KOCAELİ ÜNİVERSİTESİ * MÜHENDİSLİK FAKÜLTESİ

Dağıtık Dosya Sistemleri

DÖNEM PROJESİ

Uğurcan ERGÜN

Bilgisayar Mühendisliği

Danışman: Yard. Doç. Dr. Ahmet SAYAR

KOCAELİ, 2013

KOCAELİ ÜNİVERSİTESİ * MÜHENDİSLİK FAKÜLTESİ

Dağıtık Dosya Sistemleri

DÖNEM PROJESİ

Uğurcan Ergün

Bilgisayar Mühendisliği

Danışman: Yard. Doç. Dr. Ahmet SAYAR

KOCAELİ, 2013

Dağıtık Dosya Sistemleri

Uğurcan ERGÜN

Anahtar Kelimeler: Dağıtık dosya sistemleri, Dağıtık sistem, Hataya dayanıklılık, İşletim sistemi, NFS,AFS,GFS

Özet: Dağıtık dosya sistemleri günümüzde disklerin ve depolama kaynaklarının ortaklaşa kullanımı sağlayıp dağıtık sistemler aracılığıyla büyük ölçekte hesaplamalar ve işlemler yapılmasına olanak sağlamaktadırlar. Dağıtık dosya sistemlerinin tasarımında belli mimariler ve sınıflandıramalar mevcuttur. Bu çalışmada dağıtık dosya sistemlerinin tasarımınna dair bir inceleme yapılmış ve ardından günümüzde kullanılan dağıtık dosya sistemleri incelenmiş ve birbirleriyle karşılaştırılmıştır.

Distributed File Systems

Uğurcan ERGÜN

Keywords: Distributed systems, Distributed file systems, fault tolerance,operating systems, NFS,AFS,GFS

Abstract: In present day distributed file systems make it easy to unify disks and storage resources thus it's possible for us to do massive calculations and operations by distributed systems. There are several architectures and specification when it comes to distributed file system design. In this paper some observations has been made for the design of distributed file systems. Then modern distributed file systems has been studied and compared

ÖNSÖZ ve TEŞEKKÜRLER

Dağıtık sistemler günümüzde bilişim dünyasını değiştirmektedir. Örneğin modern web servislerinden (ör. Google,Twitter,Facebook vs.) birini kullanlmak istendiğinde kullanılan web tarayıcının yaptığı basit bir istemci-sunucu bağlantısı olduğu düşünülse de aslında gerçekleşen bu işlemleri bir sunucunun değil, tek bir sistem görüntüsü altında işi paylaşan çokça bilgisayarın dağıtık bir şekilde yapmasıdır. Zira bu kadar karmaşık servisleri tek bir bilgisayarın sağlaması günümüzde pek mümkün olmamakta, mümkün olsa bile ölçeklenebilir olmamaktadır.

Dağıtık dosya sistemleri bu sistemleri mümkün kılan araçlardan bir tanesidir. Çünkü dağıtık sistemlerde salt işlemci gücü ve çevriminin değil işlenilecek verinin kendisinin de paylaşılabilmesi gerekir. Verilerin dağıtık bir şekilde saklanabilmesi hususunda çalışmaların yürütüldüğü dağıtık dosya sistemleri konusu günümüzde önemli bir araştırma alanı olmaktadır.

Beni bu konuda araştırma yapmaya yönelten, araştırma sürecinde yardımlarını esirgemeyen proje danışmanım Yard. Doç. Dr. Ahmet SAYAR'a ve süreç içerisinde beni destekleyen aileme teşekkürlerimi sunmak isterim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER

KB : kilobyte
MB: megabyte
GB : gigabyte
TB : terabyte
PB : petabyte
v : versiyon

Kısaltmalar

AFS : Andrew File System – Andrew Dosya Sistemi

AVSG : Avaible Volume Storage Group – Musait Hacim Depolama Grubu

CMU : Carnegie Mellon Universitesi

DFS : Distributed File System(s) – Dağıtık Dosya Sistemi

DHT : Distributed Hash Table – Dağıtık Hash Tablosu

GFS: Google File System – Google Dosya Sistemi

HDFS : Hadoop File System – Hadoop Dosya Sistemi

IP : Internet Protocol – Internet Protokolü

NFS : Network File System – Ağ Dosya Sistemi

P2P : Peer to Peer

ROWA : Read Once Write All – Bir Kere Oku Hepsine Yaz.

RPC : Remote Procedure Call – Uzak Yordam Çağrısı

VFS : Virtual File System – Sanal Dosya Sistemi

VSG : Volume Storage Group – Hacim Depolama Grubu

1. GİRİŞ

Günümüzde internetin yapısı düşünüldüğünde internet üzerindeki arama,indexleme, depolama gibi işlemleri tekil sunucular üzerinde yapmak mümkün olmamaktadır. Zaten geçtiğimiz yıllarda bilgisayar teknolojisindeki gelişmeler güçlü tekil makinalardan yana değil daha sıradan makinaların birlikte çalışmasından yana olmuştur. Bu çalışmada dağıtık sistemlerin verinin paylaşılması işini yapan dosya sistemleri incelenmiştir. Öncelikle dosya sistemlerine dair temel kavramlar incelenmiş, ardından dağıtık dosya sistemi gereksinimlerine ve mimarilerine göz atılmış ve temel bir çerçeve çizilmeye çalışılmıştır. Sonrasında günümüzde kullanılan modern dosya sistemleri belli açılardan incelenmiş ve değerlendirilmişlerdir. En son olarak karşılaştırma ve sonuçlar açıklanmıştır.

2. TEMEL KAVRAMLAR

Bilgisayarlarda bütün uygulamaların veri depolaması ve gerektiğinde bu veriye tekrar erişebilmeleri gerekmektedir. Süreçler (process) çalıştıkları sürece kısıtlı miktarda veriyi kendi adres alanlarında saklayabilirler. Ancak bu alan bazı uygulamalar için yetmeyecektir. Ayrıca bu veri süreç sonlanınca kaybolacaktır.

Bazı durumlarda verilerin günlerce,aylarca,yıllarca saklanması gerekir verilerin süreç sonlanınca kaybolması kabul edilemez. Verilerin kalıcı bir şekilde sürekli depolanması bilgisayarlarda kullanılan en temel soyutlamalardan biridir. Kalıcı depolama ortamlarındaki temel problemleri (Verinin bulunması, kullanıcıların birbirlerinin verilerini değiştirmesinin önlenmesi, boş alanları tespiti) çözmek için dosya adı verilen yapı kullanılır.

Bir dosya süreçler tarafında üretilen verilerin saklandığı mantıksal birimlerdir. Dosyalar hem verinin kendisini hem de veriyle ilişkili diğer bilgileri içerir. (İngilizce

meta-data olarak adlandırılır.) Bu diğer bilgiler genelde dosya ismi, dosya boyutu, sahiplik bilgisi,zaman damgası, erişim kontrol listesi vs. kayıtları tutarlar. Dosyalar genelde disklerde veya diğer uçucu olmayan ortamlarda depolanırlar.

Dosya sistemleri dosyaların bulunması, depolanması, isimlendirilmesi, korunması, organizasyonu gibi işlerden sorumlu yazılımlardır. Ayrıca programcıları dosyalar için yer tahsisi, depolama düzeni gibi sorunlarla uğraşmaktan kurtarmak için bir programlama arayüzü sunarlar.

Dosya sistemleri işletim sistemleri tasarımı açısından her biri bir alttakini kapsayacak şekilde dört seviyeye ayrılabilir

Tek kullanıcılı, tek süreç işleten sistemler (IBM PC-DOS), Tek kullanıcılı, çoklu süreç işleten sistemler (OS/2), Çok kullanıcılı, çoklu süreç işleten klasik zaman paylaşımlı sistemler (UNIX), Çoklu bilgisayarlarda çok kullanıcılı ve çoklu süreç işleten dağıtık sistemler.

Bu sistemlere ait dosya sistemleri gereksinimleri yukarıya çıktıkça artacaktır. İlk seyivede oldukça basit bir dosya sistemi yeterliyken ikinci düzeyde tutarlılık kontrolüne ve üçüncü seviyede kullanıcılar arası güvenliğe ihtiyaç olacaktır. (Satyanarayanan, 1989, syf. 2)

Bir dağıtık sistem kullanıcılara tek bir sistemmiş gibi görünen birden fazla bağımsız bilgisayarın toplanmasıyla oluşur. (Tannenbaum,Van Steen, 2007, syf. 21)

Dağıtık dosya sistemleri için temel amaç farklı bilgisayarların dosyaları ve depolama kaynaklarını ortak kullanabilmelerini sağlamaktır. Bu amaç klasik zaman paylaşımlı sistemlerin dosya sistemlerine benzer bir şekilde gerçeklenmeye çalışılmakta ve genelde UNIX dosya sistemi temel alınmaktadır.

Bir dağıtık dosya sisteminin yapısını anlayabilmek için servis,istemci ve sunucu kavramlarını da bilmek gerekir.

Servis: Bir veya daha fazla makinada çalışan ve önceden bilinmeyen istemcilere belirli fonksyonlar sağlayan yazılımdır.

Sunucu: Servisin üzerinde çalıştığı bilgisayardan her biridir.

İstemci: Belirlenmiş bir istemci arayüzünün işlemlerini kullanarak bir servisi başlatan süreçtir.

Dağıtık dosya sistemlerinde dosya sistemi istemcilere dosya servisleri sunar. İstemciler istemci arayüzlerini kullanarak belli işlemleri gerçekleştirebilirler. Dosya yaratma,silme,okuma,yazma vs. Bu dosyalar sunucunun diskinde bulundurulur.
Bir dağıtık dosya sistemi bir dağıtık işletim sisteminin parçası olabileceği gibi bir klasik işletim sisteminde işletim sistemi ve dosya sistemi içinde ara katman görevi görebilir.(Levy, Silberschatz, 1990, syf. 322-323)

3. TEMEL GEREKSİNİM VE MİMARİLER

3.1. Temel Gereksinimler

3.1.1. Şeffaflık

Kendini kullanıcılara ve uygulamalara tek bilgisayar gibi sunabilen dağıtık sistemlere şeffaf dağıtık sistemler denir. Şeffaflık dağıtım sistem tasarımında önemli amaçlardan biri olmasına rağmen sistemin tasarımında şeffaflığın mal olacağı performansı ve arttıracağı karmaşıklığı göz önüne almak gerekir. (ISO,1995) 7 çeşit değişik şeffaflık tütü tanımlamıştır. Ancak bunlardan sadece beşi DFS'lerin alanına girmektedir.

Erişim şeffaflığı: Yerel ve uzak dosyalara erişim için tek bir komut seti bulunmalı ve yerel dosyalarla çalışmak için yazılmış programlar uzak dosyalarla değişiklik yapmadan çalışabilmelidirler.

Konum şeffaflığı: Kullanıcı tekil bir isim alanı (namespace) görmeli, Ve dosyalar yolları değiştirilmeden tekrar konumlandırılabilmelidir.

Mobilite şeffaflığı: Dosyalar kullanıcıları etkilemedentaşınabilmedir.

Performansta şeffaflık: Kullanıcı programları sistemde yük altındayken bile performansla çalışmalıdır

Ölçeklenebilirlikte şeffaflık: Servis farklı ağ boyutları, yük dağılımları ile başa çıkabilmek için genişletilebilir olmalıdır. (Coulouris et al., 2011, syf. 527)

3.1.2. Eşzamanlı Dosya Güncelleme

Bir dosyada bir istemci tarafından yapılandeğişiklikler diğer istemcilerin çalışmalarını ve dosyada değişiklik yapmalarını engellememelidir.
Paylaşılan veriler eşzamanlı dosya güncelleme çözümleri maliyetli olduğu için dosya servisleri genelde UNIX tarzı kilitleme sistemleri kullanır.

Dosya sistemlerinde bir dosyayı birden fazla süreç tarafından işlenebileceği için bu gibi durumlarda neler olacağını yani sekronizasyon semantiği tanımlamak gerekir. Temelde dosya sistemlerinde 4 farklı semantik kullanılır.

UNIX semantiği: Önce yazma, sonra okuma. Bütün değişiklikler bütün süreçlerde anlık olarak görülebilir

Oturum semantiği: Yapılan değişiklik dosya kapatılana kadar sadece değişikliği yapan sürece gözükür.

Salt okunur dosya semantiği: Dosyalara yazma işlemi yapılamaz.

Atomik işlem semantiği: Yazma isteği birleştirilir ve tek bir işlem halinde yollanır bu şekilde işlemin bir bütün olarak uygulandığından emin olunur.

3.1.3. Dosya Replikasyonu

Dosyaların birden fazla kopya halinde DFS'lerde saklanmasının iki temel avantajı vardır güvenilirlik ve performans. Eğer veri sunucular arasında kopyalanmış olursa bir kopyaya erişilememesi veya bir kopyanın bozulması durumlarında o veriler üzerinde çalışmaya devam edilebilir. Ayrıca verinin replikasyon yoluyla dağıtılması sunucular arasında yük dengelenmesini sağlayacak ve performansı arttıracaktır. Başlarda DFS'lerin pek çoğunda sunucu tarafında replikasyon desteği bulunmazken modern DFS'lerin çoğunluğu bu desteği sağlamaktadır.

3.1.4. Tutarlılık

Replikasyon güvenilirlik ve performans gibi belirli avantajlara sahip olmasına rağmen ortaya yeni bir sorun çıkarmaktadır. Tutarlılık.Eğer kopyalardan biri değiştirilirse diğer kopyalardan farklı bir hale gelecektir ve bu durumda kopyaların tutarlılığının sağlanması gerekecektir. Standart UNIX sistemlerde bu sorun one-copy-

update yöntemi ile çözülmeye çalışılır. Bir dosyada yapılan değişiklik anlık olarak diğer süreçlere yansıyacaktır. Ancak bu yöntem dağıtık sistemlere pek uygun olmadığından bu ilkeden sapılması gerekir

DFS'ler genelde performansı arttırmak için önbellekleme yaparlar. Önbellekleme performansı arttırırken önbelleğe alınmış dosyaların sunuculardaki dosyalarla aynı dosya olduğundan emin olunması gerekir ve bu ortaya bir tutarlılık sorunu çıkartır. Eğer bir dosya bir istemci de önbelleklendikten sonra sunucuda değişirse önbelleklenmiş kopya eskimiştir ve kullanımı sağlıklı olmayacaktır. Bu duruma karşı bazı önlemler alınmalıdır.

3.1.5. Platform Bağımsızlık

Dosya servis arayüzleri, sunucu ve istemci yazılımlarının farklı işletim sistemlerinde ve farklı donanımlarda çalışacak bir şekilde tasarlanmalıdır.

3.1.6. Hata Toleransı

Dağıtık sistemleri, merkezi sistemlerden ayıran en önemli kavramlardan biri kısmi hata kavramıdır. Merkezi sistemlerin aksine sistemin bir parçasında bir hata oluştuğunda sistemin geri kalanının etkilenmeme olasılığı vardır. Dağıtık sistemlerdeki önemli amaçlardan bir tanesi de sistemin hatalardan performansı baltalamadan kurtarılmasıdır. Özellikle bir hatayla karşılaşıldığında sistem gerekli onarımlar yapılana kadar çalışabilmeli yani sistem hataları tolere edebilmeli ve varlıklarında dahi çalışabilmelidir. (Tannenbaum,Van Steen, 2007, syf. 321)

Dosya sunucu semantiği dosya sistemlerinde iki türlüdür. Durumsuz servisler ve durumlu servisler.

Durumsuz servisler: Dosyaların durumu hakkında herhangi bir bilgi tutulmaz.Bütün istekler müstakil olarak yapılır. Dosya açma veya kapama gibi işlemler kullanılmaz. İstemci her istekte bulunduğunda dosya ile ilgili gerekli bilgileri ve dosya içindeki konumu bildirmelidir. Sunucu dosyalarla ilgili durum kayıtları tutmakla zaman ve hafıza harcamaz. Hata toleransı yüksektir. Herhangi bir sunucu hatasında ek işlemler yapılmaz. Sunucunun tekrar başlatılması yeterlidir.
Kilitleme ve senkronizasyon desteği zayıftır ve ayrıca performansı iyi değildir.Çünkü her bağlantıda daha fazla veri gönderilmekte ve işlenmektedir.

Durumlu servisler: Dosyaların durumlarıyla ilgili bilgiler tutulur. Sistem üzerinde açılan dosyalara dair kayıtlar tutulur. İstemci bir dosya açtığında ona bir bağlantı belirteci atanır. Bu belirteç daha sonraki bağlantılarda kullanılabilir. Sunucu yazma değişken, kilitler gibi bilgiler tutatabilir. Sunucu biten işlemlerle ilgili bilgileri temizlemedir. Olası bir sunucu hatasında sunucu istemcilerden durum bilgilerini tekrar istemelidir. Olası bir istemci hatalarından sunucu haberdar olmalıdır. İstemci hatalarında o istemcinin kayıtlarına yetim kayıtlar denir. Yetim kayıtlar bulunup temizlenmelidir. Gönderilen ve işlenen veri sayısı düştüğü için performans daha iyidir.

3.1.7. Güvenlik

Bütün dosya sistemlerinde erişim kontrolü erişim kontrol listeleri yoluyla yapılır.

Dağıtık dosya sistemlerinde istemcinin kimliğinin doğrulanması sistemin korunması için önemli olduğundan yetkilendirme dijital imza vebazen kriptolama kullanılarak yapılır.

3.1.8. Verimlilik

Dağıtık dosya sistemlerinin en az klasik dosya sistemleri kadar performanslı ve güvenilir olmaları gerekmektedir. Aynı zamanda yönetimlerinin pratik olmaları ve yöneticilere işlemlerde kolaylık sağlamalı gerekli yönetim araçlarını sağlamalıdırlar. (Coulouris et al., 2011, syf. 528)

3.2. Mimariler

3.2.1. İstemci-Sunucu Mimarisi

Bu mimaride sistemden sorumlu bir veya birden fazla sunucu vardır. Her sunucu kendi yerel dosya sistemine dair standartlaşmış bir görüntü sunar. İstemci sunucu iletişimi genelde RPC ile sağlanır. Dosyalara erişmek ve dosyaları aktarmak için iki tane model vardır. Uzaktan erişim modeli ve yükleme/indirme modeli.

Uzaktan erişim modelinde sunucunun kullanıcıya dosya işlemleri için yerel dosya sistemine benzer bir arayüz sağlaması gerekir.

Yükleme/indirme modelinde istemci sunucudan dosyaları indirir kendi

yerelinde işlemlerini,değişikliklerini yapar. Eğer bir değişiklik yapıldıysa sunucuya dosyayı tekrardan geri yükler.

3.2.2. Küme (Cluster) Tabanlı İstemci-Sunucu Mimarisi

Klasik sunucu istemci mimarisi biraz geliştirilerek sunucu kümelerine de uygulanabilir. Sunucu kümeleri genelde paralel uygulamalar için kullanıldıkları için dosya sistemleri de buna göre tasarlanır. En çok bilinen tekniklerden biri de dosya ayırma tekniğidir. Bu teknikte dosyaların kendileri de parçalara bölünür ve sistem boyunca dağıtılır.

Bu mimari sadece paralel dosya erişiminin makul olduğu uygulamalarda verimli çalışabilir ki genelde verinin matrisler gibi çok düzenli yapılarda saklanmasını gerektirir Aksi takdirde dosyalar klasik olarak bütün halinde saklanabilir.

3.2.3. Simetrik Mimari

Bu mimari de diğer mimarilerde olduğu gibi merkezi kontrol yapan sunucular yoktur ve bütün düğümler fonksyonel olarak simetriktirler. Bu mimari aynı zamanda P2P dosya sistemi olarak da anılır. Bu tarz sistemler genelde DHT kullanılarak gerçeklenirler. DHT'ler normal hash tabloları gibi anahtar değer çiftleri içerir ve düğümler anahtar alanı dağıtılır, bütün düğümler komşu düğümlerinin listesini bir yönlendirme tablosunda tutar.

DHT'nın gerçeklendiği dört temel protokolü vardır. Chord, CAN, Tapestry, Pastry. Bunlardan en popülerlerinden olan Chord DFS gerçeklemelerinde kullanılmaktadır.

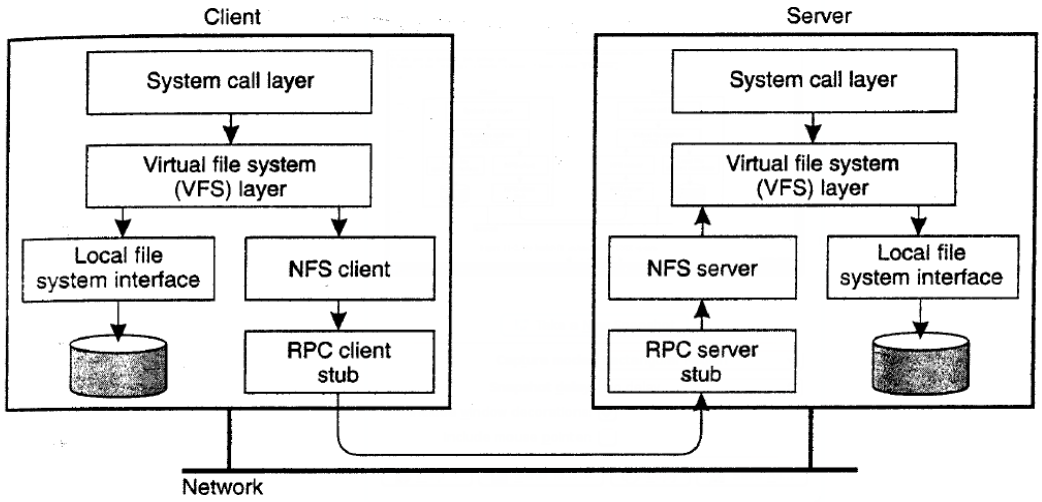
(Wikipedia DHT,2011)

4. ÖRNEK DOSYA SİSTEMLERİ

4.1. Sun Ağ Dosya Sistemi (NFS)

4.1.1. Genel Bakış

NFS, Sun Microsystems tarafından ilk olarak 1984 yılında (Sandberg et al.,1985) geliştirilmeye başlanmış hem bir dağıtık dosya sistemi protokolünün hem de dosya sisteminin kendisinin adıdır. 1989 yılında ikinci sürümü (RFC 1094), 1995 üçüncü sürümü (RFC 1813) ve 2000 yılında dördüncü sürümü çıkmıştır (RFC 3010,3530) . Açık bir standart olarak tasarlanmıştır. Ve UNIX sistemlerde açık ara en çok kullanılan ve gerçeklenmiş DFS'lerden biridir. RPC iletişim modelini kullanılır. Bu



Şekil 4.1: NFS genel mimarisi

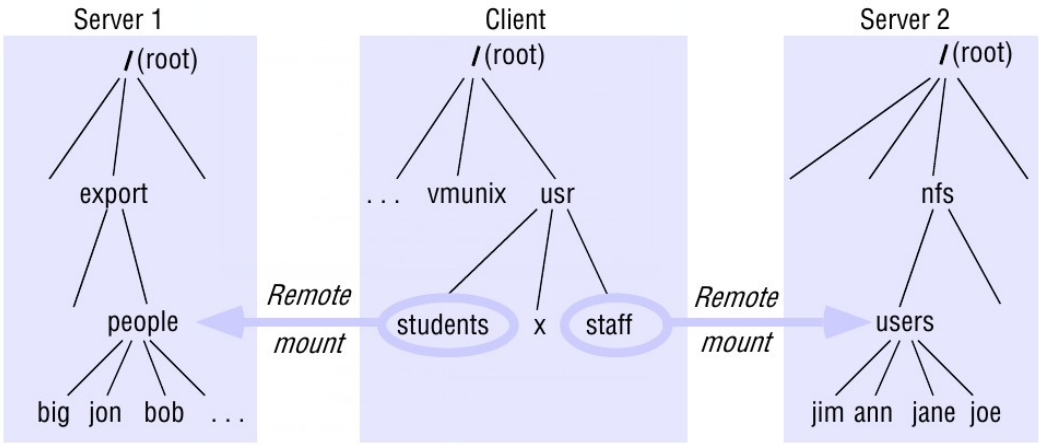
çalışmada daha çok NFSv3 ve NFSv4 sürümlerinden bahsedilecektir.

NFS'in genel mimarisi Şekil 4.1 'de (Tannenbaum,Van Steen, 2007, syf. 493) görülebilir. NFS'te genel tasarım amacı klasik UNIX dosya sisteminin dağıtık bir şekilde gerçeklenebilmesidir. Standart UNIX dosya arayüzü VFS ile değiştirilmiştir. VFS programcıların farklı dosya sistemleriyle zahmetsizce çalışabilmesi için işletim sistemi çağrıları ve dosya sistemleri arasında bir ara katman görevi görür.

İletişim için RPC kullanılırken NFSv3'te her komut için ayrı RPC yapılırken NFSv4'te bir RPC'de birden fazla komut gönderilebilir.

4.1.2. İsimlendirme ve Şeffaflık

NFS isimlendirme modelinin temellerinden biri kullanıcıların dosyalara şeffaf bir şekilde erişmesini sağlamaktadır. Bu istemcilere uzak bir dosya sistemini kendi dosya sistemlerine bağlama imkanı vererek sağlanmaya çalışılır. NFS'te dosya sistemlerinin tamamı değil sadece belli kısımları da bağlanabilir. Dizinlerin



Şekil 4.2: NFS içerisinden dizin bağlama

bağlanmasının bir örneği Şekil 4.2 de gösterilmiştir. (Coulouris et al., 2011, syf. 540) Bağlama işleminin kendisi bağlanacak sunucunun konumunun bilinmesini gerektiği için şeffaflık ilkesini sağlamamaktadır ancak bağlama işlemi tamamlandıksan sonraki erişimler şeffaf bir şekilde yapılmaktadır. Bağlama işlemi aşağıdaki komut ile yapılır "mount -t nfs4 -o proto=tcp,port=2049 nfs-server:/users /home/users"

Bir dizinin istemcilerin erişimine açılabilmesi için sunucu bu dizini erişime açtığına dair bildirimde bulunmalıdır. Bu bildirim genelde UNIX türevi sistemlerde /etc/exports dosyasına bu dizin ve gerekli parametreler eklenerek yapılır.

```
/home @myclients(rw,sync,no_subtree_check)
/usr/local @myclients(rw,sync,no_subtree_check)
/home 192.168.0.10(rw,sync,no_subtree_check) 192.168.0.11(rw,sync,no_subtree_check)
/usr/local 192.168.0.10(rw,sync,no_subtree_check) 192.168.0.11(rw,sync,no_subtree_check)
/home 192.168.0.0/255.255.255.0(rw,sync,no_subtree_check)
/usr/local 192.168.0.0/255.255.255.0(rw,sync,no_subtree_check)
```

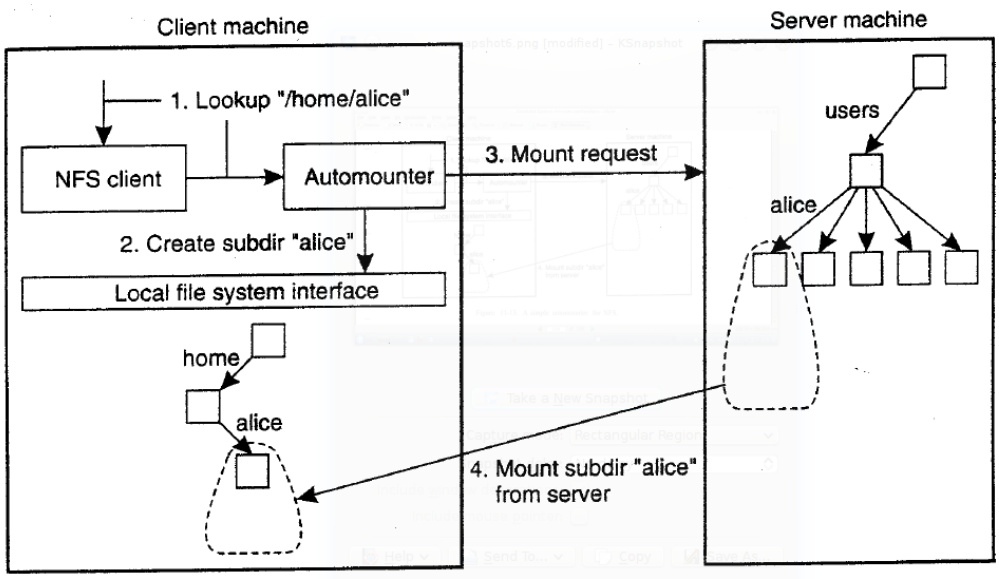
Şekil 4.3: /etc/exports dosyası örneği

Şekil 4.3'de bir /etc/exports dosyası yer almaktadır. İlk iki kayıtta /home ve /usr/local dizinleri myclients grubunun erişimine açılmıştır. 3. ve 4. kayıtlarda bu dizinler belli IP adreslerinin, 5. ve 6 kayıtlarda ise belli ip bloklarının erişimine açılmıştır. (Ubuntu ,2012)

Dosya ismi çözümlemeleri NFSv3 ve önceki versiyornlarda iteratif olarak yapılır. Bu demektir ki /home/ugurcan/proje.pdf dosyasına ulaşılacaksa 3 farklı lookup çağrısı yani 3 tane RPC isteği gönderilecektir. Bunun sebebi global bir isim alanı bulunmaması ve dosya adı doğrudan çözülmüş olsaydı her bir istemcinin kendi isim alanına göre göndereceği dosya yollarının geçersiz olabilmesidir.
NFSv4 ise bundan farklı olarak recursive çağrı yapabilmektedir.

Dosya belirteçleri (file handles) dosya sistemlerinde dosyalara karşılık gelen tekil referanslardır. Dosyayla beraber yaratılırlar ve dosyaların isimlerinden bağımsızlardır ve dosya silindiğinde tekrar kullanılamazlar. İstemciler belirteçlerin içeriğine ulaşamazlar. Dosya belirteçleri NFS v2'de 32 bit ile, v3'de 64 bit ile ve v4'te 128 bit ile temsil edilirler.(Tannenbaum,Van Steen, 2007, syf. 321)
İstemciler bir dosya sistemini veya dizini kendilerine bağladıklarında onlara bağlandıkları kök dizinin dosya belirteci gönderilir. Dosya işlemlerinin çoğu belirteçlerle yapılır. İstemci bir dosya işlemi gerçekleştirmeden önce dosya isminden belirteci sorgulamalıdır. Buna lookup çağrısı denir.

Dosya sistemlerinin elle bağlanması çok pratik olmadığından otomatik bağlama yöntemi geliştirilmiştir. Erişilmeye çalışan ve yerel dosya sisteminde bulunmayan bir dizin eğer NFS lookup çağrısı ile bulunabiliyorsa bağlanacaktır. NFS otomatik bağlayıcısı modern Linux sistemlerde autofs adı ile kullanılır.



Şekil 4.4: Otomatik bağlama örneği

Otomatik bağlama işlemine dair bir örnek Şekil 4.4'de görülebilir.

4.1.3. Senkronizasyon

DFS'lerde daha çok oturum semantiği metodu kullanılmasınna rağmen NFS'e içkin herhangi bir senkronizasyon metodu bulunmamaktadır. Bu sorunu çözebilmek için NFS'te ayrı bir dosya kilitleme sistemi kullanılır.

NFSv3 kilit sisteminde 3 farklı kilit değişkeni bulunur:

Lock : Belirli bir byte dizisini kilitler

Lockt : Belirli bir byte dizisinin üzerinde kilit olup olmadığını kontrol eder.

Locku : Belirli bir byte dizisi üzerindeki kilidi kaldırır.

4.1.4. Tutarlılık ve Replikasyon

İstemci önbellekleme için NFSv3 protokolünde bir tanımlama yapılmamış ve bu gerçeklemeye bırakılmıştır. Sun firmasının gerçeklemesinde bir dosyanın önbelleklenmiş kopyasının zaman mührü ile ve sunucudaki kopyanın zaman mührü karşılaştırılır. Eğer mühürler farklıysa dosya değişmiş demektir ve önbellekteki

kopya çöpe atılır. Ayrıca önbellekteki dosyalar 3-30 aralığında dizinler 30-60 saniye aralığında çöpe atılır.(Coulouris et al., 2011, syf. 543)

NFS'te sunucu tarafında önbellekleme yazma işlemlerinde yapılır. Önbellekteki değişiklik diske yazılmadan kullanıcıya yazma işlemi onayı gönderilmez.

NFS sunucularında dosya replikasyonu yapılmamaktadır.

4.1.5. Hata Toleransı

NFSv3'te hata toleransı için durumsuz sunucu modeli kullanılırken NFSv4'te durumlu sunucu modeline geçilmiştir.

4.1.6. Güvenlik

NFS'te yetkilendirme sunucu tarafında yapılır. Güvenlik için güvenli RPC kullanılmasının yanı sıra standart UNIX erişim kontrol modeli uygulanır. İstemci her bağlantıda UNIX kullanıcı ve grup id'sini gönderir.

Bu çok güvenli bir yöntem değildir. Ancak NFS istendiğinde Kerberos ağ yetkilendirme sistemini kullanacak şekilde ayarlanabilir.

4.2. Andrew Dosya Sistemi

4.2.1. Genel Bakış

Andrew Dosya Sistemi 1983 yılında Carnegie Mellon Universitesi ve IBM işbirliği ile geliştirilmeye başlanmıştır. Tasarımın temel önceliği ölçeklenebilir olmasıdır. AFS ekibi hedeflerini dosya sisteminin 5000 istemciye kadar desteklenmesi olarak koymuşlardır. (Satyanarayanan, 1990). AFS'in üç sürümü mevcuttur. AFSv1, AFSv2 1985, AFSv3 1988.

Tasarım esasları şöyle açıklanmıştır. (Satyanarayanan, 1990)

- Bir iş hem istemci hem sunucuda yapılabiliyorsa istemci de yapılmalıdır. Sunucuya sadece kritik işlemler yaptırılmalıdır
- Mümkün oldukça önbellekleme yapılmalıdır

- Tasarımda genel kullanım alışkanlıklarından faydalanılmalıdır.
- Sistem bazında veri ve değişim miminumda tutulmalıdır.
- Mümkün oldukça az nesneye güvenilmelidir.
- İşlemler mümkünse toplu halde yapılmalıdır.

Yukarıda da bahsedildiği gibi AFS bazı kullanım alışkanlıkları gözetilerek tasarlanmıştır bunlar aşağıdaki gibidir.

- Sistem komutları,kütüphaneleri ve kullanıcı ev dizini gibi dosyaların yerel önbellekteki kopyaları uzun süreler boyunca geçerli olacaktır. Dosya erişimlerinin bir çoğu dosyalara yapılmaktadır. Sistem dosyaları nadiren değişecektir ve kullanıcı ev dizininde yapılan değişiklik zaten önbellekte olacaktır.
- AFS dizaynı dosya boyutları ve kullanımları üzerine bazı varsayımlardan etkilenmiştir.Bu varsayımlar UNIX sistemler üzerine yapılan bazı gözlemler ve çalışmalara dayanır (Satyanarayanan 1981, Ousterhout et al. 1985, Floyd 1986) Bunlardan en önemlileri
  - Dosyalar genelde 10KB'dan küçüktür.
  - Yazma işlemlerinden fazla okuma işlemi yapılır
  - Genelde sıralı erişim yapılır,rastgele erişim nadirdir.
  - Çoğu dosya sadece tek kullanıcı tarafından kullanılır.

AFS'te çalışan istemci yazılımları Venus, sunucu yazılımlarına ve bazen sunucuların tamamına Vice ismi verilir.

4.2.2. İsimlendirme ve Şeffaflık

AFS dosya isim alanı ikiye ayrılır. Yerel ve paylaşılan, paylaşılan isim alanı şeffaf bir şekilde bütün kullanıcılarda aynıdır. İsim alanının parçalara ayrılması şeffaflığın biraz azalmasına sebep olsa da bu durumun sıradan kullanıcılar tarafından fark edilme olasılığı düşüktür. AFS-1'de adres dönüşümleri çok fazla performans kaybına yol açtığı için (Arpacı-Dusseau,2012) AFS-2'de hacim (volume) denilen mantıksal

yapılar oluşturulmuştur.

AFS aynı zamanda dosya belirteci yapısını kullanmaktadır. Her dosya ve dizin için fid denilen bir belirteç tutulur. Bu belirteç hacim numarası, dosya belirteci (Bkz. Başlık 4.1.2) ve bir tekilleştirici bulunur.

Hacim bilgileri her sunucuda kopyalanmış bir durumda bulunan hacim konum veritabanında bulunur. (Levy, Silberschatz, 1990, syf. 363)

4.2.3. Senkronizasyon

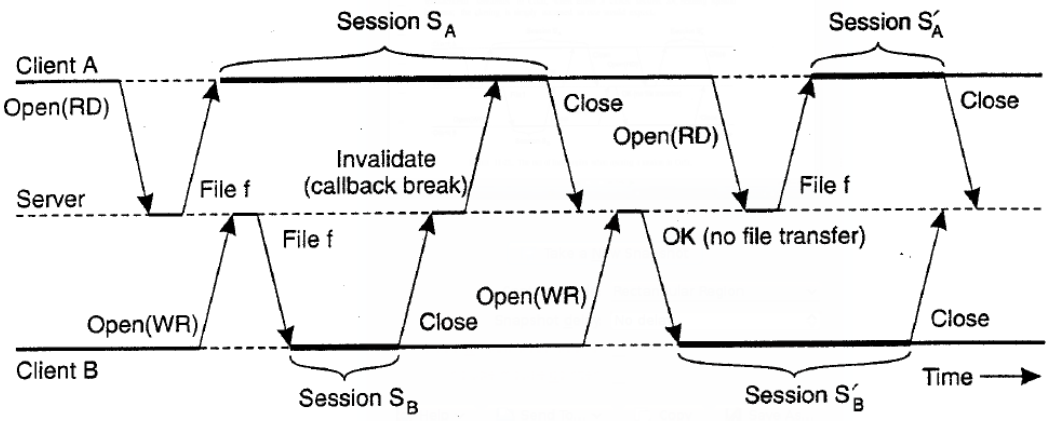
AFS senkronizasyon için oturum semantiğini kullanmaktadır. Dosyalar kapatılınca değişiklikler sunucuya gönderilir. Eğer iki istemci aynı dosyaya yazma işlemi yaptılarsa değişiklikleri sonra gönderen istemcinin değişiklikleri geçerli olacaktır.

4.2.4. Tutarlılık ve Replikasyon

NFS'in aksine AFSv1 ve AFSv2 de dosyalar ve dizinler parçalar halinde değil bütünler halinde yollanır ve bu şekilde önbelleklenir. AFSv3 için 64KB'lık parçalar kullanılır. (Coulouris et al., 2011, syf. 548) Burada amaç okuma işlemleri için sunucuyu devreden çıkarmak ve sunucuya sadece değişiklikleri göndermek veya almak için başvurmaktır. AFSv1 de önbelleğin kontrolü çok fazla kaynak tükettiği için (Arpacı-Dusseau, 2012) AFSv2 de callback adı verilen bir mekanizma getirilmiştir.

Buna göre istemci aksi söylenene kadar bütün önbellekleri geçerli kabul edecektir. Bu sistem şöyle işlemektedir. Bir istemci bir önbelleklediği zaman o dosya için sunucudan bir callback jetonu alır ve sunucu o istemcinin o dosyayı önbelleklediğine ve o dosya üzerinde yapılacak değişikliklerden haberdar edilmesi gerektiğine dair callback promise adlı bir kayıt tutar.

Callback jetonlarının geçerliliği sürekli değil sadece ilgili dosya açılırken sorgulanır. İstemci bir dosya da değişiklik yaptığı zaman bunu sunucuya bildirir. Sunucu o dosya için callback promise kaydı tuttuğu bütün istemcilere birer callback break mesajı gönderir yani o callback jetonlarını geçersiz kılar. Böyle o dosyanın kopyasını tutan bütün istemciler dosyayı açmak istediklerinde callback jetonlarının geçersiz olduğunu görüp sunucudan dosyayı tekrar indireceklerdir.



Şekil 4.5: Bir callback döngüsü

Örnek bir callback döngüsü Şekil 4.5 te görülebilir.

4.2.5. Hata Toleransı

AFS durumlu bir servis olduğu için hata toleransı NFS'e göre biraz daha düşüktür.

Bir istemci çöktüğünde callback jetonlarını çöpe atar ve sunucuya önbellekteki kopyalarının kontrol edilmesi için başvurur.

Bir sunucu çökerse istemciler bunu fark edip jetonlarını çöpe atmalı ve önbellekteki kopyalarının kontrol edilmesi için başvurmalıdırlar.

4.2.6. Güvenlik

AFS'te güvenlik işlevi Vice sunucularına devredilmiştir. Bu sunucular fiziken güvenli kabul edilir ve üzerilerinde sadece güvenilen sistem yazılımları çalıştırılır. AFS güvenli RPC ve erişim kontrol mekanizmalarını içermektedir.

Ayrıca AFSv3'ten itibaren yetkilendirme için Kerberos kullanılmaya başlanmıştır.

4.2.7. Türevleri

OpenAFS: 1989 yılında CMU 'da çalışan bazı araştırmacılar AFS için kurumsal çözümler üretmek üzere Transarc isminde bir firma kurdular. Bu firma 1994 yılında IBM tarafından satın alındı. Transarc AFS 2000 yılına kadar IBM tarafından geliştirildikten sonra OpenAFS adıyla açık kaynaklı olarak yayınlandı.

OpenAFS bir AFSv3 gerçeklemesi olup son sürümünü 10 Aralık 2012 tarihinde çıkarmıştır. Web sayfası openafs.org (OpenAFS FAQ)

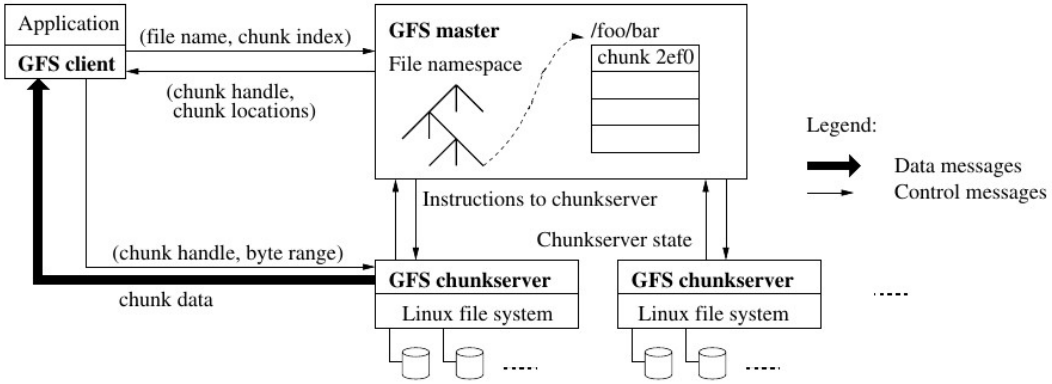
Coda : CMU'da AFS'i geliştiren Prof. Dr. Mahadev Satyanarayanan ve ekibinin AFS-2 tabanlı bir araştırma projesidir. AFS-2'den temel farkı sunucu tarafında replikasyon, çevrimdışı işlem gibi erişilebilirliği arttıran özellikler getirmesidir. Sunucu replikasyonu şu şekilde yapılır. Bir AFS haciminin bulunduğu sunucuların listesine VSG adı verilir. Bir istemcinin bir hacimin VSG'u içerisinden erişebildiği sunucuların listesine AVSG denir. AVSG 0 ise istemcinin bağlantısı kesilmiş demektir. Replikaların tutarlılığın sağlanması için ROWA adında bir teknik kullanılır. Bu teknikte bir istemci bir veri okuyacağı zaman AVSG içinde bir sunucudan okur. Ancak yazma işlemini MultiRPC yardımıyla bütün AVSG'ye yapar. VSG çökmelerinde tutarlılığı sağlamak için versiyon vektörleri kullanılır. Son kararlı sürüm 30 Mart 2010 tarihinde çıkarılmıştır Web sayfası : http://www.coda.cs.cmu.edu/ (Tannenbaum, Van Steen, 2007, syf 525)

Arla: İsveç'te KTH Kraliyet Teknoloji Enstitüsü tarafından geliştirilen bir AFS istemcisidir. Arla'nın OpenAFS'ten temel farkı temel işlevlerin çekirdeğe yerleştirilmeyip kullanıcı seviyesinde daemonlar kullanılarak halledilmeleridir. (UNIX'te sistem düzeyinde çalışan kullanıcı süreçleri) Son kararlı sürüm 10 Ocak 2007 tarihinde çıkarılmıştır. Web sayfası: http://www.stacken.kth.se/project/arla (Westerlund,Danielsson, 1998)

4.3. Google Dosya Sistemi

4.3.1. Genel Bakış

Google yaptığı iş gereği çok büyük verilerle çalışmaktadır. Verileri yüksek kapasiteli az sayıda sunucuda depolamak ve işlemek oldukça maliyetli olmakta ve ölçeklenebilir olmamaktadır. Çok miktarlarda ucuz donanım kullanıp sistemi hatalara karşı daha dayanıklı olması için tasarlamak daha uygun bir çözümdür. GFS'te mantıksal dosya gösterimine yığın denir her bir yığın 64MB veriden oluşur.



Şekil 4.6: GFS mimarisi

Şekil 4.6'da görüldüğü gibi GFS mimarisi bir adet master sunucu ve çokça yığın sunucuları bulunur. Verilerin kendileri yığın sunucularda bulunurken master sunucuda sadece metadata bulunur. Master sunucu yığın sunucularla sürekli kalp atışı adı verilen mesajlarla haberleşir. İsim alanının yönetilmesi, yığınların yaratılması,kopyalanması,dengelenmesi, eski yığınların silinmesi masterin görevidir. Master aynı zaman çöp toplama (GFS'te dosyalar direk silinmez, silinecek olarak işaretlenir. Silme işlemi çop toplama sırasında yapılır) işini de yapar.

4.3.2. İsimlendirme ve Şeffaflık

GFS'te dizinler için ayrı bir veri yapısı bulunmaz. Ayrıca kısayol,bağ gibi yapılar da mevcut değildir. GFS isim alanını mantıksal olarak her yol isimlerini metadatalara eşleyen lookup tablolarıyla düzenler. Bu sıkıştırma da kullanılarak etkin bir bellek kullanımı sağlar.

GFS master içerisinde yapılan bazı işlemler uzun sürebileceği için bunu diğer işlemleri etkilememesi için kilitleme mekanizmaları kullanılır. Kilitler dosya yolu ağacındaki her bir düğüm için alınır. Örneğin /home/ugurcan/deneme dosyasına yazılmak istendiğinde GFS /home ve /home/ugurcan için okuma kilitleri /home/ugurcan/deneme için yazma kilidi atayacaktır.

4.3.3. Senkronizasyon

GFS'te dosyalar iki türlü işlem ile değiştirilebilir. Yazma ve kayıt ekleme.

Yazma işlemleri için master yığınlar arasından bir adet birincil yığın seçer yazma işlemi bu yığına yapılır ve buradan yığının diğer kopyalarına dağıtılır. Birincil

yığın 60 saniye sonra bu yetkiyi mastera iade eder. Eğer işi bitmemişse yetkinin uzatılmasını isteyebilir. Ağın kapasitesinin gereksiz yere harcanmaması için her sunucu kendine ağ topolojisinde en yakın sunucuya gönderimde bulunur.

Kayıt ekleme işleminde yazılacak verinin konumu belirtilmez sadece veri belirtilir. GFS bu veriyi kendi seçeceği bir yere yazar ve yazdığı yeri istemciye söyler. Bu sistem birden fazla istemci bir dosyaya yazmaya çalıştığında veri kaybını önlemektedir.

4.3.4. Tutarlılık ve Replikasyon

GFS'te istemciler için önbellekler tutulmaz. Ancak her yığının kopyaları (ortalama 3 kopya) çeşitli sunucularda bulundurulur. Master'ın operasyon kayıtları ve checkpointleri çeşitli sunucularda bulundurulur. Master'ın kayıtlarına sahip gölge masterlar salt-okunur hizmet verebilir.

4.3.5. Hata Toleransı

Master ve yığın sunucular saniyeler içerisinde tekrar başlatılabilirler. Master tekrar başlatıldığında öncelikle diskten operasyon kaydını okur. Sonra bütün yığın sunucularına ellerindeki yığınları sorar. Yığın versiyonlarına göre kendi bilgisini günceller veya bir yığını eski olarak belirler. Eğer bir yığın sunucu çökerse bu kalp atışı mesajları gelmediği için fark edilir ve yığınların başka kopyaları değişik sunucularda tekrar oluşturulur.

4.3.6. Güvenlik

GFS referans dökümanında (Ghemawat et al.,2003) herhangi bir güvenlik mekanizması tanımlanmamıştır.

4.3.7. Türevleri

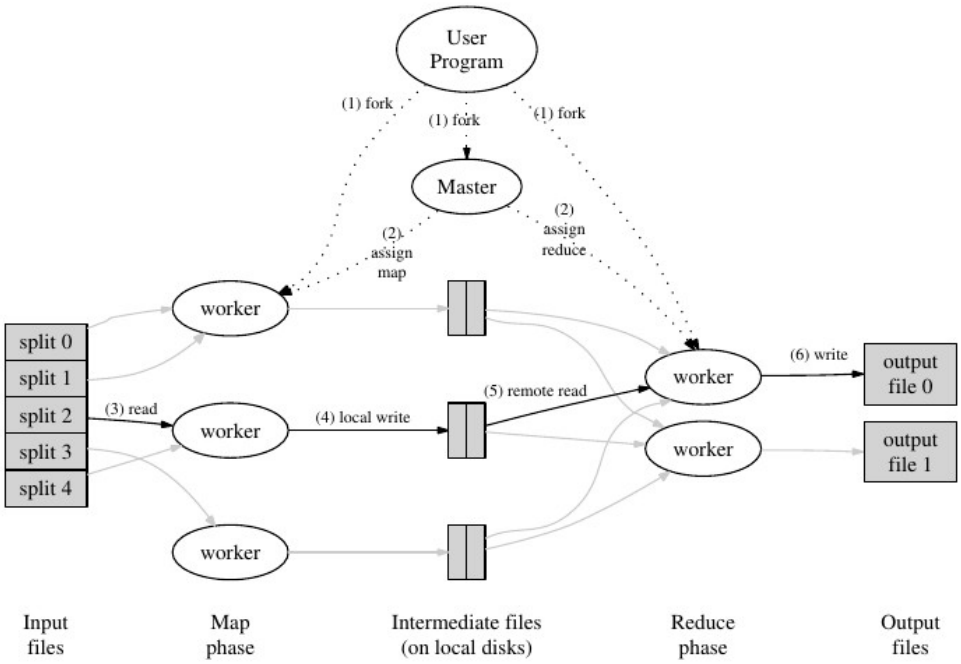
Hadoop Dosya Sistemi (HDFS): HDFS Apache Vakfı tarafından geliştirilen Hadoop projesinin bir parçasıdır. HDFS, GFS'in açık kaynaklı bir gerçeklemesidir. Orjinal GFS C++ ile yazılmışken HDFS Java ile yazılmıştır.

GFS'teki Master,Chunk server, Operation Log, Shadow server gibi yapılar HDFS'te

Namenode, Datanode,Edit log, Secondary namenode gibi isimler almışlardır. HDFS'te GFS'ten farklı olarak kayıt ekleme,çöp toplama gibi mekanizmaları sağlayamamaktadır. (Shankaran, Sharma 2011)

4.3.8. Kullanıldığı Uygulamalar

Map-Reduce: Büyük bir işlemi paralel bir şekilde farklı bilgisayarlara yaptırabilmek için Google'ın bulduğu bir yöntemdir. (Dean, Ghemawat, 2004) Adını Lisp programa dilindeki map ve reduce fonksyonlarından alır. Map fonksyonu bir anahtar değer setinden ara anahtar değer setleri üretir. Bu veriler dağıtık bir şekilde işlendikten sonra Reduce fonksyonu aynı anahtara sahip değerleri aynı değerde birleştirir.



Şekil 4.7: Map-Reduce işletim örneği

Şekil 4.7 'deki işletim şu şekilde olmaktadır.

1- Map-Reduce kütüphanesi girdi verisini M parçaya böler

2- Programın master kopyası işçi süreçlere map veya reduce işi atar.

3-Map işi yapan bir işçi süreç girdi verisinin anahtar ve değerlerini kullanıcı tanımlı map fonksyonuna gönderir

4-Map fonksyonun ürettiği ara değerler önce önbellekte tutulur ardından diske

yazılır. Yazılan konum mastera bildirilir.

5- reduce işi yapan bir işçi süreç bu yeri öğrendiğinde bu ara değerleri diskten RPC yardımıyla okur ve sıralar.

6- Reduce işi yapan işçi süreç bu sıralanmış verideki anahtarlara denk gelen ara değerleri sıralar ve kullanıcı tanımlı reduce fonksiyonuna gönderir.

7- Bütün map ve reduce işleri tamamlandığında MapReduce çağrısı bitmiş olur ve master kullanıcı programını uyandırır.

Hadoop Map-Reduce'da örnek bir Map-Reduce işlemi Java'da şu şekilde yapılır. (White,2011, syf 20-22)

```
public class MaxTemperatureMapper extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private static final int MISSING = 9999;

  public void map(LongWritable key, Text value,
      OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException {

    String line = value.toString();
    String year = line.substring(15, 19);
    int airTemperature;
    if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (airTemperature != MISSING && quality.matches("[01459]")) {
      output.collect(new Text(year), new IntWritable(airTemperature));
    }
  }
}
```

Şekil 4.8: Örnek bir map işlemi

```
public class MaxTemperatureReducer extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
      OutputCollector<Text, IntWritable> output, Reporter reporter)
      throws IOException {

    int maxValue = Integer.MIN_VALUE;
    while (values.hasNext()) {
      maxValue = Math.max(maxValue, values.next().get());
    }
    output.collect(key, new IntWritable(maxValue));
  }
}
```

Şekil 4.9: Örnek bir reduce işlemi

```
public class MaxTemperature {

  public static void main(String[] args) throws IOException {
    if (args.length != 2) {
      System.err.println("Usage: MaxTemperature <input path> <output path>");
      System.exit(-1);
    }

    JobConf conf = new JobConf(MaxTemperature.class);
    conf.setJobName("Max temperature");

    FileInputFormat.addInputPath(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    conf.setMapperClass(MaxTemperatureMapper.class);
    conf.setReducerClass(MaxTemperatureReducer.class);

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

    JobClient.runJob(conf);
  }
}
```

Şekil 4.10: Map-Reduce işinin yapılması

SONUÇ

Bu çalışmada belirli ölçütler açısından dosya sistemleri incelenmiş bazı benzerliklere ve karşıtlıklara rastlanmıştır.

NFS ve AFS dosya sistemleri birbirlerinin muadilleri gibi gözükmektedirler. Ancak

dizaynları arasında büyük farklar mevcuttur. NFS daha ziyade UNIX dosya sistemini dağıtık olarak sağlamaya çalışırken AFS performans ve ölçeklenebilirlik odaklıdır. AFS bazı konulara NFS'ten daha iyi çözümler getirmiş ve daha iyi performans sağlamış olsa bile ticari başarı yakalayamamış ve NFS'in gerisinde kalmıştır.

GFS şu anda mevcut en güçlü DFS gibi gözükmektedir. Belirli bir amaç için özelleşmiştir ve yenilikçi çözümler getirmektedir. Muadili olan HDFS bu mimariyi açık kaynaklı olarak sağlamaya çalışmasına rağmen GFS'in bir kaç sene gerisinde gözükmektedir.

KAYNAKLAR

Arpacı-Dusseau, R., Arpacı-Dusseau, A., 2012. “Operating Systems: Three Easy Pieces”, Arpaci-Dusseau Books, 0.5. Baskı, ABD. ISBN: 9781105979125

Coulourois, G. et al., 2011," Distributed Systems : Concepts and Design", Addison-Wesley, 5. Baskı, ABD.

Dean, J.,Ghemawat, S., 2004. “MapReduce: Simplified Data Processing on Large Clusters”, OSDI, San Fransisco, ABD.

Floyd, R. 1986. Short term file reference patterns in a UNIX environment. Technical Rep. TR 177, Rochester, NY: Dept of Computer Science, University of Rochester.

Ghemawat, S. et al., 2003, “The Google File System”, SOSP, New York, ABD.

ISO, 1995. "Open Distributed Processing Reference Model." International Standard ISOIlEC IS 10746.

Karasulu, B., Körükoğlu, S., 2008, “Modern Dağıtık Dosya Sistemlerinin Yapısal

Karşılaştırılması", Akademik Bilişim, Çanakkale, Türkiye.

Levy, E., Silberschatz, A., 1990, "Distributed File Systems: Concepts and Examples", ACM Computing Surveys, Vol. 22, No.4, syf 321-374.

OpenAFS, 2012 http://wiki.openafs.org/GeneralFAQ/#1%20%20General (Erişim tarihi: 28 Ocak 2013)

Ousterhout, J., Da Costa, H., Harrison, D., Kunze, J., Kupfer, M. And Thompson, J. (1985). A Trace-driven analysis of the UNIX 4.2 BSD file system. In Proc. of the 10th ACM Symposium Operating System Principles, p. 15–24.

Satyanarayanan, M. (1981). A study of file sizes and functional lifetimes. In Proceedings of the 8th ACM Symposium on Operating System Principles, Asilomar, CA, pp. 96–108.

Satyanarayanan, M.,1989, "A Survey of Distributed File Systems", Annual Review of Computer Science, Annual Reviews, Inc., Palo Alto, CA,

Satyanarayanan, M., 1990, "Scalable, Secure, and Highly-Available Distributed File Access" ,IEEE Computer

Shankaran,N. , Sharma,R., 2011, "Cloud Storage Systems - A Survey", http://bit.ly/VvgFw5 (Erişim tarihi: 28 Ocak 2013)

Tannenbaum,S., A., Van Steen, M., 2007, "Distributed Systems : Principle and Paradigms", Pearson Education. Inc., İkinci Baskı, ABD.

Westerlund, A., Danielsson, J., 1998, "Arla- a free AFS client",Usenix, Lousiana, ABD.

Ubuntu ,2012,https://help.ubuntu.com/community/SettingUpNFSHowTo, erişim

tarihi 28.01.2013

Wikipedia, 2011, http://en.wikipedia.org/wiki/Distributed_hash_table (Erişim tarihi: 28 Ocak 2013)

White, T., 2011, ”Hadoop: A Definitive Guide”, O'Reilly Media Inc. 2.Baskı, ABD.

ÖZGEÇMİŞ

1991 yılında İstanbul'da doğdu. İlk ve orta öğretimini İstanbul'da tamamladı. 2009 yılında başladığı Kocaeli Üniversitesi Bilgisayar Mühendisliği bölümündeki lisans öğrenimini hala sürdürmektedir.