı sağlamak için çeşitli yaklaşımlar geliştirilmiştir.

Hadoop, Java programı ile geliştirilmiştir ve Hadoop MapReduce işlemi, HDFS üzerinde daha önce anlatıldığı şekilde gerçekleştirilmektedir. Hadoop, *Map* ve *Reduce* görev parçacıklarını kümelerdeki nodlara dağıtarak, sonuçların elde edilmesinden sorumludur. Hadoop, *Map* ve *Reduce* işlemlerinin ikisini de aynı nod içerisinde gerçekleştirebildiği için ağ trafiği gecikmeleri engellenmiş olur ve aynı zamanda yüksek ölçüde ölçeklenebilirlik sağlanır.

6.4. Bulut Veri Sorgulama Dilleri

Bulut altyapısı üzerinde çeşitli türden veritabanları bulunduğu gibi, her veritabanı için de farklı bir veri sorgulama dili bulunmaktadır. Sorgulama dillerinin bir kısmı SQL benzeri bir yaklaşım sergilerken, bir kısmının da bilgisayar programı benzeri yapıları bulunmaktadır. Bu kısımda, örnek olarak Pig, Hive ve JAQL bulut veri sorgulama dilleri incelenmiştir.

6.4.1. Pig

Pig, MapReduce işlemlerini SQL sorguları benzeri bir script dili aracılığıyla gerçekleştirmeyi amaçlayan yüksek seviyeli bir veri akış sistemidir [41]. Pig dilinde sorgu yazmak, sorgu işletim planı oluşturmaya benzer. Esnekliği ve kolaylığı sayesinde tercih edilebilir bir yaklaşım sunmaktadır.

Pig Latin'de, kullanıcı bir adımlar zinciri tasarlar. Bu tasarımda, her adımda sadece bir yüksek seviyede veri dönüşümü gerçekleştirebilir. Bu yaklaşım sayesinde karmaşık SQL sorguları yerine bir akış sistemi elde edilmiş olur. Ayrıca, eğer ara işlemlere de ihtiyaç duyuluyor ise, rahatlıkla ara adımlar tasarlanabilir.

Pig program modelinde, standart veritabanı modelindeki tabloların kullanılması zorunluluğu yoktur. Herhangi bir verinin, klasik veritabanlarına kaydedilebilmesi için en azından 1. Normal Formda (1NF) bulunması gerekmektedir ancak Pig için böyle bir zorunluluk bulunmamaktadır. Bu sayede, içiçe, karmaşık ve atomik olmayan veri modellerini de desteklemektedir. İçiçe veri modeli kullanılarak insan düşünce yapısına daha yakın bir yaklaşım ortaya konulabilmektedir. Verinin dosyalarda tutulduğu durumda, veritabanı yaklaşım ile veriyi önce 1NF'ye çevirip daha sonra birleştirme işlemleriyle birleştirmek oldukça maliyetli olacaktır. Ancak Pig doğrudan dosyadaki veriyi kullanabilmektedir.

Tablo 1'de yüksek pagerank değerine sahip olan url'lerin ortalamasını ve kategorisini çekme işlemini gerçekleştiren Pig sorgusu verilmiştir [41].

Tablo 1. Pig sorgusu [41]

```
good_urls = FILTER urls BY pagerank > 0.2;
groups = GROUP good_urls BY category;
big_groups = FILTER groups BY COUNT(good_urls)>10^6;
output = FOREACH big_groups GENERATE
category, AVG(good_urls.pagerank);
```

Örnekte de görülebileceği gibi Pig dili adımlar aracılığıyla SQL sorgusuyla tek satırda yapılabilecek işlemi beş ayrı satırda yapmaktadır. Bu sayede her bir adımdaki sonuçlar görülebileceği gibi, paralelleştirme konusunda da oldukça iyi bir programlama ortamı oluşmaktadır.

6.4.2. Hive

Hive, MapReduce programlama modelinin düşük seviye olması ve kodların tekrar kullanılabilirliğinin olmaması üzerine SQL benzeri bir yaklaşımla Hadoop ortamı üzerine geliştirilmiş bir açık kaynak kodlu veri sorgulama yaklaşımıdır [42]. HiveQL, Hive veritabanında MapReduce scriptlerini de sorgu içerisinde kullanılabilir olarak sağlamaktadır.

Hive içerisinde veriler tablolar, bölümler ve birimler şeklinde bulunurlar. Her bir tablonun karşılığı olan HDFS yolu vardır. Tablolardaki veriler, bu yol içerisinde dosyalar halinde saklanır. Kullanıcılar kendi veri formatlarını da verebilmektedirler. Bölümler ise tablolardaki verilerin parçalanmasında kullanılırlar. Bir bölüm oluşturulduğu zaman, tablonun bulunduğu yol içerisine yeni bir yol oluşturulur. Birim ise, bölümlerdeki verilerin hash fonksiyonları aracılığıyla tekrar bölümlendirilmesinde kullanılırlar. Hive temel veri tiplerini desteklediği gibi kullanıcıların yeni tipler oluşturmasını da desteklemektedir.

HiveQL, select, project, join, aggretate, union gibi SQL sorgularını desteklemektedir. Kullanıcılar dış kaynaklardan verilerini Hive tablolarına yükleyebilir ve sonuçlarını sorgulayabilirler. Aynı zamanda HiveQL çoklu kayıt işlemini de desteklemektedir. Bu sayede, kullanıcılar aynı girdi ile pek çok sorgu aracılığıyla kayıt yapabilmektedirler.

HiveQL içerisine kullanıcı tanımlı fonksiyonlar ve birleştirme fonksiyonları tanımlanabilmektedir. Ek olarak kullanıcılar herhangi bir programlama dilinde yazılmış olan MapReduce fonksiyonlarını satır tabanlı olarak entegre edebilmektedirler.

Tablo 2. HiveQL sorgusu [42]

```
FROM (
FROM records2
MAP year, temperature, quality
USING 'is_good_quality.py'
AS year, temperature) map_output
REDUCE year, temperature
USING 'max_temperature_reduce.py'
AS year, temperature;
```

Tablo 2'deki sorguda, öncelikle yıllara göre sıcaklıklar ve kalite değerleri çıkarılmaktadır. Daha sonra ise en yüksek sıcaklığa sahip yıl ve sıcaklık değeri getirilmektedir. Bu örnekte iki adet kullanıcı tanımlı fonksiyon kullanılmıştır; *is_good_quality.py* ve *max_temperature_reduce.py*.

6.4.3. JAQL

JAQL, JSON modelinden esinlenilmiş olan ve ilişkisel tablo verilerinden yarı-yapısal verilere kadar geniş bir aralıktaki verileri analiz etmekte kullanılan bir script dilidir [43]. JAQL dört temel özelliğe sahiptir; i) esnek veri modeli, ii) tekrar kullanılabilirlik, iii) değişken boyutlardaki soyutlama mekanizmaları ve iv) ölçeklenebilirlik. JAQL'in geliştirilme amaçlarından birisi de düşük seviyeli MapReduce işlemlerinden geliştiricileri kurtarmak ve geliştirme aşamalarını kolaylaştırmaktır.

JQAL, veri modeli olarak yarı-yapısal ve yapısal verilere hizmet verecek bir yaklaşım sergilemektedir. JAQL veri modeli değer parametresi bir atom, dizi veya kayıt olabilmektedir. Atom olarak pek çok temel tipi desteklemektedir. Diziler ve kayıtlar ise iç içe birleştirilebilirler. Bir dizi değerlerin sıralı birleşmesinden

meydana gelmektedir ve vektör, liste veya set veri yapılarına modellenebilir. Bir kayıt, sırasız isim-değer ikililerinden oluşmaktadır ve struct, sözlük veya harita modellenebilmektedir.

Basit tasarımına karşın, veri modeli oldukça esnek bir şekilde kullanılabilmektedir. Bu veri modeli, değişken ve birbirinden farklı gösterimlere sahip veriler üzerinde JAQL'in çalışmasını sağlamaktadır. Yarı-yapısal verinin depolanması ve sorgulanmasında, diziler ve kayıtlar kullanılır. Aynı zamanda JSON, farklı programlama dilleri arasındaki bir veri aktarımı aracı olması nedeniyle, JAQL aracılığıyla gerçekleştirilen işlemler kolaylıkla herhangi bir programlama diline aktarılabilmektedir.

Tablo 3. JAQL sorgusu [43]

```
1. import myrecord;
2.
3. countFields = fn(records) (
4. records
5. -> transform myrecord::names($)
6. -> expand
7. -> group by fName = $ as occurrences
8. into f name: fName, num: count(occurrences) g
9. );
11. read(hdfs("docs.dat"))
12. -> countFields()
13. -> write(hdfs("fields.dat"));
```

Bir JAQL scripti, çeşitli adımların birleşiminden oluşmaktadır. Bir script adımında ya girdi, ya atama, ya da işlem gerçekleştirilmektedir. JAQL içerisinde fonksiyonlar gelişmiş bir programlama dilindeki kullanımına benzer. Parametre olarak verilebilir, herhangi bir değere atanabilir veya değer döndürebilir. Aynı zamanda JAQL, herhangi bir programlama dilinde yazılmış olan kullanıcı tanımlı fonksiyonları ve birleştirme fonksiyonlarını da desteklemektedir.

Tablo 3'teki sorguda, öncelikli olarak 11-13. adımlar işletilmektedir. 12. adımdaki *countFields()* fonksiyon çağrısı ile 3. adıma geçilmektedir. Fonksiyon içerisinde dışarıdan 1. adımda import edilmiş olan *myrecords* değişkeni üzerinden kullanıcı tanımlı fonksiyonlara erişim sağlanmaktadır. *countFields()* fonksiyonu, *docs* dosyasındaki alanları ve bu alanların yaygınlıklarını hesaplamakta ve *fields* dosyasına yazmaktadır.

7. BULUT BİLİŞİMDE VERİ YAPISI STRATEJİLERİ

Bulut bilişim sistemlerinde veri yapıları kurulması, bulut üzerinde tutulan verilerin genellikle yapısal olmaması sebebiyle oldukça maliyetlidir. Bu nedenle klasik indeksleme ve ağaç yapıları gibi sistemler bulut üzerinde mevcut durumlarıyla kullanılamazlar. Bulut sistemlerinde veri yapıları kurulması ve veri yapılarının gerçekleştirilebilmesi için yeni yaklaşımlar önerilmesi gerekmektedir.

Bu kısımda bulut üzerinde kullanılan veri yapısı stratejileri veri kopyalama, erişim ve indeksleme, veri güvenliği ve şifreleme ve veri sıkıştırma stratejileri olmak üzere üç ana başlıkta incelenmiştir. Ayrıca bulut üzerinde bu veri yapılarının uygulanabilmesi için ne tür gereksinimler olduğu ortaya konulmuştur.

7.1. Veri Kopyalama, Erişim ve İndeksleme Stratejileri

Veri kopyalama sistemleri, bulut üzerinde tutulan verilerin ulaşılabilirliğini sağlayan sistemlerdir. Literatürde çeşitli veri kopyalama yaklaşımları önerilmiş ve performansları değerlendirilmiştir [44-46]. Doğru veri kopyalama stratejisinin kullanılması, kullanıcıların memnuniyeti ve verilere erişimi açısından önemlidir. Veri kopyalama stratejileri üç temel kritere göre incelenebilir:

1. **Hangi verinin ne zaman kopyalanacağının belirlenmesi.** Kullanıcıların sistemde bulunma durumlarına göre bekleme zamanını azaltmak, doğruluğu ve veri erişim hızını artırmak için hangi verinin ne zaman kopyalanacağının belirlenmesi gerekmektedir.
2. **Kaç yeni kopya oluşturulacağının belirlenmesi.** Her yeni kopya ile sistem her zaman hızlanmaz, aksine bazı durumlarda kopyalar arası geçişler ve son sürümünün takibi yavaşlamalara sebep olabilir.
3. **Yeni kopyaların nerede bulundurulacağının belirlenmesi.** Farklı bölgelerdeki sunuculara düzgün dağıtılmış kopyalar ve istek setleri bulunuyorsa tüm kopyaların aktif olması sistemi hızlandırabilir. Ancak

karmaşık sistemlerde hangi kopyaların aktif hangi kopyaların yedek olacağı, üzerinde düşünülmesi gereken bir durumdur.

Veri kopyalama yaklaşımı olarak genellikle farklı bölgelerde ve fiziksel sunucularda bulunan 3 adet kopya tercih edilmektedir. Farklı bölgelerdeki kullanıcılara, daha hızlı ve dağıtık bir hizmet sunması yönüyle bu yaklaşım yaygın olarak kullanılmaktadır. Veri kopyalama stratejileri içerisinde, verinin güncellenmesi durumunda diğer kopyalara bu güncelleme işleminin nasıl ve ne zaman yansıyacağı soruları cevaplanmalıdır. Büyük boyutlu bir güncelleme işlemi yapılması durumunda ve güncelleme işlemi esnasında verinin diğer sunuculardaki kopyalarına erişmeye çalışan bir diğer kullanıcı için ne tür bir yaklaşım sergileneceği iyi planlanmalıdır.

Veri erişimi, bulut bilişimin ayırt edici özelliğidir. Bulut bilişim, sürekli olarak veri erişimini garanti etmesinden dolayı, altyapısını ve erişim stratejilerini buna uygun olarak oluşturmak durumundadır [47]. Veri içerisinden belirli anahtar kelimelere göre arama yapılması, veri erişimi içerisinde önemli bir noktayı oluşturmaktadır. Arama yapabilmek için ise çeşitli indeksleme stratejileri geliştirilmesi gerekmektedir.

Bulut verisinin erişimi, kullanıcının veri kopyalarından birisine yönlendirilmesiyle gerçekleştirilebilir. Bu aşamada, kullanıcının en hızlı cevap alabileceği bölgedeki veri kopyası seçilerek işlemin hızlandırılması sağlanabilir. Bulut verisinin erişiminde bir diğer önemli durum, veri içerisinde arama yapılması ve arama işleminin hızlandırılması için indeksleme yapısının sağlanmasıdır.

Bulut verisinin yapısal olmaması nedeniyle, bulut verisi içerisinde arama işlemi, klasik arama metotlarıyla gerçekleştirilemez. Verinin hangi sütunda aranacağı bilgisi ve verinin tam olarak eşleşmemesi durumlarında dahi etkili ve verimli bir arama işlemini büyük boyutlu veri içerisinden kısa sürelerde gerçekleştirmesi beklenmektedir. Bunu sağlayacak sistem yaklaşımları da geliştirilmiştir [48].

Bulut verisinin aranması esnasında aranan anahtar sözcükler tüm satırdaki sütunlarda ve benzerlikleri de içerecek şekilde kullanıcıya sunulmalıdır. Örneğin, bir imla hatası olması veya bir harfin yanlış yazılması durumunda dahi anahtar sözcüğe yakın sonuçlar kullanıcıya gösterilebilmelidir. Bu sayede arama işlemi sonucunda çok daha etkili ve verimli bir şekilde kullanıcının ilgilendiği içeriklere erişmesine imkân sağlanacaktır. Örneğin; bir kullanıcı "ev satın almak" anahtar sözcükleriyle arama yaptığı zaman "satılık daire" başlıklı içeriklerin de döndürülmesi beklenir. Klasik arama işlemlerinde birebir eşleşmeler döndürülmektedir ancak bulut arama sistemleri büyük boyutlu veriyle ilgilendiği için anahtar sözcüğe yakın sonuçları da döndürebilmelidir.

Bulut verisi üzerinde arama yapabilmek için indeks yapısının iyi tanımlanmış olması gerekmektedir. İndeks yapısı aracılığıyla sorguların kalitesi ve farklı sütunlardaki verileri bir arada arayabilme yeteneği geliştirilebilmektedir. Klasik sistemlerde kullanılan ağaç yapıları bulut sistemlerine uyarlanarak indeksleme sistemlerinde kullanılabilmektedir. Bu amaçla B-tree [49], B^+-tree [50], R-tree [51, 52] ve KD-tree [53, 54] ağaç yapıları kullanılmış ve bulut üzerinde uygulanmıştır.

7.2. Veri Güvenliği ve Şifreleme Stratejileri

Bulut verisinin güvenliğinin sağlanması, bulut üzerindeki uygulamaları kullanan kullanıcılar için öncelikli bir gereksinimdir [55-58]. Güvenlik gerekçelerinin, kullanıcıların bulut altyapılarını tercih etmemelerinin bir sebebi olarak öne sürülmesi de bunun bir göstergesidir. Bulut kullanıcıları, güvenlik nedenleriyle mahremiyet, şeffaflık ve erişilebilirlik özelliklerinin sağlanmasını isterler. Bunları sağlayabilmek için minimum olarak şifreleme şeması, erişim kontrolü ve zamanlanmış yedeklemeler yapılması sağlanmalıdır.

Veri güvenliğinin sağlanması iki ayaklı olarak düşünülebilir. Birinci ayağı bulut sağlayıcısının kullanıcının verilerine erişimdeki güvenlik kontrolü oluşturmaktadır. Bu ayak bulut sağlayıcısının kontrolü ve sorumluluğundadır. İkinci ayağı ise kullanıcı tarafından bulut üzerine konulan verisinin şifrelenmesi ve gizliliğinin sağlanması oluşturmaktadır. Bu ayakta ise bulut kullanıcısı, ya bulut sağlayıcısının sağladığı şifreleme altyapılarını kullanır veya kendi şifreleme stratejisini oluşturabilir.

Bulut sağlayıcısı, kullanıcılarının güvenini sağlayabilmek için kullanıcı verileri üzerinde yapılan tüm işlemleri şeffaf bir şekilde kullanıcının bilgisine sunmalıdır. Hangi kullanıcının hangi veri üzerinde değişiklik yaptığı, yetkisiz giriş olup olmadığı, kritik veriler üzerinde güncellemeler yapılıp yapılmadığı gibi bilgilerin bulut kullanıcıları tarafından bilinmesi sistemin güvenilirliğini artırır ve güvenlik açıklarını kapatmak adına gereklidir. Aynı zamanda bulut sağlayıcısı bu bilgileri sağlayarak kullanıcının verileri üzerindeki şeffaflığı da sağlamış olmaktadır.

Verilerin güvenliği sağlanırken, aynı zamanda erişimin de engellenmemesi gerekmektedir. Bulut kullanıcısı kendi verilerine, istediği her zaman rahatlıkla ve engelsiz bir şekilde erişebilmesi sağlanmalıdır. Bunu sağlamak için de yetkilendirmeler kullanılarak hangi verilere hangi kullanıcıların erişebileceği kontrolü ile bulut sistemi daha güvenli hale getirilebilir.

Veri şifreleme, genel bulut altyapıları üzerinde hassas ve önemli bilgilerin depolanmasında tercih edilen bir yaklaşımdır [59]. Kredi kartı, banka hesap bilgileri, tıbbi bilgiler veya kişisel özel bilgiler gibi veriler genel bulut altyapıları üzerinde saklanırken güvenilmeyen kişilere karşı açık bir şekilde bulunduğu için bu verilerin bir şekilde gizlenmesi veya şifrelenerek saklanması gerekmektedir. Bu amaçla veri şifreleme stratejileri genel bulut altyapıları üzerinde gizlilik gerektiren durumlarda kullanılmaktadır.

Genel bulut ortamı, verilerin erişilebilirliğini ve güvenilirliğini artırması açısından fayda sağlamaktadır. Bunun yanında güvenlik ve gizlilik riskleri ortaya çıkmaktadır. Veri şifreleme teknikleri de kullanıcılara ait verilerin güvenliği ve gizliliğini sağlamak için kullanılmaktadır. Diğer bir deyişle genel bulut ortamı içerisinde sanal bir özel depolama servisi sağlamak olarak düşünülebilir. Böyle bir servis, hem genel bulutun fonksiyonelliği ve düşük maliyetini, hem de özel bulutun güvenliğini sağlayabilecektir.

Bir veri şifreleme yaklaşımı, mahremiyet (confidentiality) ve şeffaflığı (integrity) sağlamalıdır. Mahremiyet; bulut sağlayıcısının, kullanıcı verileri konusunda herhangi bir bilgisinin olmamasıdır. Şeffaflık ise; bulut sağlayıcısının, kullanıcı verisi üzerindeki herhangi bir yetkisiz değişikliğinin kullanıcıya bildirilmesidir.

Mahremiyet ve şeffaflık yanında bir diğer özellik erişimdir. Bulut kullanıcısı kendi verisine doğrudan ve şifresiz bir şekilde erişebilmesi gerekmektedir. Ancak bulut sağlayıcısı ve diğer bulut kullanıcıların, şifreli verilerin içeriğini görememeleri gerekir. Bunun için de kullanıcı tarafından sunuculara yüklenen verinin sistem tarafından şifrelenmesi ve buluttan verinin kullanıcıya aktarımı esnasında şifresinin çözülmesi gerekmektedir. Bu durum, fazladan işlem yüküne sebep olsa da sistemin güvenliğinin sağlanması için gereklidir.

Bulut verisinin şifrelemesinde bir diğer önemli nokta, şifreli veri içerisinde aramanın nasıl yapılacağıdır. Bu konuda da çeşitli çalışmalar yapılmış ve farklı yaklaşımlar önerilmiştir [60-62]. İndeks kullanılarak, verinin indeks üzerinden çıkarımı yapılabilmektedir. Bu sayede sorguyu yazan kişi ve veri sahibi gerçek veriyi bilebilmekte ancak bulut sağlayıcı verinin sadece şifrelenmiş halini tuttuğu için bilgi sahibi olamamaktadır.

7.3. Veri Sıkıştırma Stratejileri

Veri sıkıştırma, verilerin taşınması ve depolanması aşamalarında daha az yer kaplaması için tercih edilen bir veri yapısı yaklaşımıdır [63-65]. Bulut depolama alanları, neredeyse sınırsız bir şekilde kaynak tahsis etmesinden dolayı, verinin depolanmasında sıkıştırılmasına ihtiyaç duyulmamaktadır. Ancak verinin ağ üzerinde sunucudan istemciye taşınması veya verinin kopyalarının bir sunucudan diğer bir sunucuya taşınması esnasında verinin sıkıştırılması, sistemi önemli bir girdi/çıktı maliyetinden kurtarmaktadır. Bu nedenle bulut sistemlerinde veri sıkıştırma yöntemleri, sıklıkla olmamakla beraber çok büyük boyutlu verilerin sürekli olarak ağ üzerinde taşındığı durumlarda kullanılmaktadır.

Veri sıkıştırma için kullanılan klasik algoritmalar bulut altyapılarında doğrudan kullanılamazlar. Bu nedenle bulut altyapılarına uygun bir şekilde bir uçta sıkıştırma yapacak ve diğer uçta sıkıştırılmış veriyi açacak bir sistem kurulması gerekmektedir. Bunun yanında bulut üzerinde veri sıkıştırma işlemini gerçekleştirebilmek için bazı temel bilgilerin de bilinmesi gerekmektedir.

Bulut üzerinde veri sıkıştırma yapabilmek için öncelikli olarak verinin türü bilinmelidir. Metin verileri oldukça yüksek bir performansla sıkıştırılabilirken, multimedya, video ve fotoğraf verilerini sıkıştırmak bazı durumlarda veri kayıplarına ve verilerin bozulmasına yol açabilmektedir. Bulut verilerinin de çoğu durumda yapısal olmadığı ve heterojen yapısı düşünüldüğünde etkin bir veri sıkıştırma stratejisinin kullanılması, işlemin başarıyla sonuçlanması açısından önemlidir.

Bir diğer önemli nokta, veri sıkıştırma işlemi için gereken hesaplama maliyetinin göz önünde bulundurulmasıdır. Büyük boyutlu verinin sıkıştırılması için gereken hesaplama gücü, verinin erişilebilirliğini düşürecek derecelerde olmamalı, ancak bunun yanında mümkün olduğu kadar kısa süreler içerisinde tamamlanabilmelidir.

Bellek yönetimi de veri sıkıştırma stratejilerinde göz önünde bulundurulmalıdır. Büyük ölçekli verilerin dağıtık bir şekilde sıkıştırılması esnasında, genellikle büyük boyutlarda bellek ihtiyacı ortaya çıkmaktadır. Bellek kullanımının sadece veri sıkıştırma işlemine ayrılmadan, diğer işlemlerle ortak hareket edebilecek şekilde bir veri sıkıştırma stratejisi belirlenmelidir.

Girdi/çıktı maliyetleri de bulutta veri sıkıştırma işlemi esnasında dikkate alınmalıdır. Sıkıştırılacak olan verinin bir noktada toplanıp toplanmayacağı veya dağıtıldığı sunuculardan hangi sırayla toplanacağı bilgileri ve verinin sıkıştırma sonucunda nereye depolanacağının da dikkatli bir şekilde planlanması gerekmektedir.

8. SONUÇLAR

Bulut bilişim son yılların popüler hesaplama ve depolama yaklaşımlarından biridir. Hesaplamaları ve veri depolama sistemlerini uzak sunucular aracılığıyla ve kullanıcıların kendi sistemleri içerisinde yerine getirmeleri

gereken bakım, sistem tasarımı, yedekleme gibi pek çok işlemi profesyonel ekipler kullanarak sağlaması, kullanıcıların daha profesyonel hizmet almaları açısından önem taşımaktadır.

Bulut bilişim altyapıları üzerinde verinin yönetimi ve işlenmesi ise, artan bulut bilişim talebi nedeniyle bir gereksinim halini almıştır. Uzak sunuculardaki verilerin, yine uzak sunuculardaki hesaplama üniteleri tarafından işlenmesi ve kullanıcıya doğru ve hassas bir şekilde sonuçların iletilmesi, bulut bilişim açısından üzerinde düşünülmesi ve çözüm üretilmesi gereken konulardan biridir. Bu nedenle bulut altyapıları üzerinde veriye dayalı işlemler, üzerinde yaygın olarak çalışılan konulardan biridir.

Bu derleme çalışmasında, bulut bilişim altyapıları üzerinde verinin farklı boyutları incelenmiş ve her boyuttaki veri işleme stratejilerine yer verilmiştir. Bulut bilişim altyapılarını kullanarak, eldeki verileri analiz etmek isteyen kullanıcılar için verinin farklı boyutları ortaya konulmaya çalışılmıştır. Verinin depolanmasından sorgulanmasına, yüksek ölçekli hesaplamalardan verinin güvenliğinin sağlanmasına kadar pek çok veriye dayalı yaklaşım incelenmiştir. Bu çalışma aracılığıyla bulut bilişim altyapılarında veri ile ilgili çalışılmış olan konular detaylı bir şekilde ortaya çıkarılmış ve az çalışılmış olan ve potansiyel çalışma alanları gözlemlenmiştir. Ayrıca bulut altyapıları üzerinde veri ile ilgili çalışma yapmak isteyen araştırmacı ve şirketler için, hangi veri boyutunun problemlerine en uygun boyut olduğunu ortaya koymaları sağlanmaya çalışılmıştır.

KAYNAKLAR

[1] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A.D., KATZ, R.H., KONWINSKI, A., LEE, G., PATTERSON, D.A., RABKIN, A., STOICA, I., ZAHARIA, M., "Above the Clouds: A Berkeley View of Cloud Computing", 1-23, 2009.

[2] MELL, P., GRANCE, T., "The NIST Definition of Cloud Computing", NIST Special Publication 800-1452011, 2011.

[3] BUYYA, R., YEO, C.S., VENUGOPAL, S., BROBERG, J., BRANDIC, I., "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility", Future Generation Computing Systems, 25, 599-616, 2009.

[4] BUYYA, R., BROBERG, J., GOSCINSKI, A.M., Cloud Computing Principles and Paradigms (1st ed.), Wiley Publishing, 2011.

[5] HURWITZ, J., NUGENT, A., HALPER, F., KAUFMAN, M., Big Data For Dummies (1st ed.), For Dummies, 2013.

[6] COULOURIS, G., DOLLIMORE, J., KINDBERG, T., BLAIR, G., Distributed File Systems. In M. HORTON, M. HIRSCH, M. GOLDSTEIN (Eds.), Distributed Systems: Concepts and Design (pp. 521-564), Addison-Wesley, 2011.

[7] GHEMAWAT, S., GOBIOFF, H., LEUNG, S.T., "The Google File System", SIGOPS Oper. Syst. Rev., 37, 29-43, 2003.

[8] https://www.gluster.org/ (erişim tarihi 27.08.2016)

[9] HUPFELD, F., CORTES, T., KOLBECK, B., STENDER, J., FOCHT, E., HESS, M., MALO, J., MARTI, J., CESARIO, E., "The XtreemFS Architecture - A Case for Object-based File Systems in Grids", Concurr. Comput.: Pract. Exper., 20, 2049-2060, 2008.

[10] ABADI, D.J., "Data Management in the Cloud: Limitations and Opportunities", IEEE Data Engineering Bulletin, 32, 3-12, 2009.

[11] GROSSNIKLAUS, M., "The Case for Object Databases in Cloud Data Management", Proceedings of the Third Conference on Objects and Databases, 25-39, Frankfurt, Germany, 2010.

[12] CURINO, C., JONES, E.P C., POPA, R.A., MALVIYA, N., WU, E., MADDEN, S., BALAKRISHNAN, H., ZELDOVICH, N., "Relational cloud: A Database-as-a-Service for the Cloud", Proceedings of the 5th Biennial Conference on Innovative Data Systems Research, 235-241, Asilomar, California, USA, 2011.

[13] HAERDER, T., REUTER, A., "Principles of Transaction-oriented Database Recovery", ACM Comput. Surv., 15, 287-317, 1983.

[14] TIWARI, S., Professional NoSQL (1st ed.), John Wiley & Sons Inc., 2011.

[15] DEKA, G.C., "A Survey of Cloud Database Systems", IT Professional, 16, 50-57, 2013.

[16] MONIRUZZAMAN, A.B.M., HOSSAIN, S.A., "NoSQL Database: New Era of Databases for Big Data Analytics - Classification, Characteristics and Comparison", International Journal of Database Theory and Application, 6, 1-14, 2013.

[17] DECANDIA, G., HASTORUN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PILCHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P.,VOGELS, W., "Dynamo: Amazon's Highly Available Key-value Store", SIGOPS Oper. Syst. Rev., 41, 205-220, 2007.

[18] http://aws.amazon.com/dynamodb/ (erişim tarihi 27.08.2016)

[19] SUMBALY, R., KREPS, J., GAO, L., FEINBERG, A., SOMAN, C., SHAH, S., "Serving Large-scale Batch Computed Data with Project Voldemort", Proceedings of the 10th USENIX Conference on File and Storage Technologies, 1-13 California, USA, 2012.
[20] http://www.project-voldemort.com/voldemort/ (erişim tarihi 27.08.2016)
[21] http://redis.io/ (erişim tarihi 27.08.2016)
[22] CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W.C., WALLACH, D.A., BURROWS, M., CHANDRA, T., FIKES, A., GRUBER, R.E., "Bigtable: A Distributed Storage System for Structured Data", ACM Trans. Comput. Syst., 26, 1-26, 2008.
[23] http://hypertable.org/ (erişim tarihi 27.08.2016)
[24] LAKSHMAN, A., MALIK, P., "Cassandra: A Decentralized Structured Storage System", SIGOPS Operating Systems Review, 44, 35-40, 2010.
[25] http://cassandra.apache.org/ (erişim tarihi 27.08.2016)
[26] SCIORE, E., "SimpleDB: A Simple Java-based Multiuser System for Teaching Database Internals", SIGCSE Bull., 39, 561-565, 2007.
[27] http://aws.amazon.com/simpledb/ (erişim tarihi 27.08.2016)
[28] LENNON, J., Introduction to CouchDB. In F. POHLMANN, A. COLLETT, K. WİMPSETT, Beginning CouchDB (pp. 3-9), Apress, 2009.
[29] http://couchdb.apache.org/ (erişim tarihi 27.08.2016)
[30] BANKER, K., MongoDB in Action (1st ed.), Manning Publications Co., 2011.
[31] http://www.mongodb.org/ (erişim tarihi 27.08.2016)
[32] WARCHAL, L., "Using Neo4j Graph Database in Social Network Analysis", Studia Informatica, 33, 271-279, 2012.
[33] http://www.neo4j.org/ (erişim tarihi 27.08.2016)
[34] http://infogrid.org/trac/ (erişim tarihi 27.08.2016)
[35] http://www.objectivity.com/infinitegraph (erişim tarihi 27.08.2016)
[36] DEAN, J.,GHEMAWAT, S., "MapReduce: Simplified Data Processing on Large Clusters", Commun. ACM, 51, 107-113, 2008.
[37] FOX, G., GANNON, D., Using Clouds for Technical Computing. In C. CATLETT, W. GENTZSCH, L. GRANDINETTI, G.R. JOUBERT, J.L. VASQUEZ-POLETTI (Eds.), Cloud Computing and Big Data (pp. 81-102), IOS Press, 2013.
[38] LAM, C., Hadoop in Action (1st ed.), Manning Publications Co., 2010.
[39] http://hadoop.apache.org/ (erişim tarihi 27.08.2016)
[40] SHVACHKO, K., KUANG, H., RADIA, S., CHANSLER, R., "The Hadoop Distributed File System", Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 1-10, Nevada, USA, 2010.
[41] OLSTON, C., REED, B., SRIVASTAVA, U., KUMAR, R., TOMKINS, A., "Pig Latin: A Not-so-Foreign Language for Data Processing", Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 1099-1110, Vancouver, Canada, 2008.
[42] THUSOO, A., SARMA, J.S., JAIN, N., SHAO, Z., CHAKKA, P., ANTHONY, S., LIU, H., WYCKOFF, P., MURTHY, R., "Hive: A Warehousing Solution Over a Map-Reduce Framework", Proc. VLDB Endow., 2, 1626-1629, 2009.
[43] BEYER, K.S., ERCEGOVAC, V., GEMULLA, R., BALMIN, A., ELTABAKH, M.Y., KANNE, C.C., ÖZCAN, F., SHEKITA, E.J., "JAQL: A Scripting Language for Large Scale Semistructured Data Analysis", PVLDB, 4, 1272-1283, 2011.
[44] BONVIN, N., PAPAIOANNOU, T.G., ABERER, K., "A Self-Organized, Fault-Tolerant and Scalable Replication Scheme for Cloud Storage", Proceedings of the 1st ACM Symposium on Cloud Computing, 205-216, Indianapolis, Indiana, USA, 2010.
[45] SUN, D.W., CHANG, G.R., GAO, S., JIN, L.Z., WANG, X.W., "Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments", J. Comput. Sci. Technol., 27, 256-272, 2012.
[46] LEI, M., VRBSKY, S.V., HONG, X., "An Online Replication Strategy to Increase Availability in Data Grids", Future Generation Computer Systems, 24, 85-98, 2008.
[47] WU, S., WU, K.L., "An Indexing Framework for Efficient Retrieval on the Cloud", IEEE Data Engineering Bulletin, 32, 75-82, 2010.
[48] JI, S., LI, G., LI, C., FENG, J., "Efficient Interactive Fuzzy Keyword Search", Proceedings of the 18th International Conference on World Wide Web, 371-380, Madrid, Spain, 2009.
[49] AGUILERA, M. K., GOLAB, W.,SHAH, M. A., "A Practical Scalable Distributed B-Tree", Proc. VLDB Endow., 1, 598-609, 2008.

[50] WU, S., JIANG, D., OOI, B.C.,WU, K.L., "Efficient B-Tree Based Indexing for Cloud Data Processing", Proc. VLDB Endow., 3, 1207-1218, 2010.
[51] WEI, L.Y., HSU, Y.T., PENG, W.C., LEE, W.C., "Indexing Spatial Data in Cloud Data Managements", Pervasive and Mobile Computing, 15, 48-61 2013.
[52] WANG, J., WU, S., GAO, H., LI, J., OOI, B.C., "Indexing Multi-Dimensional Data in a Cloud System", Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, 591-602, Indianapolis, Indiana, USA, 2010.
[53] LO, Y.-L., TAN, C.-Y., "A Study on Multi-Attribute Database Indexing on Cloud System", Proceedings of the International MultiConference of Engineers and Computer Scientists, 299-304, Hong Kong, China, 2012.
[54] ZHANG, X., AI, J., WANG, Z., LU, J., MENG, X., "An Efficient Multi-Dimensional Index for Cloud Data Management", Proceedings of the First International Workshop on Cloud Data Management, 17-24, Hong Kong, China, 2009.
[55] NAYAK, D., HUAWEI, B., "Understanding the Security, Privacy and Trust Challenges of Cloud Computing", Journal of Cyber Security and Mobility, 1, 277-288, 2012.
[56] YU, S., WANG, C., REN, K., LOU, W., "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing", Proceedings of the 29th Conference on Information Communications, 534-542, San Diego, California, USA, 2010.
[57] WANG, H., "Privacy-Preserving Data Sharing in Cloud Computing", J. Comput. Sci. Technol., 25, 401-414, 2010.
[58] KHAN, A.N., MAT KIAH, M.L., KHAN, S.U., MADANI, S.A., "Towards Secure Mobile Cloud Computing: A Survey", Future Generation Computer Systems, 29, 1278-1299, 2013.
[59] KAMARA, S., LAUTER, K., Cryptographic Cloud Storage. In R. SION, R. CURTMOLA, S. DIETRICH, A. KIAYIAS, J.M. MIRET, K. SAKO, F. SEBE, Financial Cryptography and Data Security (pp. 136-149), Springer Berlin Heidelberg, 2010.
[60] WANG, C., CAO, N., LI, J., REN, K., LOU, W., "Secure Ranked Keyword Search over Encrypted Cloud Data", Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems, 253-262, Genoa, Italy, 2010.
[61] JIN, L., QIAN, W., CONG, W., NING, C., KUI, R., WENJING, L., "Fuzzy Keyword Search over Encrypted Data in Cloud Computing", IEEE 29th Conference on Computer Communication INFOCOM, 1-5, San Diego, USA, 2010.
[62] NING, C., CONG, W., LI, M., KUI, R., WENJING, L., "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data", IEEE 30th Conference on Computer Communication INFOCOM, 829-837, Shangai, China, 2011.
[63] NICOLAE, B., High Throughput Data-Compression for Cloud Storage. In A. HAMEURLAIN, F. MORVAN, A. MIN TJOA, Data Management in Grid and Peer-to-Peer Systems (pp. 1-12), Springer Berlin Heidelberg, 2010.
[64] WISEMAN, Y., SCHWAN, K., WIDENER, P., "Efficient End to End Data Exchange using Configurable Compression", SIGOPS Operating Systems Review, 39, 4-23, 2005.
[65] KRINTZ, C., SUCU, S., "Adaptive On-the-Fly Compression", IEEE Transactions on Parallel and Distributed Systems, 17, 15-24, 2006.