Yüksek Lisans Tezi
Trakya Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Bölümü

ÖZET

Kümeler yüksek hesaplama gücü, yük dengeleme ve hata toleransı sağlayan ve düğüm adı verilen bilgisayarlardan oluşan paralel işlem yapan süper bilgisayarlardır. Küme yapılarını oluşturan düğümler için özel sunucu veya iş istasyonu gibi yüksek maliyetli bileşenler kullanılması zorunlu değildir. Kümelerin amacı sunulmakta olunan servislerin iyileştirilmesidir.

Sunucu kümeleri küreselleşme nedeniyle yaşanan acımasız rekabet ortamında şirketlerin bilgi işlem altyapıları için çok önemlidir. Günümüzde yaşanan acımasız rekabet ortamında en büyük hedef kesintisiz servisler sunabilmektir. Kesintisiz servis sunabilmek şirketin rakiplerine göre avantajlı duruma geçmesini sağlayacaktır.

Linux Virtual Server sistemi hata toleranslı, yük dengelemeli ve yüksek erişilebilir küme yapıları oluşturmak için kullanılan bir yazılım grubudur. Tez kapsamında bir yönetici, iki adet sunucu düğümü ve bir adet veritabanı sunucusundan oluşan küme yapısı oluşturulmuştur. Kümede veritabanı sunucusu olarak MySQL ve web sunucusu olarak Apache kullanılmıştır. Apache sunucu üzerine farklı sayfalar ve bir alışveriş sistemi kurularak performans ve hata toleransı testleri gerçekleştirilmiştir.

Tezin tamamı yedi bölümden oluşmaktadır.

Tezin birinci bölümünde kümeleme, kümelemenin amacı ve önemi ele alınmıştır.

Tezin ikinci bölümünde kümeleme konusundaki temel kavramlara değinilmiştir. Ayrıca kümelemenin gereksinimleri, küme bileşenlerinin seçimi ve küme düğümlerinin yapısı detaylı olarak incelenmiştir.

Tezin üçüncü bölümünde Linux işletim sistemi, kümeleme yapısında Linux işletim sisteminin önemi, Linux işletim sistemi çekirdek yapısı, Linux işletim sistemi dosya sistemleri, önemli INIT scriptleri, kümeleme için işletim sistemine ince ayar yapılması ve işletim sistemi optimizasyon konusu incelenmiştir.

Tezin dördüncü bölümünde Linux kümeleri, Linux kümelerinin diğer kümelere göre avantajları, Linux işletim sisteminde paketlerin işlenmesi, Linux Virtual Server, LVS-NAT kümeleri, LVS-DR kümeleri, LVS-TUN kümeleri, yük dengeleyicinin çalışma prensipleri ve yüksek erişilebilir küme yapılarının tasarımı konuları incelenmiştir.

Tezin beşinci bölümünde hata toleranslı uygulamalar ve Linux kümelerinin kullanılabileceği hata toleranslı uygulamalar ele alınmıştır.

Tezin altıncı bölümünde tez kapsamındaki küme uygulaması, Linux kümesini meydana getiren bileşenler, Linux hata toleranslı yük dengelemeli küme yapısının kurulması, MySQL veritabanı sunucusunun kurulumu ve ayarları ve rsync ile düğümlerdeki içeriklerin senkronizasyonu ele alınmıştır.

Tezin son bölümü olan yedinci bölümünde tez kapsamında kurulumuş olan kümeleme yapısında gerçekleştirilmiş olan performans testleri ele alınmıştır.

Anahtar Kelimeler: Linux Kümeleri, Hata Toleransı, Yük Dengeleme, Yüksek Erişilebilirlik, LVS-NAT, LVS-DR, LVS-TUN.

Yıl: 2008
Sayfa: 193

Master Thesis
Trakya University Graduate School of
Natural and Applied Sciences
Department of Computer Engineering

ABSTRACT

Clusters are parallel processing super computers,which provide high computing power, load balancing and fault tolerance, and composed of computers called nodes. It is not required to use high cost components like special servers or workstations for the nodes forming clusters. The aim of clusters is to improve the services currently offered.

Clusters are very important for companies' information system infrastructures in the merciless competition environment due to the globalization. Nowadays, in the merciless competitive environment, the biggest aim is to offer uninterrupted services. Being able to offer uninterrupted services will make the company advantageous against its competitors.

Linux Virtual Server system is a group of softwares to build fault tolerant, load balancing and high available cluster structures. Within the context of the thesis, a cluster composed of one director, two server nodes and one database server have been set up. In the cluster, MySQL server as the database server and Apache server as the web server has been used. Different web pages and one shopping system have been loaded on the Apache server and performance and fault tolerancy tests have been conducted.

The thesis is composed of seven sections.

In the first section of the thesis, clustering and the aim and the importance of clustering are addressed.

In the second section of the thesis, the main concepts about clustering are mentioned. Moreover, the requirements of clustering, the selection of cluster components and the structure of cluster nodes are examined in detail.

In the third section of the thesis, the Linux operating system, the importance of the Linux operating system in clustering, the Linux operating system kernel structure, the Linux operating system file systems, important INIT scripts, fine tuning of operating system for clustering and operating system optimization are examined.

In the fourth section of the thesis, Linux clusters, the advantages of Linux clusters against other clusters, packet processing in Linux operating system, Linux Virtual Server, LVS-NAT clusters, LVS-DR clusters, LVS-TUN clusters, working principles of load balancer, and the design of high available cluster structures are examined.

In the fifth section of the thesis, fault tolerant applications and the fault tolerant applications that Linux clusters can be used for are addressed.

In the sixth section, the cluster application in the scope of the thesis, the components which build the Linux cluster, the installation of Linux fault tolerant load balancing cluster structure, the installation of MySQL database server and the synchronization of nodes' contents with rsync are addressed.

In the seventh section, which is the last section, the performance tests that were conducted in the cluster which was built for the thesis are addressed.

Keywords: Linux Clusters, Fault Tolerance, Load Balancing, High Availability, LVS-NAT, LVS-DR, LVS-TUN.

Year: 2008
Page: 193

TEŞEKKÜR

Bu tez için gerçekleştirdiğim çalışma sırasında bana yardımlarından, katkılarından ve sağladığı mükemmel çalışma ortamından dolayı değerli hocam ve danışmanım Sayın Yrd. Doç. Dr. Erdem UÇAR'a sonsuz teşekkürlerimi sunmak isterim.

Tez jürimde yer alan hocalarım Sayın Yrd. Doç. Dr. Yılmaz KILIÇASLAN ve Sayın Yrd. Doç. Dr. M. Tahir ALTINBALIK'a tez çalışmama yaptıkları olumlu eleştiriler ile katkıda bulundukları için sonsuz teşekkürlerimi sunarım.

Ayrıca çalışmam sırasında yardımcı olan ULAKBİM personeli Murat SOYSAL'a ve Trakya Üniversitesi Bilgi İşlem Daire Başkanlığında görevli Bilgisayar İşletmeni Özcan GÜLBABA'ya teşekkür ederim.

Son olarak bana yardımlarını esirgemeyen ve bana sonsuz destekleri için eşim Ayşe TUNA'ya ve aileme teşekkür etmeyi bir borç bilirim.

Mayıs 2008 Gürkan TUNA

BÖLÜM 1

1. GİRİŞ

Günümüzde yaşanan acımasız rekabet ortamında en büyük hedef her alanda kesintisiz servisler sunabilmektir. Kesintisiz servis sunabilmek şirketin rakiplerine göre avantajlı duruma geçmesini sağlayacaktır. Bu durum tüm sektörler için geçerlidir. Her sektörde kesintisiz servis sunabilmek şirketler için bir ayrıcalıktır.

Bilgi İşlem gelişmekte olan bir sektör olduğu gibi rekabetin en yoğun yaşandığı sektörlerden de birisidir. Bilgi İşlem sektöründe farklı firmalar farklı alanlarda çalışmaktadır. Internetin yaygınlaşmasıyla beraber özellikle web tabanlı uygulamalar hızla yaygınlaşmaktadır. Hosting firmaları sunucu barındırma ve yayınlanama amacıyla çalışan firmalardır. Hosting firmaları reklamlarında ve sitelerinde %99.9 civarında bir hizmet garantisi sunmaktadırlar. Sunulan hizmet garantisi bu firmaların kalitesini belirleyen unsurlardan birisidir.

Sunucu kümeleri küreselleşme nedeniyle yaşanan acımasız rekabet ortamında şirketlerin bilgi işlem altyapıları için çok önemlidir. Kümeler yüksek hesaplama gücü, yük dengeleme ve hata toleransı sağlayan ve düğüm adı verilen bilgisayarlardan oluşan paralel işlem yapan süper bilgisayarlardır. Kümelerin amacı sunulmakta olunan servislerin iyileştirilmesidir. Küme yapıları yüksek erişilebilirlik sunan tüm firmalar için önemlidir.

Linux Virtual Server sistemi hata toleranslı, yük dengelemeli ve yüksek erişilebilir küme yapıları oluşturmak için kullanılan bir yazılım grubudur. Tez kapsamında bir yönetici, iki adet sunucu düğümü ve bir adet veritabanı sunucusundan oluşan küme yapısı oluşturulmuştur. Kümede veritabanı sunucusu olarak MySQL ve

web sunucusu olarak Apache kullanılmıştır. Apache sunucu üzerine farklı sayfalar ve bir alışveriş sistemi kurularak performans ve hata toleransı testleri gerçekleştirilmiştir.

Tez çalışmasına başlarken amaç küme yapılarının diğer sistemlere göre avantajını göstermekti. Küme yapısı kurulduktan sonra gerekli testler yapılmış ve yapının avantajları ortaya konmuştur. Küme yapılarının avantajlarını göstermek tezin tek amacı değildi. Bir diğer amaçta küme yapılarının kullanılabileceği farklı uygulamaları göstermek ve bu uygulamalarda küme yapılarının önemini ortaya koymaktı. Internetin gelişimi ve web tabanlı uygulamaların yaygınlaşması nedeniyle özellikle web tabanlı uygulamalar üzerinde durulmuş ve küme yapılarının web uygulamalarında önemi ortaya konmuştur. Küme yapıları alternatif sistemlere göre büyük avantajlara sahip olan çözümlerdir ve çok farklı uygulamalarda farklı amaçlarla kullanılabilirler.

BÖLÜM 2

2. TEMEL KAVRAMLAR

2.1. Küme Nedir?

Kümeler yüksek hesaplama gücü, yük dengeleme ve hata toleransı sağlayan ve düğüm adı verilen bilgisayarlardan oluşan paralel işlem yapan süper bilgisayarlardır. Küme yapılarını oluşturan düğümler için özel sunucu veya iş istasyonu gibi yüksek maliyetli bileşenler kullanılması zorunlu değildir. Kümelerin amacı sunulmakta olunan servislerin iyileştirilmesidir.

Düğümlerin her birinde bir veya daha fazla işlemci, bellek ve diğer yardımcı birimler bulunur. Düğümler birbirine ağ ile bağlıdır. Böylece düğümler arasında veri alışverişi mümkündür. Düğümler genellikle kişisel bilgisayarlar veya düşük maliyetli sunuculardır. Eğer bir düğümde birden fazla işlemci varsa bu düğüm simetrik çoklu işlemcili düğüm olarak isimlendirilir [1].

Düğümleri birbirine bağlayan ağ alternatifleri bulunmaktadır. Genellikle Ethernet tabanlı ağlar kullanılmaktadır. Ağı oluşturmada kullanılan switch veya hub düğümler arasında iletişimi sağlar.

Neden tek bir bilgisayar yerine düğüm kullanılır? Bu sorunun temelde iki cevabı vardır. Birincisi performans, ikincisi ise hata toleransıdır. Kümelemenin ilk geliştirilme amacı bilimsel uygulamalar için gerekli olan düşük maliyetli hesaplama gücünü sağlamaktı. Yani tek işlemcili veya çoklu işlemcili bir bilgisayarının sağlayabileceğinden fazla performans gerektiren uygulamalar için düşünülmüştü. Bir

uygulama çok farklı nedenler için hesaplama gücü gerektirebilir. Bu nedenlerden en yaygın üçü şunlardır:

- *Gerçek zaman sınırlaması:* Uygulamanın hesaplamayı belirli bir zaman dilimi içinde bitirmesi zorunluluğudur.
- *Üretilen iş:* Bilimsel bir simülasyon veya bir mühendislik simülasyonu birçok hesaplama gerektirebilir. Küme birbirine bağlı birçok simülasyonu gerçekleştirmek için gerekli olan kaynakları sağlayabilir. Diğer taraftan bazen tek bir simülasyon öyle büyük bir hesaplama gücü gerektirebilir ki, bu işi tamamlamak için tek bir işlemcinin bazen günlerce bazen yıllarca çalışması gerekir. Kümelemeye mükemmel bir örnek olarak Google verilebilir. Google'da 15.000'den fazla kişisel bilgisayar hata toleransı sağlayan bir yazılım ile bağlanmıştır ve yüksek performanslı web tabanlı arama servisi hizmeti vermektedir.
- *Bellek:* Bazı uygulamalar simülasyonun bir parçası olarak çok büyük boyutlarda veri kullanabilmektedir. Kümeleme uygulama için terabaytlara kadar ulaşan bellek sağlamaktadır.

Kümeler gerekli olan hesaplama gücünü paralel programlama ile sağlarlar. Paralel programlama birçok işlemcinin aynı anda kullanımının koordinasyonunu sağlayan tekniktir. Kümeleme kullanmanın diğer bir nedeni hata toleransı sağlamaktır. Böylece gerekli olan hesaplama gücünün daima olacağı garanti edilebilir. Çünkü kümeler aynı veya benzer bileşenlere sahip düğümlerden meydana gelir. Kümeyi oluşturan düğümlerden birinde meydana gelecek sorun sadece kümenin hesaplama gücünü azaltır. Bu nedenle kümeler veri toplamada kullanılan web sunucular ve sistemler için ideal çözümdür. İyi tasarlanmış bir küme teorik olarak %100 çalışabilirlik süresi garantisi sağlar.

2.2. Uygulama Gereksinimleri

2.2.1. Hesaplama gereksinimleri

Özellikle bilimsel ve teknik uygulamalardaki en belirgin gereksinim hesaplamayı gerçekleştirebilmek için gereksinim duyulan kayan noktalı işlemlerin sayısıdır. Basit hesaplamalar için bu numarayı tahmin edebilmek göreceli olarak kolaydır. Karmaşık işlemlerde bile kabaca bir tahmin yapabilmek genellikle mümkündür. Nümerik analiz üzerine olan kitaplar kullanılarak tahmin yapmak mümkün olacaktır.

Hesaplama gereksinimlerine bir örnek şöyle verebiliriz. 2 GHz bir işlemci saniyede $2x10^9$ kayan noktalı işlem yapabilir. Bu işlemcide 1 milyar kayan noktalı işlem gerektiren bir hesaplama sadece yarım saniye alacaktır [1]. Ancak hesaplama gereksinimlerine bakarken elde edilebilecek genel sistem performansı göz önüne alınmalıdır. LINPACK ve STREAM gibi standart karşılaştırmalı değerlendirme testleri kullanılabilir.

Bellek: Uygulamanın bellek ihtiyaçları uygulamanın performansını ve kümenin maliyetini önemli ölçüde etkiler. Düğümlerde farklı bellek türleri bulunur. Ana bellek uygulama tarafından ihtiyaç duyulan tüm veriyi saklayabilecek büyüklükte olmalıdır. Küme yapısında bellek farklı düğümlere dağıtılmıştır ve düğümlerdeki bellek alanları tek bir alan olarak düşünülür. Önbellek daha küçük fakat daha hızlı bir bellektir. Önbellek uygulamaların performansını iyileştirmek için kullanılır. Bazı uygulamaların performansı için önbellek çok önemlidir. Sanal belleklerde disk alanının bir kısmı bellek olarak kullanılır. Sanal bellek alanının tamamına uygulama tarafından erişilebilir. Sanal bellekler düşük maliyetle bellek kapasitesini arttırırlar. Ancak veriler disk üzerinde saklandığı için erişim süresi ana belleğe göre çok yüksektir. Bu nedenle yüksek performanslı kümelerde sanal bellekler kullanılmaz.

Giriş / Çıkış: İşlemlerin sonuçları disk üzerinde bulunan dosyalar gibi kalıcı depolama alanlarında saklanmalıdır. Paralel çalışma işlemlerin çok hızlı yapılmasını sağlar. Bununla orantılı şekilde giriş / çıkış sistemi performansı beklenir. Literatürde

kısaca NFS olarak geçen ağ dosya sistemi kümede bulunan her düğümün herhangi bir dosyaya erişebilmesini sağlar. Ancak NFS hem yüksek performans sağlamaz hem de aynı dosyaya aynı anda erişirken sorunlar yaratır. Linux sistemlerde kullanılabilecek yüksek performanslı paralel dosya sistemleri bulunmaktadır.

2.2.2. Diğer gereksinimler

Kümelerin iyi bir ağ gibi diğer gereksinimleri de olabilir. Yedekleme sistemleri gibi sistemlerde kümelerde kullanılabilir.

Paralellik: Paralel uygulamalar 2 türe ayrılabilir. 1. türdeki uygulamalar yapısı gereği paralel olan uygulamalardır. Bu uygulamalar bağımsız olarak çalıştırılabilecek daha küçük görevlere kolaylıkla bölünebilecek uygulamalardır. Bir web sunucuyu göz önüne alırsak web sunucuda saklanan bilgiyi talep eden her istek birbirinden bağımsızdır. Bu tür uygulamalar kolaylıkla küme yapısına taşınabilir.

2. türdeki uygulamalar birbirinden bağımsız alt görevlere ayrılamayan uygulamalardan oluşur. Bu tür uygulamalar küme yapısına taşınırken detaylı inceleme gerekir.

2.3. Küme Bileşenlerinin Seçilmesi

Kümede kullanılacak bileşenler seçilirken, kümede çalışacak olan uygulamalar göz önüne alınmalıdır. Küme yapısı oluşturulurken aşağıdaki maddelere detaylı olarak bakılmalıdır.

- Uygulamanın ihtiyaçları anlaşılmalıdır.

- Kümede kullanılacak düğüm sayısı ve tipine karar verilmelidir. Uygulama ihtiyaçlarına bakılarak, düğüm tipi, işlemci tipi ve bellek belirlenmelidir. Sadece CPU hızı performansı belirlemez.
- Ağ yapısına karar verilmelidir. Uygulamaların ağda düşük gecikme ve / veya yüksek bant genişliği ihtiyaçlarına bakılmalıdır. Bazı küme yapılarında yüksek maliyetli yüksek performanslı ağ, bazılarında ise düşük maliyetli Ethernet ağları yeterli olacaktır.
- Fiziksel altyapı gereksinimlerine karar verilmelidir. Ne kadar alan gereklidir? Nasıl bir güç kaynağı ve nasıl bir soğutma sistemi kullanılmalıdır?
- Kullanılacak işletim sistemine karar verilmelidir. Küme yapılarının çoğunda Linux tabanlı işletim sistemleri kullanılmaktadır. İşletim sistemi seçiminde aşağıdaki maddeler göz önüne alınmalıdır.
  - Seçilen işletim sisteminde kullanılacak uygulamalar çalışabilir mi?
  - Seçilen işletim sisteminde yeterli deneyiminiz var mı?
  - İşletim sistemi ve diğer kullanılacak yazılımların lisanslama modeli nedir?
- Maliyet unsurlarına bakılmalıdır. Küme alınabilecek en hızlı düğümlerden mi oluşmalıdır, yoksa daha düşük maliyetli düğümler yeterli olacak mıdır? Eğer yeterli bütçe var ise küme alınabilecek en hızlı düğümlerden oluşturulabilir. Diğer bir yöntem ise düğüm sayısını arttırarak daha düşük maliyetli düğümler almaktır.

2.4. Küme Düğümlerinin Yapısına Detaylı Bakış

Kümedeki her düğüm uygulama programının çalıştırılması ile ilgili tüm işlemlerden ve karmaşık yazılım ortamının desteklenmesinden sorumludur. Bu süreç birçok bileşenden oluşur. Uygulama aslında merkezi işlem biriminde çalıştırılır. Merkezi işlem birimi önbellek ve ana bellekten yazmaçlara veri yükler. Bütün uygulamalar çevre birimlerini kullanır. Tüm çevre birimleri ana belleğe bilgi yükler veya ana bellekteki bilgiyi işler.

Kodun çoğu sistemdeki merkezi işlem biriminde çalışır. Düğümde birden fazla işlemci bulunabilir. Bu tür düğümler simetrik çoklu işlem modundadır. İşlemcide bir miktar önbellekte bulunur. Önbellek ana bellekten ortalama olarak on kat daha hızlıdır. Dolayısıyla verinin önce önbelleğe aktarılması ve oradan çağrılması daha avantajlıdır. Ana bellek programların çalıştığı yerdir. İşletim sistemini içerir ve tüm veriyi saklar. Ana bellek kalıcı bellek olmadığı için daha sonra ulaşılması gereken bilgiler sabit disk gibi kalıcı medyalarda saklanmalıdır. Giriş / Çıkış yolu ana belleği çevre birimlerine bağlar. Disk kontrolcüsü, ağ kartları ve ekran kartları gibi çevre birimleri ana bellekteki veriyi işleyerek çalışırlar.

Bir uygulama çalıştığında öncelikle sabit diskten ya da başka bir kalıcı bellekten ana belleğe yüklenir. Çalışma başladığında uygulamanın bölümleri işlemci önbelleğine kopyalanır. Buradan veri işlemci üzerindeki yazmaçlara yazılır ve bu veriye işlemci doğrudan ulaşabilir. İşlemcinin bu veriyle işi tamamlandığında bu veri ana belleğe geri yazılır. Uygulama çevre birimlerinde bulunan veriye bağlı olduğunda veriyi yazmaçlara yüklemek daha karmaşık hale gelmektedir.

Mikroişlemci: Mikroişlemci her bilgisayarın en önemli bileşenidir. Komutların çalıştırılmasını sağlayan tek bileşendir. Farklı mikroişlemcilerin farklı karakteristik özellikleri vardır. Komutların en düşük seviyede ikili kodlanması ve gerçekleştirdiği işlemler ISA adı verilen mikroişlemci komut set mimarisi tarafından sağlanır. Küme düğümlerindeki işlemcilerde en yaygın kullanılan ISA IA32 veya X86'dır. Bu işlemci ailesi Pentium ve AMD Athlon işlemci ailelerini kapsar. Paylaşılan ISA mimarisinde aynı komut seti zorunluluğu yoktur. Yeni işlemcilerde eski işlemcilerde bulunmayan ekstra özellikler bulunur.

İşlemci belirli bir saat hızında çalışır. Megahertz veya gigahertz olarak hesaplanan belirli bir frekansta komutları çalıştırabilirler. Örneğin 2 GHz hızında çalışan bir işlemci saniyede 2 milyar komut çalıştırabilir. Performansı sadece işlemcinin hızı belirlemez. Bazı görevleri farklı saat hızlarında çalışan işlemciler yakın performanslarla yerine getirebilirler. Bazı görevlerde ise büyük performans farkları oluşur [1].

Her işlemcinin ulaşabileceği teorik bir zirve hızı vardır. Teorik zirve hızı bir işlemcinin ulaşabileceği maksimum komut çalıştırma oranıdır. Bu değer işlemci saat hızına, ISA mimarisine ve işlemcinin yapısındaki bileşenlere bağlıdır. Bu oran saniyedeki kayan nokta işlemleri veya flops'lar ile ölçülür. Gerçek uygulamalarda bu teorik hıza ulaşmak çok zordur [1].

Hem komutlar hem de veri düğümdeki rastgele erişimli bellekte saklanır veya bu bellekten yüklenir. İşlemci hızlı sıklıkla megahertz veya gigahertz olarak ölçülür. Rastgele erişimli bellek genellikle daha düşük saat hızlarında çalışır ve genellikle megahertz olarak ölçülür. Dolayısıyla işlemci genellikle belleği bekler ve programların genel çalışma hızları işlemci hızı kadar bellek tarafından da belirlenir. Bu problemin çözümü için işlemci önbelleği kullanılır. Önbellek işlemci üzerinde bulunan daha hızlı küçük bir bellektir. Veri ana bellekten kopyalandığında önbellekte de saklanır. Eğer aynı veriye tekrar ulaşmak gerekirse önbellekten çağrılabilir.

Bellek: Sistemde bulunan rastgele erişimli bellek komutları ve veriyi geçici olarak saklamak için kullanılan birimdir. Komutlar işlemcinin çalıştırdığı gerçek işlemlerdir. Bellekte saklanan veri farklı kaynaklardan gelebilir. Sabit disk veya ağ kartı gibi çevre birimlerinden gelebilir veya programın çalışması sırasında oluşan ara sonuçlardan olabilir. Hem komutlar hem de veri işlemcinin doğru sonuçlar oluşturması için gereklidir. Dolayısıyla sürekli olarak işlemci bellek yolunu kullanarak bellekten veri yükler veya bellekte veri saklar. Bellek yolları genellikle 100 – 800 MHz arası hızlarda çalışır. Bu yol FSB olarak adlandırılır.

Sistemdeki belleğin sürekli olarak kullanılması ve işlemci saat hızı ile bellek yol hızı arasındaki büyük uçurumdan dolayı teorik zirve hızına ulaşmanın önündeki en büyük engel bellek yoludur. Bellek yol performansı iki karakteristik kullanılarak ölçülür. Birincisi bellek bant genişliğinin zirve hızıdır. İkinci karakteristik belleğin gecikme değeri olup, RAM bellek ve CPU arasında veri aktarırken geçen zamandır. RAM bellek bant genişliği değerleri saniyede 1-4 gigabyte arasında değişir. RAM bellek gecikme değerleri 6 nanosaniyenin altına inmiştir [1].

Çok dikkatli olarak tasarlanmış uygulamalar haricinde genellikle programın tüm veri seti RAM bellekte bulunur. Alternatif yöntemler saklama aracı olarak diski

kullanmak veya sanal bellek kullanmaktır. Bu iki yöntem de sistem performansını çok düşürür.

Giriş / Çıkış Kanalları: Giriş / Çıkış kanalları çevre birimleri ile ana belleği birbirine bağlayan yollardır. Çevre birimleri sabit disk, ağ kartı, video kontrolcüsü, USB veya firewire olabilir. Bilgisayarlarda bu yollardan her biri belleğe köprü adı verilen bileşenlerle bağlanır. Giriş / Çıkış birimleri bilgisayarlarda bulunan en önemli bileşenlerden biridir. Giriş / Çıkış kanallarına örnek olarak PCI, PCI-X, AGP verilebilir. PCI veri yolları 1994 yılından beri vardır. İlk versiyonları 32 bitlik olup 33 MHz veri yollarıydı. Bu yollar teorik olarak 132 MB/s hızına ulaşabilirdi. PC veri yolunun yeni versiyonları 64 bitlik yollar olup 66 MHz veya daha yüksek hızlarda çalışırlar. Bu yollar teorik olarak 500 MB/s hızını destekleyebilir. AGP yüksek hızlı grafik kartları için tasarlanmış bir port olup PCI ve PCI-X in destekleyemediği daha yüksek hızlarda çalışır. AGP PCI'ın aksine bir veri yolu değildir. AGP sadece bir cihazı destekler. AGP cihazları bellekten direkt okuma yapabilir. AGP 2.0 versiyonu 1 GB/s hızını, AGP 3.0 ise 2.1 GB/s hızını destekler.

Anakart: Anakart baskı devre tablası olup PC de bulunan aktif elektronik bileşenlerin çoğunu ve bunların bağlantılarını barındırır. Anakart küme düğümünde bulanan alt sistemlerin entegrasyonu için gereken mantıksal ve fiziksel altyapıyı sağlar. Anakart düğümün işlevselliğini, kullanılabilecek performans sınırını, maksimum saklama kapasitesini ve birbirine bağlanabilecek alt sistemlerin sayısını belirler. Mikroişlemci seçiminin dışında PC'yi oluştururken verilmesi gereken en önemli diğer karar anakart seçimidir.

Anakart işlevselliği sağladığı gibi kısıtlamalara da neden olur. Küme düğümünde kullanılacak anakart seçimi yapılırken aşağıdaki gereksinimlere dikkat edilmelidir.

- İşlemci ailesi
- İşlemci saat hızı
- İşlemci adedi
- Bellek kapasitesi
- Bellek tipi

- Disk arabirimi
- Gereken giriş/çıkış yuvaları sayısı
- Giriş / çıkış yolları sayısı ve tipi

Kalıcı depolama birimleri: Sisteme gelen güç kesildiğinde BIOS kodu ve konfigürasyonu haricinde bellekte saklanan tüm bilgiler kaybolur. Veriyi sürekli olarak saklamak için geçici olmayan kalıcı depolama birimleri gereklidir. Genellikle uygulamanın ihtiyaç duymadığı veriler disk üzerinde saklanır. İhtiyaç olduğunda diskten yüklenir. Kümelerin çoğunda her bir düğümde depolama için bir sabit disk gereklidir. Sabit disk bir manyetik saklama birimi olup bir veri yolu üzerinden sisteme bağlıdır. Sabit disklerin çoğunda performans artışı sağlamak için önbellek bulunur. Sabit diskler farklı arabirimlerden anakarta bağlanırlar. IDE, SCSI, SATA gibi farklı arabirimlere sahip sabit diskler bulunur. Yeni IDE diskler 133 MB/s civarı transfer hızlarını desteklerler. SCSI arabirime sahip diskler genellikle sunucularda kullanılır. Maliyetleri yüksektir. Performansları IDE disklerden daha yüksektir. SCSI diskler 320 MB/s civarı transfer hızlarını destekler. Yeni arabirimlerde bu hızın daha da üstüne çıkılmıştır. SATA diskler yeni standartlardan birisidir. SATA diskler kişisel bilgisayarlarda kullanılan IDE disklerin yerini almaktadır. Genellikle 150 MB/s transfer hızlarını desteklerler.

Disk performansını belirleyen sadece arabirimi değildir. Disk dönme hızı ve önbellek kapasitesi de disk performansını belirleyen diğer unsurlardır. Disk dönme hızları 5400-15000 RPM arasında değişmektedir. RPM dakikadaki dönme sayısını gösteren birimdir. Yüksek dönme sayısı yüksek performans anlamına gelir. Yüksek dönme sayısı diskten okuma süresini ve gecikmeyi azaltır.

RAID birbirinden bağımsız disklerin performansı ya da veri güvenliğini arttırması için beraber kullanılmasıdır. Genellikle veri güvenliğini artırmak için kullanılır. Yazılım veya donanım olarak RAID desteği sağlayan ürünler bulunmaktadır. Yüksek performans ve kararlılık beklenen uygulamalarda donanım tabanlı RAID ürünleri tercih edilir. Donanım tabanlı RAID ürünlerinde sistem kaynakları kullanılmaz. Yazılım tabanlı RAID desteği sağlayan ürünler sistem kaynaklarını çok kullanırlar.

RAID 0 en az iki disk ile oluşturulan ve performans artışı sağlayan bir yapıdır. Bilgi bloğu disklere paylaştırılarak yazıldığından yazma ve okuma hız performansı artmaktadır. Ancak bu durumda veri güvenliği bulunmamaktadır. Disklerden biri arızalandığında bütün veriler kaybolacak dolayısı ile sistem kapanacaktır. Kapasite, disklerin toplamıdır. Aynı yapıdaki disklerin kullanılması tercih edilmelidir. Ancak farklı diskler kullanıldığında en küçük diskin kapasitesi dikkate alınır. Daha çok CAD/CAM gibi grafik uygulamaları için kullanılmaktadır.

RAID 1 ile bilgi blokları iki diske birden yazılırlar. Burada en az iki disk kullanılabilir. Böylece birbirinin kopyası olan diskler oluşur. Kapasite tek bir disk kapasitesidir. Farklı disk kapasitedeki disklerde en küçük kapasiteli disk referans alınacaktır. Herhangi bir disk arızası durumunda ikinci disk görevi üstlenerek sistemin çalışmasını sağlıyor. Böylece iş akışı durmamış oluyor. Arızalı disk sistem çalışırken çıkartılıp yerine sağlam disk takılır, sistem konfigürasyonu eski haline getirilir. Disk okuma hızı artarken yazma hızı ise yavaş olmaktadır. Disk güvenliğinin en üst seviyede olduğu durumlarda kullanılır.

RAID 5 hem hızın hem güvenliğin beraber oluşturulduğu bir yapıdır. En az 3 disk gereklidir. Yandaki örnekte 3 disk RAID 5 oluşturulmuştur. Bir algoritma ile bilgiler disklere sırası ile yazılırken her defasında bir diske yazılan bilgilerin algoritması kaydedilir. RAID 5 sisteminde herhangi bir diskin arızalanması durumunda sistem çalışmaya devam eder. Arızalı diskin sistem kapanmadan değiştirilebilir ve RAID 5 yapısının tekrar oluşturulması sağlanır. Burada tek bir disk güvenlik için kullanılmaktadır. Toplam kapasite bir eksik olacaktır.

RAID 1/0 ise iki ayrı RAID 1 kümesinin RAID 0 ile birleştirilmesi durumudur. Toplamda 4 disk kullanılmaktadır. Sistem performansı ve güvenliği yüksektir. Buna karşılık iki adet disk veri güvenliği için kullanılmaktadır. Kapasite azalmaktadır.

Ekran kartı: Ekran kartı bilgisayarların en önemli bileşenlerinden birisidir. Çünkü kullanıcıya direkt olarak hitap edilmesini sağlar. Bilgisayarın ekran kartı görüntünün monitöre aktarılmasını sağlar. Yeni ekran kartlarının çoğu sisteme AGP portundan bağlanır. Daha eski ekran kartları sisteme PCI slotlarından bağlanmaktaydı. PCI slotları yoğun grafik işlemleri için gerekli olan bant genişliğini sağlayamamaktaydı.

Kişisel bilgisayarlarda bulunan ekran kartlarının 3D özelliği oyunlarda, grafik uygulamalarda önem kazanmaktadır. Küme yapısını oluşturan düğümlerde ekran kartının belleği, işlemcisi ve diğer özellikleri çok önemli değildir. Kurulumdan sonra sadece donanım kaynaklı problemlerin nedenlerini anlamak ve BIOS işlemlerinde kullanılmaktadır.

Çevre Birimleri: Çevre birimleri küme yapılarındaki önemli bileşenlerden değildir. USB ve Firewire çevre birimlerinin kullandığı yollardır. Aygıtlar bu yollara bağlanabilir. USB klavye, mouse, kamera, yazıcı çevre birimlerine örnek verilebilir. Eski bilgisayarlarda seri ve paralel portlarda bulunmaktadır. Ayrıca eski bilgisayarlarda klavye ve mouse gibi bileşenler sisteme özel klavye ve mouse portlarından bağlanmaktadır.

Kasa: Küme yapısında kasa kişisel bilgisayarlardan daha önemlidir. Kasa seçimi direkt olarak kümenin kaplayacağı alanı etkilemektedir. Desktop, Mini tower, Midi tower ve Rack tipi kasalar bulunmaktadır. Sistem odasında kümenin kabinet içinde olması planlanıyorsa rack tipi kasalar seçilmelidir. Böylece yerden kazanılabilir. Rack tipi kasaların dezavantajı genişleme olanaklarının sınırlı olmasıdır.

Ağ Kartı: Düğümlerin ağa bağlanmasını sağlayan bileşen olup, küme yapısında çok önemli bir yere sahiptir.

2.5. Kümelerde Ağ Yapıları

Farklı küme yapılandırmaları bulunmasına ve bunların her birinin kendisine has karakteristik özellikleri olmasına rağmen bütün küme yapılarında en önemli bileşen ağ dır. Ağ kümedeki düğümlerin birbirine bağlanmasını sağlar. Ağ yapısında aktif / pasif cihazların yanında önemli bileşenlerden diğeri de kablodur. Fiber optik, UTP kablo, STP kablo gibi kablo çeşitleri vardır. Fiberoptik kabloların multi-mode, single-mode gibi çeşitleri bulunmaktadır. Single-mode fiber optik kablo genelde uzun mesafelerde

kullanılır. Kısa mesafelerde Multi-mode fiber optik kablo kullanılır. 62,5/125 mikron multi-mode fiber optik kabloda 1000Mbps için iletişim mesafesi 260m'dir. 50/125 mikron multi-mode kabloda 1000Mbps için iletişim mesafesi 550m civarındadır. 9/125 mikron single-mode fiber optik kabloda 1000Mbps için iletişim mesafesi 5km civarındadır.

UTP kablonun da Kategori 5, Kategori 5E, Kategori 6 gibi türleri bulunmaktadır. 100Mbps hızında Kategori 5 ve Kategori 5E UTP kablolar kullanılabilir. Gigabit Ethernet için Kategori 5E veya Kategori 6 kablo tercih edilmelidir.

Switch ağ yapısında önemli diğer bir bileşendir. Switch ağ performansını önemli derecede etkiler. Farklı port sayısına, port tiplerine ve şasi özelliklerine sahip switchler bulunmaktadır.

BÖLÜM 3

3. LINUX İŞLETİM SİSTEMİ

Linux Unix işletim sistemi temel alınarak geliştirilmiş bir işletim sistemidir. Linus Torvalds tarafından geliştirilmiş ve 1991 yılında duyurulmuştur. Linux açık kaynak kodlu bir işletim sistemi olup dünyanın her tarafından yazılım geliştiricilerin katkılarıyla sürekli olarak gelişmekte, değişmekte ve iyileşmektedir. Şu anda Linux işletim sisteminin birçok türevi bulunmaktadır.

3.1. Küme Yapısı İçin Neden Linux Kullanılmalıdır?

Linux dünyadaki açık kaynak kodlu işletim sistemleri içinde en popüler olanıdır. Başarısının sırrı tek bir nedene bağlı değildir. Kararlı olması, Unix işletim sistemini temel alması nedeniyle gelişim sürecini tamamlamış olması ve güvenilirliği Linux işletim sisteminin avantajları arasındadır. Bunun yanında çekirdek kaynak kodunun açık ve ücretsiz olması nedeniyle IBM, Fujitsu, NEC, HP ve Dell gibi firmalarda Linux işletim sistemlerini kendi ürün aileleriyle birlikte sunmaktadırlar.

Küme yapılarında Linux işletim sistemi kullanılmasının en önemli nedeni esnek yapısıdır. Linux işletim sistemi açık kaynak kodu olduğu için kolaylıkla değiştirilebilir, düzenlenebilir ve belirli bir görev için ince ayarlanabilir. Bazı kullanıcılara işletim sisteminin üzerinde oynamak korkutucu gelebilir ancak sanıldığı gibi değildir. Küme yapısının kurulabilmesi için Linux işletim sistemi üzerinde değişiklikler yapmak zorunlu değildir. Linux işletim sistemi üzerinde deneyimli sistem yöneticileri işletim sistemi üzerinde değişiklikler yapabilir.

Popüler arama motorlarından birisi olan Google Linux küme yapısını kullanan en büyük sistemlerden birisidir. Google altyapısında binlerce Linux sunucu çalışmaktadır. Google bilimsel amaçlı kullanılan kümelerden birisi olmamasına rağmen Linux işletim sisteminin esnekliğini ve uygulanabilirliğini gösterme açısından mükemmel bir örnektir.

Linux işletim sisteminin kümeleme uygulamalarında kullanılmasının diğer bir nedeni de çok çeşitli işlemcileri desteklemesidir. Çok farklı donanımlar üzerinde yüksek performansla çalışabilir.

Linux işletim sistemi sofistike çok görevli sanal bellek mimarisine sahip çekirdek üzerinde bulunur. Ancak istenirse çekirdeği küçültülebilir. Linux birçok elektronik cihazda gömülü işletim sistemi olarak yaygın şekilde kullanılmaktadır. Küçük çekirdekler genelde daha kararlıdır. Linux sahip olduğu özelliklerle kümeleme uygulamalarındaki öncülüğünü sürdürmektedir.

3.2. Linux Çekirdek Yapısı

Çekirdeğin en önemli görevi bilgisayarın donanımına erişim için bir arabirim olmak ve süreçler ve bellek yönetimi için gerekli olan ortamı sağlamaktır. Kullanıcı tarafından yazılan kod bir dosyayı açmak isterse, kullanıcı verileri için bellek ayrılmasını isterse veya bir TCP/IP mesajı göndermek isterse çekirdek kaynak yönetimi görevini yerine getirir. Kullanım amacına göre çekirdeğe eklentiler yapılabilir. Normal olarak küme yapısı için çekirdeğe eklenti yapılmasına gerek yoktur. Genellikle amaç çekirdeği küçük tutmak ve böylece performansı arttırmaktır.

3.2.1. Çekirdek ve dağıtımı

Çekirdek işletim sisteminin temeli olup önemli sistem fonksiyonlarını yerine getirir. Çekirdek işletim sisteminin diğer bileşenleriyle beraber çalışarak diğer işletim sistemi fonksiyonlarını da yerine getirir. Çekirdek sistemin düzgün çalışmasından, bilgisayar kaynaklarının düzenlenmesinden, kullanıcıların görevlerinin sırayla yapılmasından, bellek denetiminden ve yardımcı birimlerin (CD-ROM, disket sürücü vb.) çalışmasından sorumludur.

Linux dağıtımı Linux işletim sisteminin gerekli bileşenlerini ve en çok kullanılan programlarını bir araya getiren bir yazılım paketidir. Dağıtımda Linux çekirdeği, kurulum yazılımı ve diğer programlar bulunur. Farklı firmalar ve organizasyonlar tarafından dağıtımı yapılan farklı sürümler vardır. Bunun yanında küme desteğiyle beraber gelen dağıtımlarda vardır. Yaygın Linux sürümlerinin bazıları şunlardır;

- RedHat
- SuSE
- Mandriva
- Debian
- Slackware
- Turbolinux

Kümeleme uygulamalarında hangi Linux sürümü daha uygundur? Bu sorunun cevabını vermek kolay değildir. Genellikle çözüm 3 faktöre bağlıdır. Bunlar destek, dil ve kullanım kolaylığıdır. Eğer kümeleme için özel olarak hazırlanmış sürümler isteniyorsa ROCKS ve OSCAR kullanılabilir. ROCKS ve OSCAR yüksek performanslı işlem kümeleri oluşturmak için kullanılır. Eğer hata toleranslı, yük dengelemeli veya yüksek erişilebilir küme yapıları kurulacaksa Linux Virtual Server kullanılabilir. Linux Virtual Server dışında başka yazılım çözümleri de bulunmaktadır.

Linux kümeleme uygulamalarında tercih edilecek Linux sürümleri kararlı sürümler olmalıdır. Geliştirilme sürümleri tercih edilmemelidir. Kararlı sürümler kurulumun problemsiz olmasını sağlar.

3.2.2. Çekirdeğin derlenmesi

Piyasadaki tüm Linux dağıtımları çalışmaya hazır halde bulunan bir çekirdekle gelir. Aşağıdaki komutlar sistemde çalışmakta olan çekirdek hakkında bilgi almak için kullanılır.

```
% ls -l /proc/version
-r--r--r--    1 root  root     0 Jun 19 13:49 /proc/version
% cat /proc/version
Linux version 2.5.67 (root@linux.net) (gcc version 2.96 20000731
(Red Hat Linux 7.3 2.96-110)) #4 SMP Fri Apr 18 09:36:21 CDT
2008

% cd /usr/src
% ls -ld linux
lrwxrwxrwx 1 root root 21 Apr 22 07:19 linux -> /usr/src/linux-
2.5.67
```

Geçerli olan çekirdek versiyonunu görmenin bir diğer yolu da aşağıdaki komutu kullanmaktır.

```
#uname -a
```

'/proc' dosya sistemi gerçek anlamda bir dosya sistemi değildir. Disk üzerinde dosya saklamak için kullanılmaz. Sahte bir dosya sistemi olup çekirdek veri yapılarına bir arabirim olarak kullanılır. Yani çalışmakta olan çekirdeğe doğru bir penceredir.

Yeni bir Linux çekirdeği install etmek için yapılması gereken beş işlem vardır.

1. Kaynak kodun alınması
2. İstenilen seçeneklerin belirlenmesi
3. Kodun derlenmesi

4. Object kodunun install edilmesi
5. Boot yükleyicinin konfigüre edilmesi

Çekirdek indirilirken çift numaralı olan kararlı sürümler tercih edilmelidir. 2.2, 2.4, 2.6 gibi.

Çekirdeğin derlenebilmesi için aşağıdaki paketler gereklidir.

- binutils
- cpp
- gcc
- glibc-debug
- glibc-devel
- glibc-kernelheaders
- modutils (2.4 serisi çekirdeklerde) veya module-init-utils (2.6 serisi çekirdeklerde)

Eğer metin tabanlı bir terminal kullanılacaksa aşağıdaki paketlerde gereklidir.

- ncurses
- ncurses-devel

Çekirdek kaynak kodu genellikle Internetteki sunuculardan indirilir.

Çekirdeğin kaynak kodu genellikle '/usr/src' de saklanır. '/usr/src/linux' den bir sembolik link çekirdeğin oluşturulduğu dizini gösterir. Eğer farklı bir çekirdek indirilmek ve tekrar derlenmek isteniyorsa '/usr/src' altına kopyalanır ve çekirdek tekrar derlenir. Çekirdeğin kaynak kodu genellikle aşağıdaki gibi görünür.

```
# cd /usr/src/linux
# ls
COPYING         Makefile         crypto   init    mm         sound
CREDITS         README           drivers  ipc     net        usr
Documentation   REPORTING-BUGS   fs       kernel  scripts
MAINTAINERS     arch             include  lib     security
```

Eğer Linux dağıtımı çekirdeğin kaynak kodunu sağlamışsa aşağıdaki komutlar kullanılarak çekirdek tekrar derlenebilir.

```
# make clean ; make bzImage
```

Sunucu işlemleri tamamladığında çekirdek hazır hale gelecektir.

```
% ls -l /usr/src/linux-2.2.14/arch/i386/boot/bzImage
-rw-r--r-- 1 root root 906584 Jun 19 00:13
              /usr/src/linux-2.5.67/arch/i386/boot/bzImage
```

3.2.3. Yüklenebilen çekirdek modülleri

Linux dağıtımları ile gelen çekirdeklerin çoğu modüler yapıdadır. Linux yüklenebilen çekirdek modülleri için bir arabirime sahiptir. Bu arabirim çekirdeğin işlevlerinin genişletilmesi için dinamik bir yöntem sağlar. Böylece her yeni eklenen ve güncellenen modülden dolayı çekirdeğin tekrar derlenmesine gerek kalmaz. Modüller genellikle aygıt sürücüleri, dosya sistemleri, ve özel çekirdek özellikleri için kullanılır. Örneğin Linux MSDOS dosya sistemlerinden okuyabilir ve bu dosya sistemlerine yazabilir. Ancak bu fonksiyon sürekli olarak gerekli değildir. Linux çekirdeği MSDOS dosya sistemi için gerekli olan çekirdek modülünü bir MSDOS dosya sistemi mount edileceğinde dinamik olarak yükler. Çekirdeğin boyutu ilave işlevler eklenene kadar küçüktür. Fonksiyonların çoğu çekirdeğin merkezinden alınmış ve dinamik olarak yüklenebilen modüller haline getirilmiştir. Böylece Linux işletim sisteminin kararlı bir yapıda olması sağlanmıştır. Linux dağıtımlarının çoğu farklı donanım konfigürasyonlarını ve kullanımlarını desteklemek için önceden derlenmiş çekirdek modülleri ile birlikte gelir.

3.3. Linux Dosya Sistemleri

Linux birçok dosya sistemini destekler. Modüler yapıdaki çekirdeği ve çekirdeğinde kullanılan sanal dosya sistemi arabiriminden dolayı dinamik olarak yüklenebilen modüller yüklenebilir veya kaldırılabilir. Küme yapısında bir veya iki dosya sistemi yeterli olacaktır. Linux işletim sisteminde standart olarak kullanılan dosya sistemi EXT2 kabul edilir. EXT2 hızlı ve kararlı bir dosya sistemidir. Her küme yapısında EXT2 bulunacaktır. Ama tek başına EXT2 yeterli değildir. Çünkü EXT2 günlüklü bir dosya sistemi değildir.

3.3.1. Günlüklü dosya sistemleri

Günlüklü dosya sistemlerinin ardındaki fikir oldukça basittir. Tüm disk yazma işlemlerinin gerçekleştirileceğini garantilemektir. Böylece disk daima kararlı durumda bulunacaktır. EXT2 gibi dosya sistemlerinde bu mümkün değildir. Örneğin EXT2 dosya sistemine yazarken bilgisayarın enerjisi kesilirse, dosya sistemi kararsız durumda kalır. Bilgisayar tekrar başlatıldığında dosya sistemi kontrolü yapılır ve dosya sistemi tekrar kararlı duruma getirilir. Bu önemsiz bir nokta değildir. Çünkü zaman kaybettirici bir işlemdir. Kullanılan disk alanı arttıkça sorun daha da büyür. Bazen saatlerce sürebilir. Eğer RAID yapısı kullanılıyorsa günlüklü dosya sistemi kullanılması uygun olacaktır.

Günlüklü dosya sistemleri disk yazma işlemleri sırasında günlüksüz dosya sistemlerinden daha yavaştır. Günlüklü dosya sistemleri bir sorun olduğunda diski kararlı halde tutabilmek için diske günlük adı verilen kayıtları yazar. Bu bilgi günlüklü dosya sistemlerini kararlı yapar ancak biraz da yavaşlatır.

Eğer kullanılacak uygulama sürekli olarak diske yazma işlemleri gerçekleştirecekse küme yapısında EXT2 gibi günlüksüz bir dosya sistemi yeterli olacaktır.

3.3.1.1. Hangi günlüklü dosya sistemi kullanılabilir?

Eski işletim sistemlerinin aksine Linux işletim sisteminde seçenekler çok fazladır. En yaygın kullanılan günlüklü dosya sistemleri ReiserFS, IBM JFS, SGI XFS dir. Bunlar arasında EXT3 en uygun seçim olacaktır. EXT3 EXT2 dosya sisteminin dosya formatını kullanır ve buna günlük yeteneği ekler. ReiserFS SuSE Linux işletim sisteminde kullanılan günlüklü dosya sistemi olup EXT3'e göre daha karmaşık algoritmalar kullanır. IBM JFS ve SGI XFS dosya sistemleri Linux işletim sisteminden önce AIX ve IRIX işletim sistemlerinde kullanılmaktaydı. Birçok küme yapısı için sürüm bağımsız olduğundan dolayı EXT3 uygun seçim olacaktır [1].

3.3.2. Ağ ve dağıtık dosya sistemleri

Linux kümelerinde genellikle verilerin saklanması için yerel dosya sistemleri kullanılır. Ancak verilerin paylaşılması gerektiğinde ağ tabanlı veya dağıtık dosya sistemleri kullanılabilir. Ağ tabanlı dosya sistemi düğümün uzaktaki makinelere dosya okuma ve yazma işlemleri için erişebilmesini sağlar. En yaygın ve popüler olanı NFS olup uzun zamandır kullanılmaktadır. NFS istemcisi IP protokolü kullanan ağ üzerinde bulunan uzak bir dosya sistemini yükleyebilir. NFS sunucusu birçok uzak istemciden dosya erişim isteklerini kabul edebilir ve verileri yerel olarak saklayabilir. NFS farklı platformlarda standartlaşmış bir sistemdir.

3.4. Düğümde Çalışan Linux İşletim Sisteminde Gereksiz Bileşenlerin Çıkarılması

Linux kümeleri kurulurken temel fikir; daha küçük işletim sisteminin daha iyi olduğudur. Ancak gereksiz bileşenler çıkarılırken iyice incelenmelidir. Çekirdek veya işletim sistemi bileşenlerinden gereksiz olanlar boşu boşuna işlemci zamanını harcarlar.

Linux düğümünde gereksiz bileşenlerin atılması işleminde başlangıç noktası kurulum aşamasıdır. Birçok dağıtımda iş istasyonu, sunucu veya yazılım geliştirme ortamı gibi konfigürasyonları içeren seçenekler vardır. Genellikle sunucu kurulumları doğru başlangıç noktasıdır. İş istasyonu konfigürasyonları varsayılan olarak pencere sistemleri ile gelir ve amaç Linux işletim sistemini masaüstü kullanıcısı için kullanıcı dostu yapmaktır.

3.5. Önemli Linux Init Scriptleri

Linux işletim sistemlerinde kullanılan önemli Linux scriptleri hakkındaki bilgiler aşağıda bulunmaktadır.

atd: at komutuyla işlerin zamanlanmasını sağlar. Genelde cron tercih edilir.

crond: İşlerin zamanlanmasını yapar.

functions: /etc/init.d dizininde bulunan diğer scriptler tarafından kullanılan kütüphane rutinlerini içerir. Bu scriptin değiştirilmemesi gerekir. Aksi halde sistemde çalışan tüm scriptler bozulabilir.

halt: Sistemin problemsiz olarak kapatılmasını sağlar.

httpd: Web sunucu için apache daemonlarının başlatılmasını sağlar.

identd: Sisteme uzaktan bağlananları tanımlamayı sağlar.

ipchains veya iptables: Kernel 2.2 öncesinde ipchains, kernel 2.4 ve sonrasında iptables kullanılır. Linux çekirdeğinde ağ paketlerinin incelenmesini ve yönetimini sağlar. Linux platformunda en yaygın kullanılan firewall uygulamasıdır.

keytable: Klavye tablosunun yüklenmesini sağlar.

killall: Sistemin kapatılmasına yardımcı olur.

lpd: LPRng yazdırma sisteminin başlatılmasını sağlayan scripttir.

netfs: Küme düğümlerinde boot sırasında çalıştırılan bu script diğer küme düğümleriyle paylaşılan veri dosyalarına erişim elde ederken gereklidir.

network: Ethernet arabirimlerini çalışır duruma getirir ve sistemin küme ağına ve NAS cihazına bağlanmasını sağlar.

nfs: NFS sunucu için gereklidir.

nfslock: NFS kilitleme mekanizması tarafından kullanılır.

ntpd: Network Time Protocol daemonunu başlatır.

portmap: NFS ve NIS'ın RPC bağlantılarını yönetmesini sağlar.

random: Şifreleme rutinleri tarafından kullanılır.

rawdevices: Aygıt yönetimi için çekirdek tarafından kullanılır.

rusersd: Makineye log on olmuş kullanıcıların diğer kullanıcılar tarafından görülebilmesini sağlar.

sendmail: Linux işletimde bulunan SMTP sunucu.

*single***:** init süreci tarafından kullanılır ve runlevelların yönetimini sağlar.

smb: Windows istemcilerle dosya ve yazıcı paylaşımını sağlayan Samba paketidir.

squid: Web sayfalarının daha hızlı gösterilebilmesi için cache imkanı sunan bir vekil sunucudur.

syslog: Çalışmakta olan daemonlardan ve programlardan gelen hata mesajlarını loglar.

xinetd: Uzak bağlantı isteği gelince FTP ve Telnet gibi servisleri başlatır.

init İle Servislerin Başlatılması: Boot yükleyicisi çekirdeği başlatıp çalışır duruma getirdiğinde *init* daemon'u çalıştırılır. Daha sonra init /etc/inittab konfigürasyon dosyasındaki satırlara göre sistemin işletimi için gerekli olan daemonları başlatır.

/etc/inittab dosyası: /etc/inittab konfigürasyon dosyasının kalbi sistemin runlevellarını tanımlayan aşağıdaki yedi satırdır. Sistem sadece bir runlevel'da olabilir.

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

rc programı (/etc/rc.d/rc scripti) sistemin runlevel'ı değiştiği her sefer çalışmalıdır. *init rc* programına 0-6 arasında değer alan sistemin runlevel numarasını gönderir. *rc* programı iletilen runlevel'a göre /etc/rc.d dizininde bulunan runlevel dizininden ilgili scriptleri çalıştırır. Varsayılan runlevel /etc/inittab dosyasında initdefault satırında tanımlanmıştır. Hangi runlevel'da bulunulduğu bilgisi *runlevel* komutuyla görülebilir. Genelde runlevel 3'te çalışılır.

İşletim sisteminin sona ermiş bir daemonu otomatik olarak başlatması sağlanabilir. Bu işlem için respawn seçeneği ile çalıştırılabilir program dosyası /etc/inittab dosyasında belirtilir. init runlevel'a girildiğinde daemonu çalıştıracak ve daemonu izleyecektir. Daemon sonlanırsa tekrar çalıştırılacaktır. Örnek bir satır aşağıda görülebilir.

```
sn:2345:respawn:/usr/local/scripts/start_snmpd > /dev/null
```

Bu satır init daemona /usr/local/scripts/start_snmpd scriptini 2, 3, 4, ve 5 runlevellarında çalıştırmasını ve her çıktıyı /dev/null 'a göndermesi gerektiğini belirtir. init daemonu start_snmpd scriptinin çalışması sonlandığında snmpd daemonunun sonladığını kabul eder ve scripti tekrar çalıştırır. start_snmpd scriptine bir örnek aşağıda görülebilir.

```
#!/bin/bash
```

```
exec /usr/sbin/snmpd –s –P /var/run/snmpd –l /dev/null
```

Scriptin ilk satırı bash kabuğunu başlatır. İkinci satır ise snmpd daemonunu çalıştırır. Bunu yaparken de init daemonunun atanmış olan PID değerini bilmesini ve bu PID değerini izlemesini sağlar. Böylece sona eren daemon tekrar başlatılabilir.

Gereksiz olan servisler durdurulabilir. Bunun için ilk yapılması gereken daemonun öldürülmesidir.

```
#/etc/init.d/<script_adı> stop
```

Red Hat tabanlı sistemlerde aşağıdaki komutla servis durdurulabilir.

```
#service <script_adı> stop
```

Daha sonra scripti gösteren sembolik linkler aşağıdaki komutla ortadan kaldırılır.

```
#chkconfig --del <script_adı>
```

İşletim sistemi süreçlerini ve harici programlardan gereksiz olanları çıkarmanın birinci adımı sistemde nelerin çalıştığını ortaya çıkarmaktır. Linux sistemlerinin çoğu için süreçleri ve işlemleri başlatmak için en az iki standart yöntem vardır.

inetd: Yeni Linux dağıtımlarında programın yeni versiyonu olan xinetd bulunur. Her iki programın da temel işlevi bir portlar setinde bağlantılar için beklemek ve bağlantıları çoğaltmak, içeriye doğru bir bağlantı kurulduğunda da ağ bağlantısını uygun programa sunmaktır. Inetd ve xinetd programları tarafından dinlenen portların konfigürasyonu ve üzerinde işlem yapılacak portlar ‘/etc/inetd.conf’ ve ‘/etc/services’ veya ‘/etc/xinetd.conf’ ve ‘/etc/xinetd.d’ dosyalarına bakılarak görülebilir.

/etc/rc.d/init.d: Bu özel dizin boot sırasında çalışan scriptleri ve makine shut down edilene kadar çalışacak daemonları çalıştıran scriptleri içerir.

inetd.conf: inetd.conf basit bir konfigürasyon dosyasıdır. Dosyadaki her bir satır servisle ilişkilendirilen portla ve porta bir bağlantı yapıldığında başlatılacak programla birlikte tek bir servisi simgeler.

```
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd
in.proftpd
finger  stream  tcp     nowait  root    /usr/sbin/tcpd
in.fingerd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
```

İlk sütun servisin adını simgeler. '/etc/services' dosyası port adı port numarası eşleştirmesini yapar.

```
% grep ^talk /etc/services
talk 517/udp # BSD talkd(8)
```

Düğümde çalışan Linux işletim sistemini hafifletmek için 'inetd.conf' dosyasındaki ekstra servislerden kurtulmak gerekir. Örneğin talk servisine birçok düğümde gerek yoktur. Çok katı güvenlik uygulanan ortamlarda 'inetd.conf' dosyasında hiçbir satır olmayabilir.

/etc/rc.d/init.d: Sonraki adım boot sırasında başlatılan daemonları ve süreçleri engellemektir. Linux dağıtımlarının çoğunda /etc/rc.d/init.d run level'a girerken veya çıkarken çalışan scriptleri içerir. Aşağıda bir örneği bulunmaktadır.

```
% cd /etc/rc.d/init.d
% ls
anacron     functions     kdcrotate  nfslock     sendmail    wine
apachectl   gpm           keytable   nscd        single      xfs
apmd        halt          killall    ntpd        snmpd
xinetd
arpwatch    http_sanity   kudzu      portmap     snmptrapd
ypbind
atd         http_sanity~  lpd        radvd       sshd
yppasswdd
autofs      identd        netfs      random      syslog
ypserv
crond       ipchains      network    rawdevices  vncserver
ypxfrd
cups        iptables      nfs        rhnsd       winbind
```

Ancak scriptin olması çalışacağı anlamına gelmez. Diğer dizinler ve sembolik linkler çalışacak scriptleri kontrol eder. Çoğu sistemde chkconfig arabirimi scriptleri ve sembolik linkleri kontrol eder. Örneğin run level 3'e girerken çalışacak scriptleri görmek için aşağıdaki komut kullanılabilir.

```
% chkconfig --list | grep '3:on'
syslog 0:off 1:off 2:on 3:on 4:on 5:on 6:off
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
lpd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
mysql 0:off 1:off 2:on 3:on 4:on 5:on 6:off
httpd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
sshd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
atd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
named 0:off 1:off 2:off 3:on 4:on 5:on 6:off
dhcpd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
gpm 0:off 1:off 2:on 3:on 4:on 5:on 6:off
inet 0:off 1:off 2:off 3:on 4:on 5:on 6:off
network 0:off 1:off 2:on 3:on 4:on 5:on 6:off
nfsfs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
random 0:off 1:off 2:on 3:on 4:on 5:on 6:off
keytable 0:off 1:off 2:on 3:on 4:on 5:on 6:off
nfs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
nfslock 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ntpd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
portmap 0:off 1:off 2:off 3:on 4:on 5:on 6:off
sendmail 0:off 1:off 2:on 3:on 4:on 5:on 6:off
serial 0:off 1:off 2:on 3:on 4:on 5:on 6:off
squid 0:off 1:off 2:off 3:on 4:on 5:on 6:off
tltime 0:off 1:off 2:off 3:on 4:off 5:on 6:off
crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Örneğin seri portların boot sırasında başlatılması istenmiyorsa “serial” script’i çıkarılabilir. lpd,mysql,sendmail,squid vb. gibi servislerden gerekmeyenler kaldırılabilir. chkconfig komutu gereksiz olan servislerin sonlandırılması için kullanılır.

Diğer Süreçler: ‘inetd.conf’ dosyası ve ‘/etc/rc.d/init.d’ de bulunan scriptler dışında da işlemci ve bellek kaynaklarını tüketen bileşenler vardır. ‘cron’ programı programları belirli zamanlarda çalıştırmak için kullanılır. Genelde yedekleme ve sistem dosyalarının temizlenmesi için kullanılır.

‘ps’ komutu çalışmakta olan süreçlerin işlemci, bellek kaynaklarını ne kadar kullandığı bilgisini de gösterir.

```
% ps -eo pid,pcpu,sz,vsize,user,fname --sort=vsize
```

Yukarıdaki komut süreçleri kullandıkları sanal bellek boyutuna göre sıralar.

```
PID %CPU   SZ   VSZ  USER COMMAND
26593  0.0  804  3216   web httpd
26595  0.0  804  3216   web httpd
 3574  0.0  804  3216   web httpd
  506  0.0  819  3276  root squid
```

```
  637  0.0  930  3720  root AgentMon
  552  0.0 1158  4632 dbenl postmast
13207  0.0 1213  4852  root named
13209  0.0 1213  4852  root named
13210  0.0 1213  4852  root named
13211  0.0 1213  4852  root named
13212  0.0 1213  4852  root named
  556  0.0 1275  5100 dbenl postmast
  657  0.0 1280  5120 dbenl postmast
  557  0.0 1347  5388 dbenl postmast
  475  0.0 2814 11256 mysql mysqld
  523  0.0 2814 11256 mysql mysqld
  524  0.0 2814 11256 mysql mysqld
  507  0.0 3375 13500 squid squid
```

Yukarıdaki süreç bilgilerine bakıldığında squid programının az CPU kullanmasına rağmen yüksek bellek kullanımı olduğu görülebilir. Küme yapısında düğüm kaynaklarını tüketen süreçleri bulmak için ps komutu vazgeçilmez bir araçtır.

3.6. /proc ile İnce Ayar Yapma

/proc dosya sistemi gerçek bir dosya sistemi değildir. Ancak çalışmakta olan çekirdeğe açılan bir pencere olarak değerlendirilebilir.

Düzgün olarak ayarlanmış bir küme yapısındaki düğümde neredeyse mevcut olan tüm CPU ve bellek kaynakları kümeye atanmıştır. Çekirdeği küçültmek ve gereksiz olan süreçleri ve arka plan yordamlarını azaltmak host üzerinde çalışan uygulamaya kaynak ayırmayı sağlayacaktır. Ufak performans sıkışıklıkları basit çekirdek ayarlamalarıyla çözülebilir [1].

```
% cat /proc/net/dev
Inter-| Receive | Transmit
face |bytes packets errs drop fifo frame compressed
multicast|bytes
packets errs drop fifo colls carrier compressed
lo:363880104 559348 0 0 0 0 0 0 363880104 559348 0 0 0 0 0 0
eth0:1709724751 195793854 0 0 357 0 0 0 4105118568 202431445
0 0 0 0 481 0
brg0: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Yukarıdaki Ethernet cihazının bilgilerine bakılarak giden-gelen paket sayısı, çarpışma sayısı, hatalı paket sayısı ve düşen paket sayısı elde edilebilir. Bu bilgiler Ethernet ağında yaşanabilecek sıkıntıları görmeyi sağlayabilecektir. Eğer çok fazla çarpışma varsa hub yerine switch kullanılabilir. Çünkü ağ küme yapısı için çok önemli bir bileşendir.

İnce ayar yapılabilecek çekirdek parametreleri '/proc/sys 'de bulunur. Ağ parametreleri genellikle '/proc/sys/net' dosyasında bulunur. Parametrelerin çoğu değiştirilebilir. Küme çekirdeği tcp_sack, tcp_-timestamps, tcp_window_scaling, rmem_default, rmem_max, wmem_default, ve wmem_max parametreleri değiştirilerek ayarlanabilir.

Bellekle ilgili değerler '/proc/meminfo' dosyasında görülebilir.

```
% cat /proc/meminfo
MemTotal:      1032828 kB
MemFree:         24916 kB
Buffers:        114836 kB
Cached:         436588 kB
SwapCached:      58796 kB
Active:         720008 kB
Inactive:       210888 kB
HighTotal:      130496 kB
HighFree:         2016 kB
LowTotal:       902332 kB
LowFree:         22900 kB
SwapTotal:      530136 kB
SwapFree:       389816 kB
Dirty:              64 kB
Writeback:           0 kB
Mapped:         390116 kB
Slab:            57136 kB
Committed_AS:   761696 kB
PageTables:       7636 kB
ReverseMaps:    202527
```

'/proc/sys/ vm' düğümdeki sanal bellek ile ilgili parametreleri bulundurur.

Dosya sistemine hızlıca göz atmak için aşağıdaki komut kullanılabilir.

```
% cat /proc/sys/fs/file-nr
1157 728 4096
```

Harddisk parametrelerine bakmak için aşağıdaki komut kullanılabilir.

```
% /sbin/hdparm -I /dev/hda

/dev/hda:

 Model=DW CDW01A0 A , FwRev=500.B550, SerialNo=DWW-AMC1211431 9
 Config={ HardSect NotMFM HdSw>15uSec SpinMotCtl Fixed DTR>5Mbs
FmtGapReq }
 RawCHS=16383/16/63, TrkSize=57600, SectSize=600, ECCbytes=40
 BuffType=3(DualPortCache), BuffSize=2048kB, MaxMultSect=16,
MultSect=8
 DblWordIO=no, maxPIO=2(fast), DMA=yes, maxDMA=0(slow)
 CurCHS=17475/15/63, CurSects=16513875, LBA=yes
 LBA CHS=512/511/63 Remapping, LBA=yes, LBAsects=19541088
 tDMA={min:120,rec:120}, DMA modes: mword0 mword1 mword2
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, PIO modes: mode3
mode4
 UDMA modes: mode0 model *mode2 }
```

Aşağıdaki komutla basit bir test yapılabilir.

```
% /sbin/hdparm -t /dev/hdal

/dev/hdal:
Timing buffered disk reads: 64 MB in 20.05 seconds = 3.19 MB/sec
```

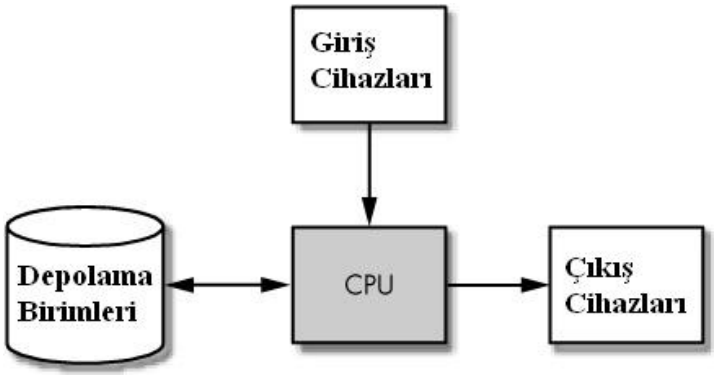
'/proc/sys/kernel' çekirdek ile ilgili ince ayarlar yapmak için kullanılabilir. Mesaj iletme ile ilgili değerler '/proc/sys/kernel/shmmax' dosyasında bulunur. Paylaşılan bellek bölümlerinin maksimum boyutunu öğrenmek veya değiştirmek için kullanılabilir.

```
% cat /proc/sys/kernel/shmmax
33554432
```

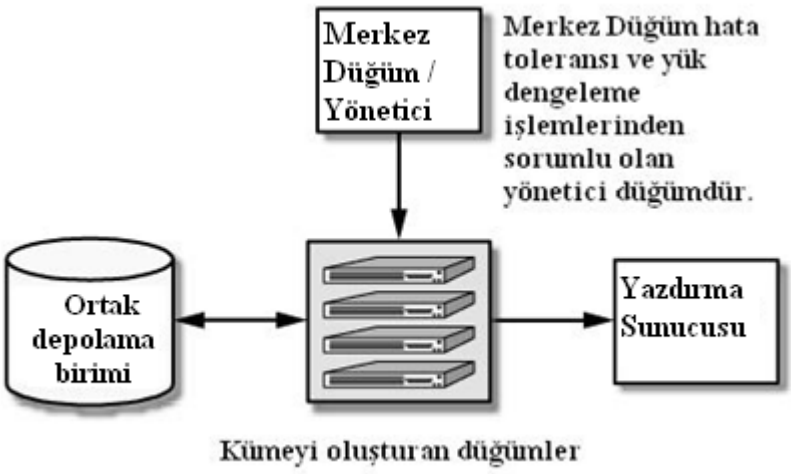
BÖLÜM 4

4. LINUX KÜME MİMARİSİ

Bir bilgisayarın yapısını oluşturan ve bir kümenin yapısını oluşturan bileşenler aşağıdaki şekillerde görülmektedir.



Şekil 4.1. Bilgisayar Yapısını Oluşturan Bileşenler



Şekil 4.2. Kümeyi Oluşturan Bileşenler

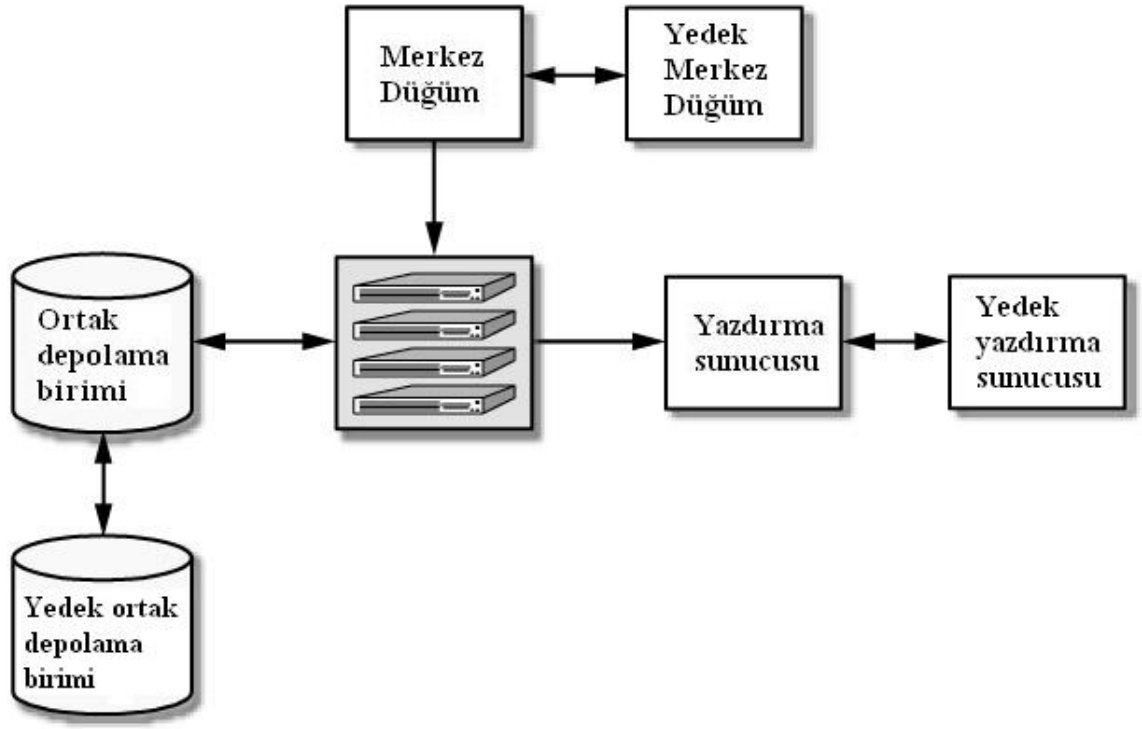
Merkez Düğüm / Yönetici: Merkez düğüm küme yapısını oluşturan düğümlerle kullanıcılar arasında bulunan bileşendir. Merkez düğüm kullanıcıdan gelen isteği kümeyi oluşturan düğümlerden çalışmakta olan bir düğüme yönlendirir. Bu işlem sırasında yük dengeleme işlemini de yerine getirebilir.

Ortak Depolama Birimi: Ortak depolama birimi kümedeki veriler için tek depolama alanıdır. Kişisel bilgisayarlarda bulunan sabit disk sürücünün görevini yerine getirir. Küme yapısında ortak depolama birimi kullanılmasındaki amaç dosya kilitleme mekanizması kullanmak ve farklı düğümlerde çalışan programların aynı veriye ulaşmasını engellemektir. Ortak depolama birimi olarak genelde NAS (Network Attached Storage) veya SAN (Storage Area Network) yapıları kullanılır. NAS ve SAN küme yapısındaki switch'e 1000Mbps hızında yedekli olarak bağlıdır. NAS veya SAN kullanılmasındaki amaç veri güvenliğini sağlamak ve yüksek performansla düğümlere hizmet sunabilmektir.

Yazdırma Sunucusu: Yazdırma sunucusu küme yapısında zorunlu bir bileşen değildir. Sadece çıktı alınması gerekli olan kümelerde kullanılır.

4.1. Tek Hata Noktası Problemini Ortadan Kaldırma ve Yüksek Devamlılığı Sağlama

Bütün kritik öneme sahip sistemlerde en önemli gereksinimlerden birisi tek hata noktası problemini ortadan kaldırmak ve yüksek devamlılığı sağlamaktır. Kümedeki düğümlerden herhangi birisi kapatılıp açılabilir ve bu servislerde bir kesintiye yol açmaz. Aşağıdaki şekilde yüksek devamlılığı sağlayabilecek bir küme yapısı örneği bulunmaktadır. Tek hata noktası problemini ortadan kaldırmak için merkez düğüm sayısı en az ikidir. Merkez düğümde bir problem meydana gelirse kümedeki operasyonların yönetimini yedek merkez düğüm alacaktır.



Küme Yapısında Kaynaklar

Şekil 4.3. Yüksek Erişilebilir Küme Yapısını Oluşturan Bileşenler

4.2. Linux Kümelerinin Diğer Sistemlere Göre Avantajları

Linux kümelerinin diğer sistemlere göre avantajları şunlardır:

- Genişleyebilir kapasite
- Yüksek fiyat/performans oranı
- Konfigürasyon esnekliği
- Teknoloji takibi
- Kısa sürede geliştirilebilme
- Herhangi bir donanım üreticisine bağımlı olmama
- Kullanılan sunucu işletim sisteminin Linux gibi ücretsiz, güçlü ve istikrarlı olması

- Herhangi bir kümede meydana gelebilecek donanımsal bir arızanın tüm kümenin çalışmasının etkilemeyecek olması

4.3. Linux Kümelerinin Dezavantajları

Linux kümelerinin diğer sistemlere göre dezavantajları şunlardır:

- Linux kümelerinin çok fazla yer işgal etmesi
- Servis yokluğu

4.4. Servislerin Başlatılması

Bir küme ortamında daemonlar veya servisler birçok yöntemle çalıştırılabilir. Bu yöntemler şunlardır:

- Sistem boot ederken veya runlevela giriş sırasında init daemon'u tarafından
- Merkez düğüm ilk defa çalışırken veya bir kaynağın sahipliğini alması gerektiğinde merkez düğüm tarafından
- Bir ağ isteği geldiğinde xinetd tarafından
- Standart dışı yöntemler kullanılarak

Hangi yöntemin kullanılacağı aşağıdaki soruların cevaplarına bağlıdır. Bu sorular şunlardır:

- Servis her zaman mı çalışacak, yoksa sadece ağ istekleri geldiğin de mi?
- Birincil sunucu çöktüğünde servis yedek merkez düğüme geçecek mi?
- Servis durduğunda veya anormal olarak sona erdiğinde otomatik olarak tekrar başlayacak mı?

Eğer servis gelen bir ağ isteği sonucunda çalışacaksa *xinetd* tarafından başlatılmalıdır. Ancak Linux sunucuda ve kümedeki düğümlerde çalışan servislerin çoğu her zaman çalışacağı için boot sırasında *init* tarafından başlatılmalıdır. Eğer servis sistem çöktüğünde bile çalışacaksa kümeleme yazılımı tarafından başlatılmalıdır [2].

4.5. Paketlerin İşlenmesi

Linux sunucuya bir istemciden paket geldiğinde paket daemon veya servise iletilmeden önce çekirdekten geçmelidir.

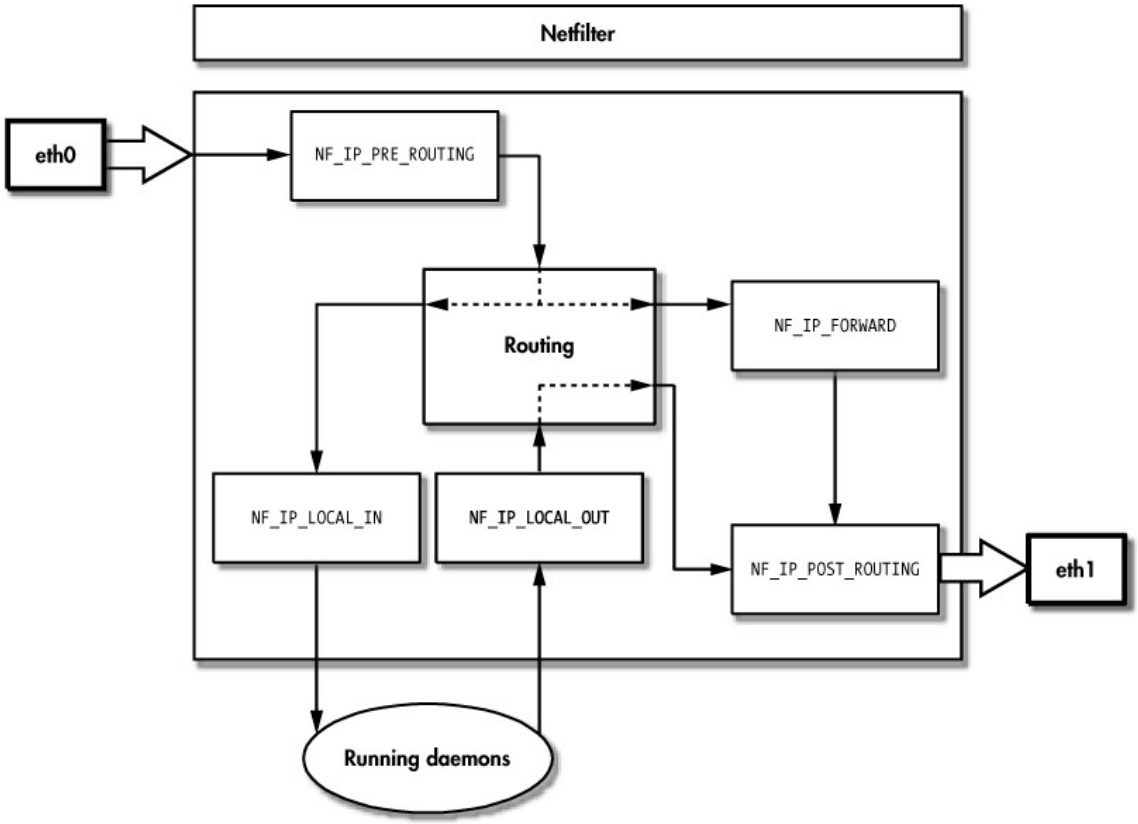
4.5.1. Netfilter

Sisteme ağ yoluyla gelen her paket protokol yığıtına aktarılır. Protokol yığıtı ağ kartına gelen elektronik sinyali servislerin ve daemonların anlayabileceği mesajlara dönüştürür. Daemon bir yanıt gönderdiği zaman çekirdek bir paket yaratır ve bunu ağ kartından gönderir. Gelen paketler protokol yığıtından yukarı veya aşağı yönde işlenirken, Netfilter Linux çekirdeğine bağlanır ve paketlerin kontrolünün yapılmasını sağlar [2].

Netfilter içinde paket bilgisi birçok noktada hem geliş hem de gidiş yönünde değiştirilebilir. Aşağıdaki şekilde iki ağ kartlı bir sunucuda Netfilter'ın yapısı görülebilir.

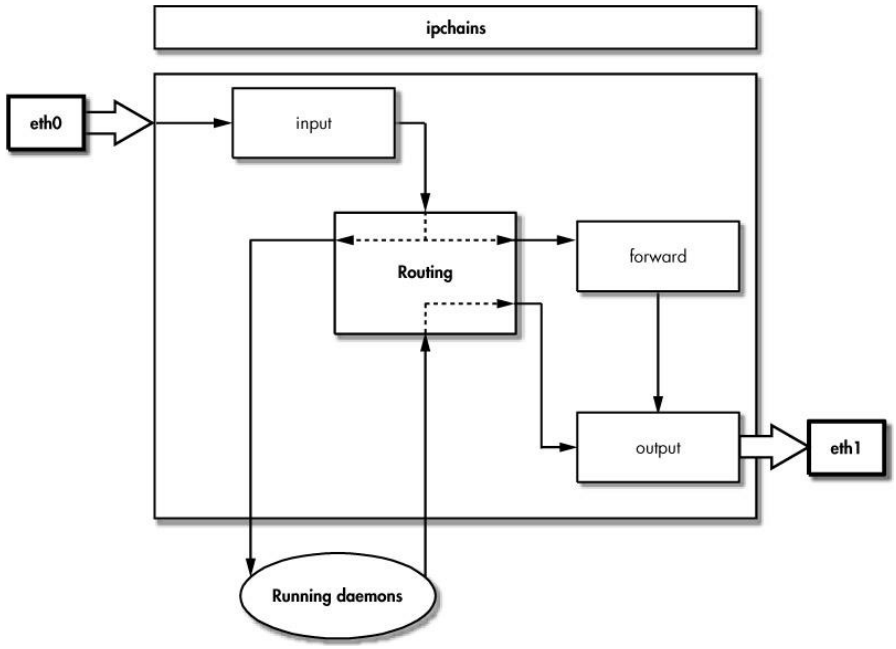
Netfilter da bulunan Routing bileşeni çekirdeğin her paket aldığında veya gönderdiğinde vermesi gereken yönlendirme kararından sorumludur.

Netfilter INPUT, FORWARD ve OUTPUT kurallar listesini kullanır ve çekirdekten geçen her paketi bu kurallara eşleştirmeye çalışır. Eğer bir eşleşme bulunursa paket kuralda belirtilen işleme tabi tutulur ve sonraki kurallara bakılmaz. Eğer eşleşen hiçbir kural bulunmazsa paket varsayılan politikaya göre işlenir. Internet gibi güvenilir olmayan dış ağlara bağlı olan sunucularda varsayılan politika genelde DROP olup hiçbir kurala uymayan paketler düşürülür [2].



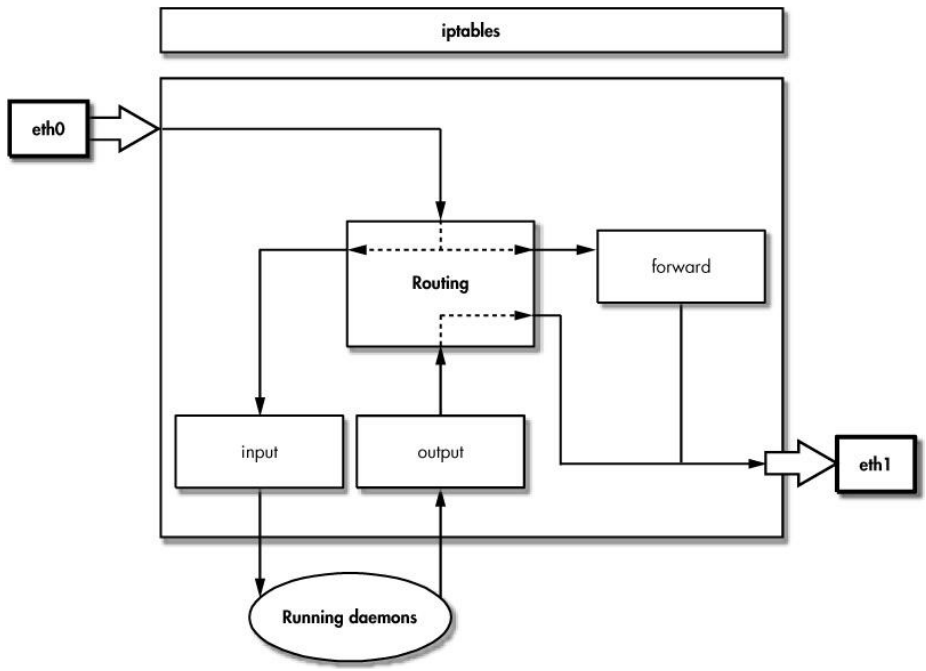
Şekil 4.4. Linux çekirdeğinde bulunan beş Netfilter kancası

Netfilter kodu çekirdek 2.2'den 2.4'e geçişte oldukça değişmiştir. Aşağıdaki şekilde Linux 2.2 serisi çekirdeklerde ipchains tarafından kullanılan input, forward ve output zincirleri görülmektedir. Eth0 ağ kartından gelen bir paket eth1 ağ kartından çıkana kadar input, forward ve output zincirlerinden geçmelidir.



Şekil 4.5. Linux 2.2 serisi çekirdeklerde ipchains tarafından kullanılan zincirler

Aşağıdaki şekilde Linux 2.4 serisi çekirdeklerde bulunan iptables görülmektedir. INPUT kuralları yerel olarak çalışan daemonlara hedeflenen paketlere uygulanır. FORWARD kuralları uzak bir hosttan gelen ve ağa geri gönderilecek olan paketlere uygulanır. OUTPUT kuralları yerel olarak çalışan daemonlar tarafından yaratılmış paketlere uygulanır.



Şekil 4.6. Linux 2.4 serisi çekirdeklerde iptables tarafından kullanılan zincirler

ipchains ve iptables uygulamalarında her bir zincir için varsayılan politika tanımlama imkanı vardır. Bu zincirlerin varsayılan politikalarında bulunan ACCEPT zincirdeki hiçbir kurala uymayan her paketin geçirilmesini, DROP paketin düşürülmesini, REJECT ise göndericiye paketinin kabul edilmediğinin bildirilmesini sağlar. Bu ağ paketleri üzerindeki kontrol Linux makinelerinin kolaylıkla firewall olarak kullanılabilmesini sağlar.

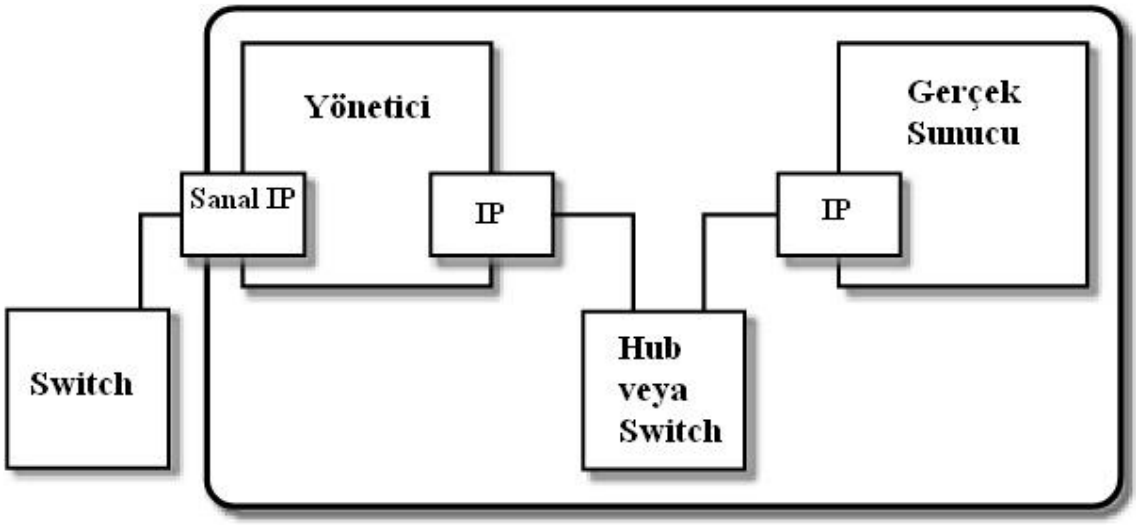
4.6. Linux Virtual Server

IP Virtual Server (IPVS) Linux sistemlerde kullanılan küme yük dengeleme yazılımıdır. IPVS yazılımı Linux 2.4.23 çekirdeği ile gelmeye başlayan bir çekirdek yamaları bütünüdür. Çekirdeğin yönlendirme ve paket filtreleme yetenekleriyle birleştirildiğinde IPVS'in devreye alındığı bir çekirdek bir Linux makineyi yük dengeleyici haline çevirir. IPVS'in devrede olduğu bir küme yük dengeleyici ve küme düğümleri LVS (Linux Virtual Server) olarak adlandırılır.

LVS kümesi yük dengeleyicisi servisler için gelen tüm istemci bilgisayar isteklerini alır ve her bir isteğe yanıt verecek küme düğümüne karar verir. Küme yapısındaki yük dengeleyicisi LVS Yöneticisi olarak isimlendirilir.

LVS kümesinin içindeki düğümler işlemlerin gerçekleştiği gerçek sunuculardır. İstemci bilgisayarlar, Yönetici ve düğümler birbirleri ile ağ yapısındaki bilgisayarların haberleşmesinde olduğu gibi IP adresini kullanarak haberleşirler.

Aşağıdaki şekilde yönetici düğüm ve gerçel sunucuların şematik bağlantısı görülmektedir.



Şekil 4.7. Yönetici ve Gerçek Sunucu Düğümlerinin Şematik Bağlantısı

LVS Küme Tipleri: LVS kümeleri genellikle LVS Yöneticisinin gelen istekleri küme içindeki düğümlere iletirken kullandığı yönteme göre tanımlanır. Üç yöntem vardır.

- LVS-NAT (LVS Network Address Translation)
- LVS-DR (LVS Direct Routing)
- LVS-TUN (LVS IP Tunneling)

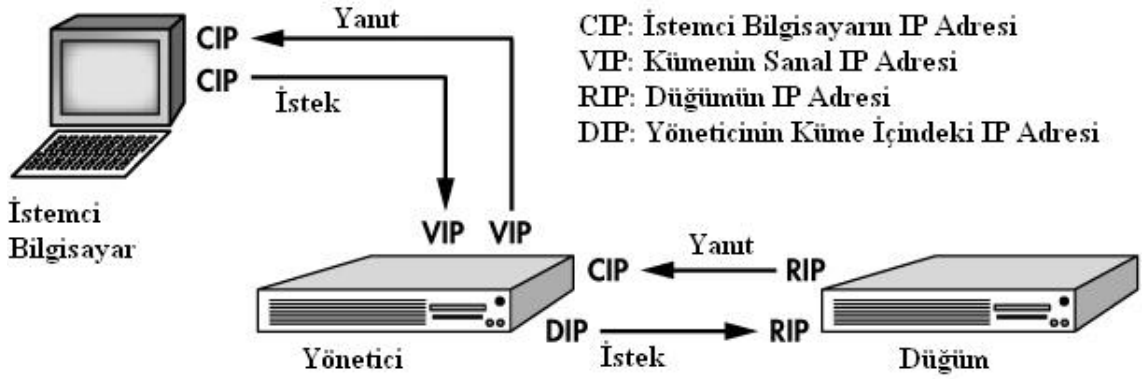
4.6.1. LVS-NAT kümesi hakkında genel bilgiler

LVS-NAT konfigürasyonunda Yönetici Linux çekirdeğinin NAT yeteneğini kullanır.

4.6.1.1. LVS-NAT konfigürasyonun temel özellikleri

LVS-NAT konfigürasyonunun temel özellikleri şunlardır:

- Küme düğümleri ve Yönetici aynı ağa bağlı olmalıdır.
- Düğümlerin IP adresleri 10.X.X.X, 172.16.X.X ve 192.168.0.X gibi private IP adres aralığından seçilmelidir.
- Yönetici istemci bilgisayarlar ve küme düğümleri arası tüm iletişimden sorumludur.
- Küme düğümleri istemci bilgisayarlara yanıt vermek için Yöneticinin iç ağa bağlı olan IP adresini varsayılan ağ geçidi olarak kullanırlar.
- Yönetici ağ port numaralarını değiştirebilir. Yöneticinin sanal IP adresindeki bir porttan alınan istek küme içinde başka bir porta yönlendirilebilir.
- Küme içindeki düğümlerde herhangi bir işletim sistemi kullanılabilir.
- Tek bir Yönetici küme için bir darboğaz olabilir. Yüksek erişilebilirlik sağlamak için Yönetici sayısı da arttırılabilir.



Şekil 4.8. LVS-NAT Kümesi

Düğümlerden gelen tüm yanıtlar Yöneticiden geçmek zorunda olduğu için LVS-NAT kümesinde düğüm sayısı arttıkça sıkıntı yaşanabilir. LVS-NAT kümesinin yönetimi LVS-DR kümesinden daha zordur. Bunun nedeni küme yapısındaki düğümlere erişimin Yönetici tarafından engellenmesidir. Bundan dolayı düğümleri yönetebilmek için önce Yöneticiye erişim sağlanmalı ve yönetici üzerinden düğümlere erişilmelidir.

4.6.1.2. LVS-NAT konfigürasyonunun avantajları ve dezavantajları

LVS-NAT konfigürasyonunun en büyük avantajı gerçek sunucu düğümlerinin TCP/IP protokolü destekleyen herhangi bir işletim sistemi olabilmesidir. Diğer bir avantajı düğümlerin iç ağlarda kullanılmak üzere rezerve edilmiş private IP adres blokları (10.0.0.0/255.0.0.0, 172.16.0.0/255.240.0.0 ve 192.168.0.0/255.255.0.0) kullanabilmesidir. Public IP adresi sadece Yöneticinin Internete bağlı olan ağ kartında bulunan sanal IP adresi için gereklidir.

LVS-NAT konfigürasyonunun dezavantajı genişleyebilirliğinin sınırlı olmasıdır. Yöneticinin yük dengeleme yaptığı düğüm sayısı 20 veya daha fazla olursa yönetici düğüm darboğaz yaşatabilir. Bunun nedeni tüm istek ve yanıt paketlerinin yönetici düğüm tarafından işlenmesi zorunluluğudur. Örneğin TCP paketlerinin ortalama uzunluğunu 536 byte olarak kabul edersek işlemci türüne ve ağ kartına bağlı olarak 60us ye varan bir gecikme meydana gelecektir. Gecikme sayısı düğüm sayısı arttıkça artacak ve buna bağlı olarak maksimum throughput düşecektir [2]. LVS-NAT birçok sunucunun performans isteklerini karşılayabilir. LVS-NAT konfigürasyonunda Yöneticide bir problem olursa tüm küme etkilenebilir. Bunun aşmak için yedekli LVS yöneticileri kullanılabilir.

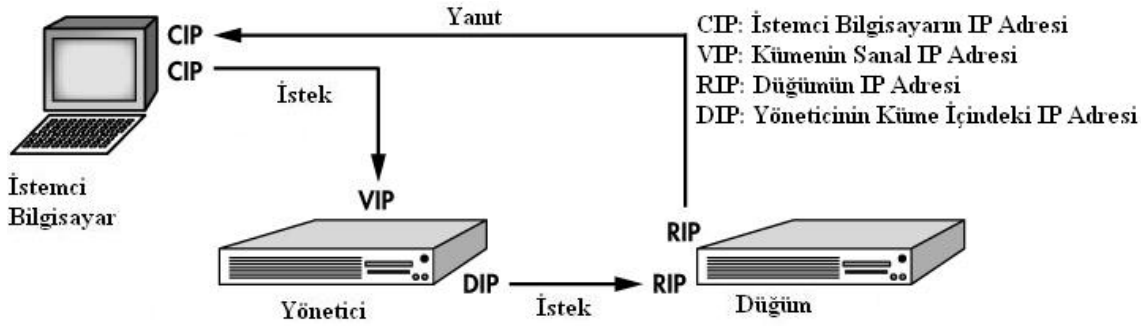
4.6.2. LVS-DR kümesi hakkında genel bilgiler

LVS-DR konfigürasyonunda Yönetici gelen istekleri küme içindeki düğümlere iletir. Ancak küme içindeki düğümler yanıtlarını direkt olarak istemci bilgisayarlara geri gönderir.

4.6.2.1. LVS-DR konfigürasyonun temel özellikleri

LVS-DR konfigürasyonunun temel özellikleri şunlardır:

- Küme düğümleri ve Yönetici aynı ağa bağlı olmalıdır.
- Düğümlerin IP adresleri private IP adresi olmak zorunda değildir. Public IP adresleri de kullanılabilir.
- Sadece istemci bilgisayarlardan gelip düğümlere giden paketler Yöneticiden geçmek zorundadır.
- Küme düğümleri istemci bilgisayarlara gönderdikleri yanıt paketleri için Yöneticiyi varsayılan ağ geçidi olarak kullanmazlar.
- Yönetici ağ port numaralarını değiştiremez.
- Küme içindeki düğümlerde farklı işletim sistemleri kullanılabilir.
- LVS-DR Yöneticisi LVS-NAT Yöneticisinden daha fazla düğümü yönetebilir.



Şekil 4.9. LVS-DR Kümesi

LVS-NAT Yöneticisinin yaptığı ağ port numaralarını değiştirme işlemini LVS-DR Yöneticisi yapamaz ama LVS-DR Linux küme yapıları için en iyi iletme yöntemidir. Bunun nedeni küme düğümlerinin dışarıdan direkt olarak erişilebilmesidir.

LVS-DR yöntemi kümelerin güvenilirliğini arttıran birçok avantaja sahiptir. Bu avantajlar şunlardır;

- Eğer Yönetici de sorun meydana gelirse küme düğümleri dağıtık sunucular haline gelir.

- Küme düğümlerinin her birininin durumunu test etmek ve performansını ölçmek için izleme küme dışındaki bir sunucuda çalışan izleme yazılımı ile yapılabilir.
- Düğümün durumunu hızlıca test etmek için düğüme direkt olarak telnet, ping, ssh işlemleri yapılabilir.
- Eğer yazılımsal bir problemden şüpheleniliyorsa kullanıcılara IP adreslerini kullanarak düğümlere bağlanma izni verilebilir.

4.6.2.2. LVS-DR konfigürasyonunun avantajları ve dezavantajları

LVS-TUN yönteminde olduğu gibi, LVS-DR yönteminde de Yönetici sadece bağlantının yarısı olan istemciden sunucuya giden paketleri işler. Yanıt paketleri ayrı yolları izleyebilirler. Bu da LVS-DR kümesinin genişleyebilirliğini arttırır.

LVS-TUN yöntemiyle karşılaştırıldığında bu yöntemin avantajı tunneling overhead olarak adlandırılan fazlalığı olmamasıdır. LVS-DR yönteminde düğümlerin ağ kartları aynı fiziksel segmentte bulunmalıdır.

4.6.3. LVS-TUN kümesi hakkında genel bilgiler

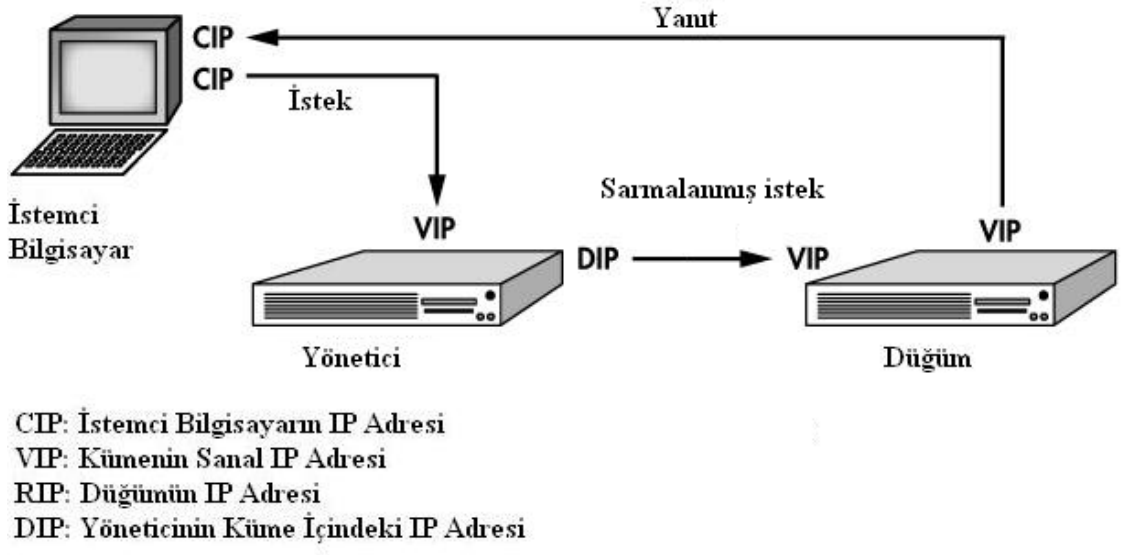
LVS-TUN (IP Tünelleme) bir alt ağdan veya VLAN (sanal ağ) den diğer bir alt ağa veya sanal ağa paketleri iletirken kullanılır. Bu durum paketler başka bir ağdan veya Internetten geçerken de geçerlidir. IP tünelleme yeteneği Linux çekirdeğinden gelir. LVS-TUN iletme yöntemi Yönetici ile aynı ağ bölümünde olmayan küme düğümlerinin de küme ağına yerleştirilmesine izin verir.

LVS-TUN konfigürasyonu LVS-DR yönteminin paket iletme yöntemini iyileştirmektedir. İstemci bilgisayarlarından küme servisleri için gelen istekler paketlenir ve böylece paketler Yönetici ile aynı fiziksel ağ bölümünde olmayan küme düğümlerine iletilebilir.

4.6.3.1. LVS-TUN konfigürasyonun temel özellikleri

LVS-TUN konfigürasyonunun temel özellikleri şunlardır:

- Küme düğümleri Yönetici ile aynı fiziksel ağ bölümünde olmak zorunda değildir.
- Düğümlere verilebilecek IP adresleri private IP adresleri olmamalıdır.
- Yönetici sadece normal olarak istemci ve küme düğümleri arasındaki içeriye doğru olan iletişimi yakalayabilir.
- Gerçek sunucudan istemciye doğru olan dönüş paketleri Yöneticiden geçmemelidir.
- Yönetici ağ port numaralarını değiştiremez.
- Küme içindeki sunucularda sadece IP tünelleme protokolünü destekleyen işletim sistemleri kullanılabilir.



Şekil 4.10. LVS-TUN Kümesi

4.6.3.2. LVS-TUN konfigürasyonunun avantajları ve dezavantajları

LVS-NAT yönteminde istek ve yanıt paketlerinin hepsi yöneticiden geçmek zorundadır. Düğüm sayısı 20 ve üstünde olduğunda LVS-NAT yönteminde performans orıblemleri meydana gelmektedir. Çünkü throughput ağ kartının kapasitesine bağlıdır. Bu da Internet servisleri gibi küçük istek paketli fakat büyük yanıt paketli uygulamalarda darboğaz yaşatmaktadır [2].

LVS-TUN yönteminde yönetici gelen istekleri farklı gerçek sunucu düğümlerine gönderir. Düğümlerden gelen yanıtlar direkt olarak kullanıcılara döner. Dolayısıyla 100 den fazla düğüm kullanılsa bile LVS-TUN yönteminde darboğaz olmaz. Bu yöntemde yöntecinin ağ kartı 100Mbps full-duplex çalışsa bile maksimum throughput 1 Gbps hızına ulaşabilir.

LVS-TUN yöntemi çok yüksek performanslı kümeler oluşturmak için kullanılabilir. Özellikle sanal proxy sunucular için ideal yöntemdir. Çünkü proxy sunucular istekleri aldığında direkt olarak Internetten sayfaları alıp kullanıcılara döndürür.

4.6.4. LVS zamanlama yöntemleri

Zamanlama yöntemleri küme düğümleri arasında iş yükünün dağıtımını belirleyen mekanizmalardır. Yönetici bir küme servisi için bir istek aldığı zaman hangi düğümün bu görevi yerine getireceğine karar vermelidir.Yöneticinin karar verirken kullandığı zamanlama yöntemleri temel olarak sabit zamanlama ve dinamik zamanlama olarak iki kategoriye ayrılır.

4.6.4.1. Sabit zamanlama yöntemleri

Sabit zamanlama yöntemlerinde Yönetici düğüme daha önceden atadığı bağlantıların kaç tanesinin aktif olduğunu kontrol etmeksizin gelen istek için kullanacağı küme düğümünü belirler.

Sabit zamanlama yöntemleri şunlardır;

- Round-robin (RR)
- Weighted round-robin (WRR)
- Destination hashing
- Source hashing

Round-robin (RR) yöntemi: Yeni bir istek alındığında Yönetici sunucu listesinde sıradaki sunucuyu seçer. Sonsuz bir döngü içinde mevcut sunucular arasında döner.

Weighted round-robin (WRR): Yönetici bir ağırlık/sıralama bilgisine dayanarak küme düğümlerine iş atar. Ağırlık bilgisi düğümün ne kadar işlem gücü sağlayabileceğini gösterir. Bu teknik round-robin tekniği ile beraber kullanılarak yeni bir iş geldiğinde sıradaki küme düğümünü seçmeyi sağlar. Örneğin ağırlığı 2 olan bir sunucu ağırlığı 1 olan sunucunun 2 katı kadar iş alır. Eğer sunucu ağırlığı 0 yapılırsa sunucuya yeni iş atanmaz.

Destination hashing: Bu yöntemde aynı IP adresi için olan istekler kümedeki aynı sunucuya gönderilir. Bu yöntem kümedeki sunucular cache veya proxy sunucu olduğu zaman kullanışlıdır.

Source hashing: Bu yöntem Yöneticinin yanıt paketlerinin isteklerin geldiği router veya firewalla geri gönderilmesi gerektiğinde kullanılır. Bu zamanlama yöntemi normal olarak sadece Yöneticinin birden fazla fiziksel ağ bağlantısı olduğunda kullanılır. Böylece Yönetici hangi firewall veya routerın yanıt paketlerini geri göndereceğini bilebilir.

4.6.4.2. Dinamik zamanlama yöntemleri

Dinamik zamanlama yöntemleri gelen iş yükü üzerinde daha fazla kontrol sağlarlar. Bu yöntemin mahsuru Yöneticinin üzerinde biraz ekstra işlem yükü gerektirmesidir. Dinamik zamanlama yöntemleri kullanıldığında Yönetici her bir küme düğümü için aktif olan ve aktif olmayan bağlantıların sayısını takip eder ve bu bilgiyi kullanarak bir küme servisi için yeni bir istek geldiğinde hangi küme düğümünü kullanacağına karar verir.

Aktif bir bağlantı bir TCP ağ oturumu olup istemci bilgisayar ve küme düğümü birbirine veri gönderdiği sürece açık kalır ve ESTABLISHED durumunda bulunur. Aktif olmayan bir bağlantı ise ESTABLISHED durumunda bulunmaz. Eğer TCP inactivity timeout oluşursa veya istemci bilgisayar FIN paketi gönderip bağlantıyı koparırsa LVS bağlantıyı IPVS tablosunda geçici bir süre için tutar. Bunun nedeni sonraki paketlerin TCP bağlantısını tekrar kurabileceği ihtimalidir.

Dinamik zamanlama yöntemlerinden bazıları şunlardır;

- Least-connection (LC)
- Weighted least-connection (WLC)
- Shortest expected delay (SED)
- Never queue (NQ)
- Locality-based least-connection (LBLC)
- Locality-based least-connection with replication scheduling (LBLCR)

Least-Connection (LC): LC zamanlama yönteminde küme düğümlerinden birinde çalışan bir servis için Yöneticiye bir istek geldiğinde, Yönetici aktif olan ve aktif olmayan bağlantıların sayılarına bakarak hangi küme düğümünün isteği alacağına karar verir.

Yönetici kümedeki her düğüm için düğümdeki aktif bağlantıların sayısını 256 ile çarpar. Düğüme gelen aktif olmayan bağlantıların sayısını da bu değere ekleyerek overhead değerini hesaplar. Servis için yeni bir istek geldiğinde en düşük overhead

değerine sahip olan düğüm isteği alır. Eğer kümedeki tüm düğümler için overhead değerleri aynı ise, IPVS tablosunda bulunan ilk düğüm seçilir.

Weighted Least-Connection (WLC): WLC yöntemi LC yöntemini bir ağırlık/sıralama değeri ile birleştirir. WLC varsayılan yöntemdir. WLC yönteminin amacı farklı işlem kapasitesine sahip küme düğümlerinden etkili bir şekilde faydalanmaktır.

Yönetici küme servisi için gelen bir isteği atamadan önce şu işlemleri yapar. Önce her bir küme için overhead değeri hesaplar. Sonra bu değeri küme düğümlerine atanmış olan ağırlık değerine böler. Elde edilen değer WLC değeridir. Yeni bir istek geldiğinde en küçük WLC değerine sahip olan düğüm isteği alır.

Eğer küme düğümlerinin hepsinin WLC değeri aynı ise küme düğümleri listesinde bulunan ilk düğüm seçilir.

Shortest Expected Delay (SED): SED LVS zamanlama yöntemlerine yeni eklenen bir yöntemdir. TCP servisi kullanan servisler için WLC yöntemine göre ufak bir üstünlüğü vardır.

SED yönteminde her bir küme düğümü için aktif bağlantıların sayısına 1 eklenerek overhead değeri hesaplanır. Bu değer küme düğümlerine atanmış olan ağırlık değerine bölünerek SED değeri hesaplanır. En küçük SED değerine sahip olan küme düğümü isteği alır.

SED zamanlama yönteminde iki noktaya dikkat etmek gereklidir. Overhead değeri hesaplanırken aktif olmayan bağlantılar dikkate alınmaz. Yeni gelen bir bağlantı atandıktan sonra bu overhead değerine 1 eklenir.

Never Queue (NQ): NQ zamanlama yöntemi SED yöntemine yeni bir özellik eklemiştir. Eğer bir küme düğümünün aktif bağlantısı yoksa SED değerlerine bakılmaksızın bu düğüme yeni gelen istek atanır.

Locality-Based Least-Connection (LBLC): Yönetici dışarıya doğru olan trafiği de saydam proxy sunucu kümesine yönlendirebilir. Bu konfigürasyonda küme düğümleri saydam proxy veya web cache sunucular olup istemci bilgisayarlar ve Internet arasında bulunurlar.

LBLC zamanlama yöntemi kullanıldığında Yönetici belirli bir IP adresini hedefleyen tüm istekleri aynı saydam proxy sunucuya gönderir. Internetteki bir web sunucu için bir istek geldiğinde, Yönetici bu hedef IP adresine servis vermek için bir proxy sunucu seçecektir. Bu seçimi WLC zamanlama yönteminin değiştirilmiş bir versiyonuyla yapacaktır. Bundan sonra aynı IP adresine giden istekler aynı proxy sunucu üzerinden gidecektir. Yönetici atanmış olan sunucunun WLC değerinin yarısı olan bir düğüm görene kadar bu işleme devam eder.

Locality-Based Least-Connection with Replication Scheduling (LBLCR): LBLCR zamanlama yöntemi her bir hedef IP adresine servis veren proxy sunucuların listesini tutarak LBLC metodunu iyileştirmeyi hedefler. Yeni bir bağlantı isteği geldiğinde Yönetici bu sunucu setinden en az aktif bağlantıya sahip olan proxy sunucuyu seçecektir.

Bir istemci bilgisayardan yeni bir bağlantı isteği geldiğinde eğer bir küme düğümünün WLC değeri sunucu setindeki en az yüklü olan sunucunun WLC değerinin yarısı ise proxy sunucu bu sete eklenir.

Eğer sunucu setindeki sunucu listesi son 6 dakika içinde değişmemişse proxy sunucular setten çıkarılır. Bu gerçekleştiğinde Yönetici en fazla aktif bağlantıya sahip olan sunucuyu bu setten çıkartır. Bu sunucu listeden çıkarılsa bile üzerinde mevcut olan aktif bağlantılara hizmet vermeye devam edecektir.

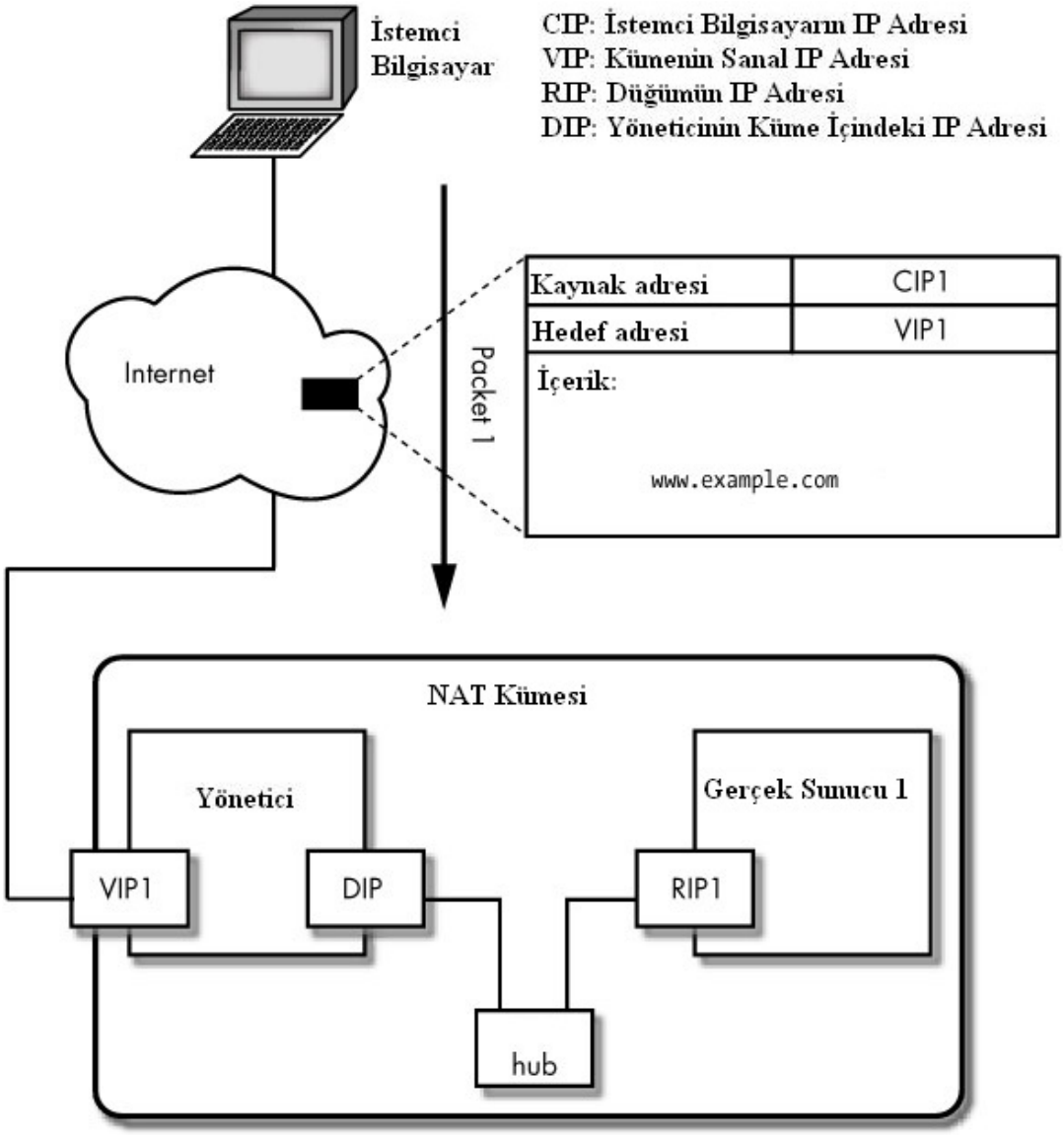
4.7. LVS-NAT Kümesi

LVS-NAT Kümesinde istemci bilgisayarlar küme servislerine LVS-NAT Yöneticisi üzerinden erişirler.

Aşağıdaki şekilde görülebileceği gibi istemci bilgisayar Internet aracılığıyla kümenin sanal IP adresine bir istek göndererek küme ile ağ iletişimini başlatmaktadır. Paketin kaynak adresi CIP1 hedef adresi ise VIP1 dir. Paketin içeriğinde istemci

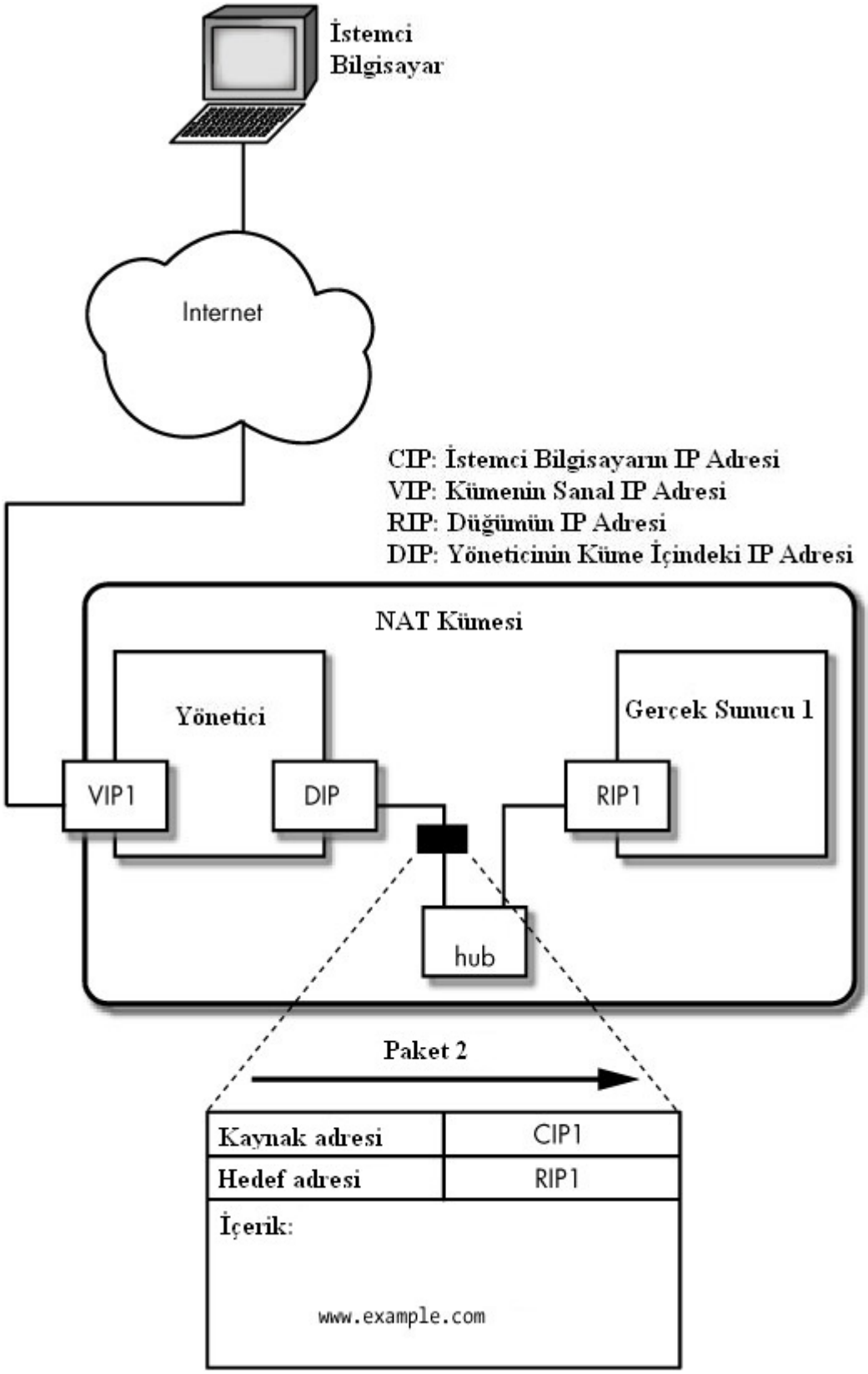
bilgisayardan gelen bir web sayfası içeriği görüntüleme isteği olan “HTTP GET request” dir.

Paket Yönetici düğüme ulaştığında Yöneticide çalışan LVS kodu zamanlama yöntemlerinden birisini kullanarak hangi sunucuya istek atanacağına karar verir. Yönetici bu atama sırasında gelen paketin veri yükünün içeriğini incelemez ve değiştirmez. Yönetici sadece paketin hedef adresini ve hedef port numarasını değiştirir. Sadece LVS-NAT yönteminde Yönetici ağ port numaralarını değiştirilir.



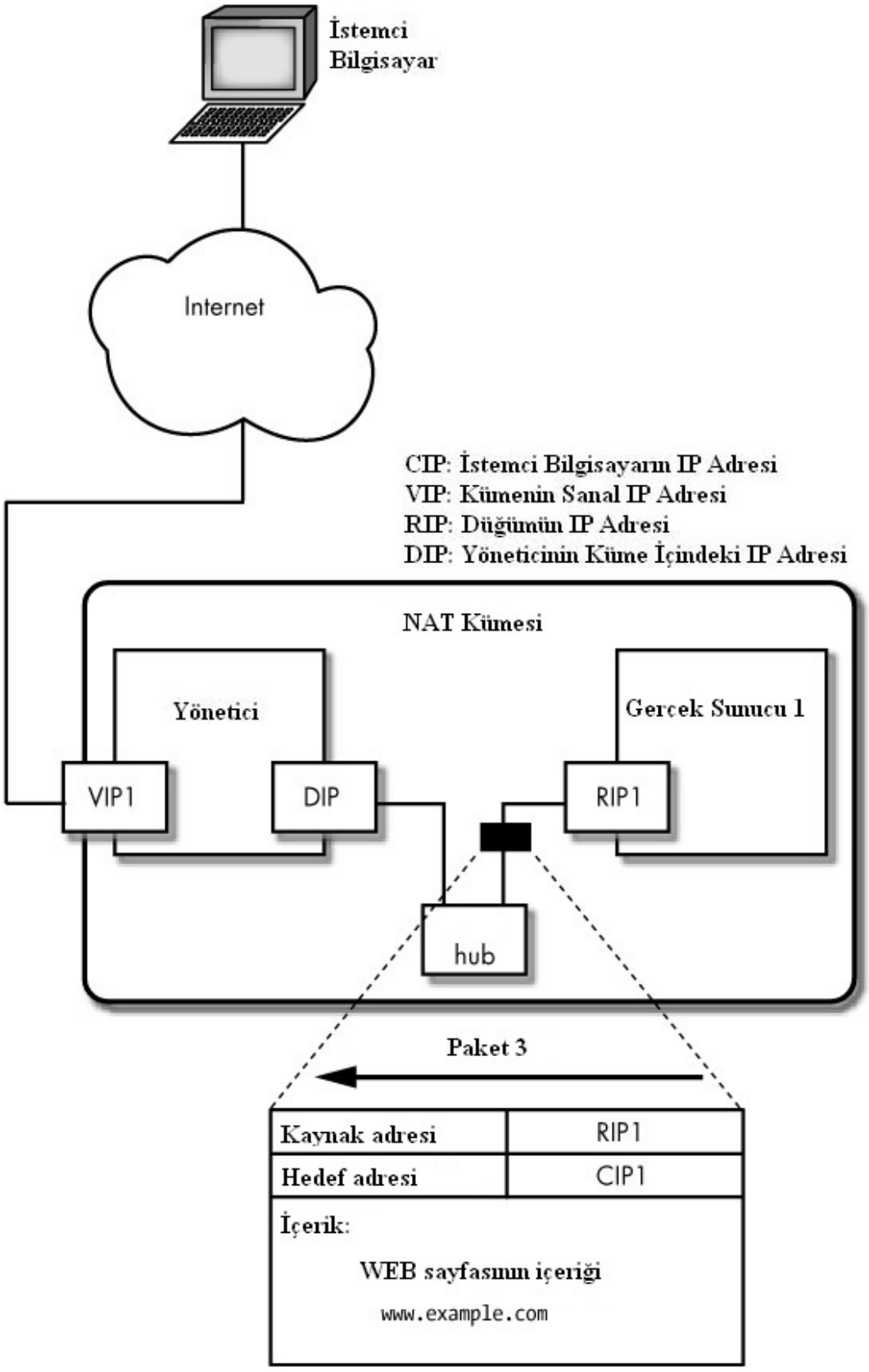
Şekil 4.11. Paket 1’de istemci bilgisayarın LVS-NAT kümesine gönderdiği istek

Paket aşağıdaki şekilde görüldüğü gibi Yönetici tarafından gerçek sunucuya yeni bir hedef adresi ve gerekiyorsa yeni bir port numarasıyla iletilir. Paket 1'deki CIP (istemci bilgisayar IP adresi) ve veri yükü (HTTP GET request) değişmeden kalır. Hedef adresi değişir. Yönetici hedef adres değerini gerçek sunucunun IP adresi (RIP) ile değiştirir.



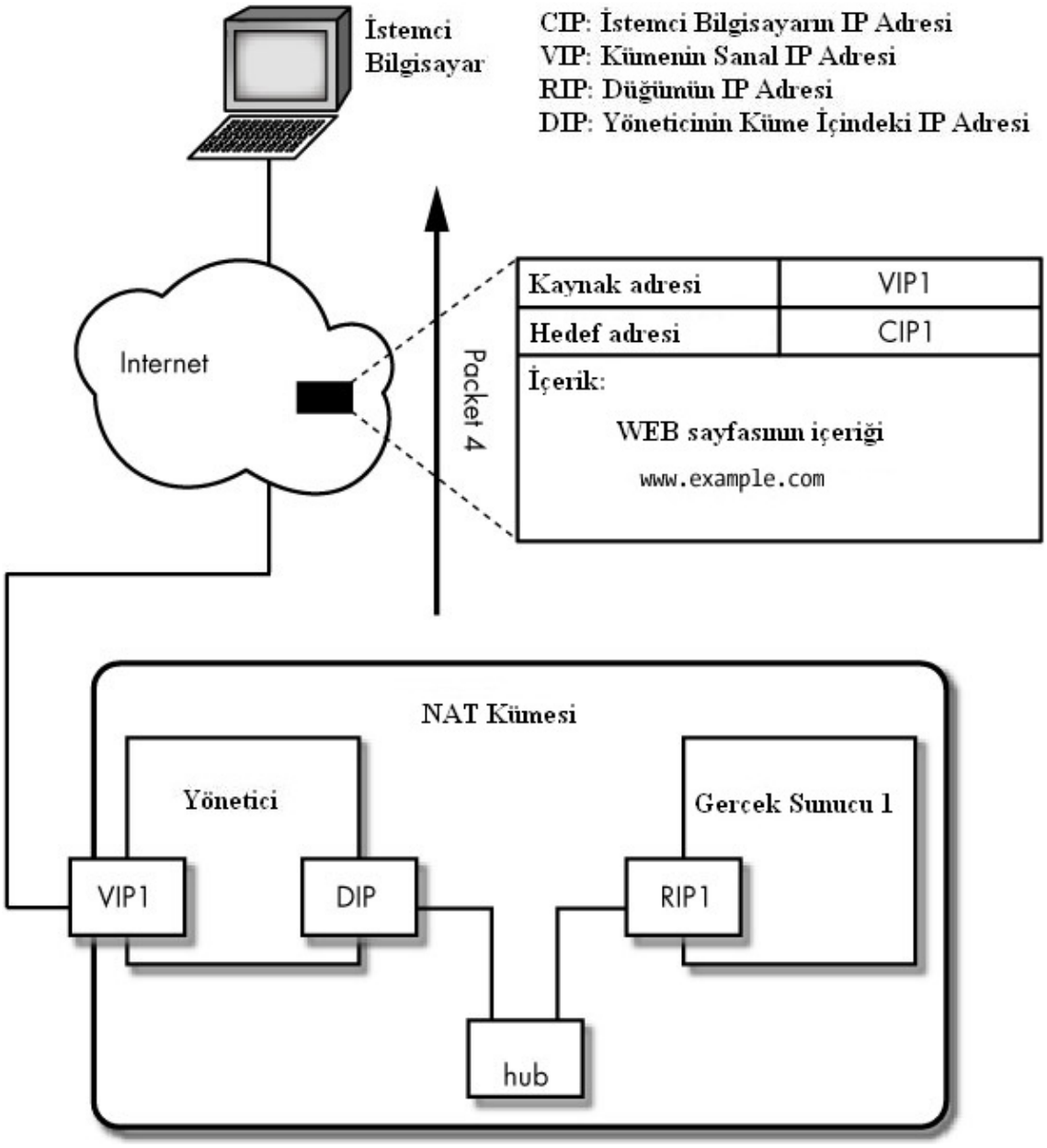
Şekil 4.12. Paket 2'nin Yönetici tarafından küme düğümüne isteği iletmesi

Aşağıdaki şekilde görüldüğü gibi Paket 2 sunucuya vardığında HTTP sunucu web sayfasının içeriği ile yanıt gönderir. İstek istemci bilgisayarın IP adresi olan CIP1'den geldiği için Paket 3'te hedef IP adresi CIP1, kaynak adresi ise RIP1 olacaktır. LVS-NAT yönteminde gerçek sunucu için varsayılan ağ geçidi Yönetici düğümün IP adresidir. Dolayısıyla yanıt paketi Yönetici aracılığı ile yönlendirilir.



Şekil 4.13. Yönetici aracılığıyla küme düğümü Paket 3 ile yanıt göndermektedir.

Küme ağından Yöneticiye gelen Paket 3'ün içeriği istemci tarafından istenen web sayfasıdır. Yönetici Paket 3'ü aldığında paketi çekirdeğine gönderir. LVS yazılımı paketin kaynak adresini tekrar yazar ve RIP1 değerini VIP1 değeri ile değiştirir. Veriyi değiştirmez. Yönetici paketi Internet üzerinden istemciye iletir.



Şekil 4.14. Yönetici yanıt paketi olan Paket 4'ü istemci bilgisayara iletmektedir.

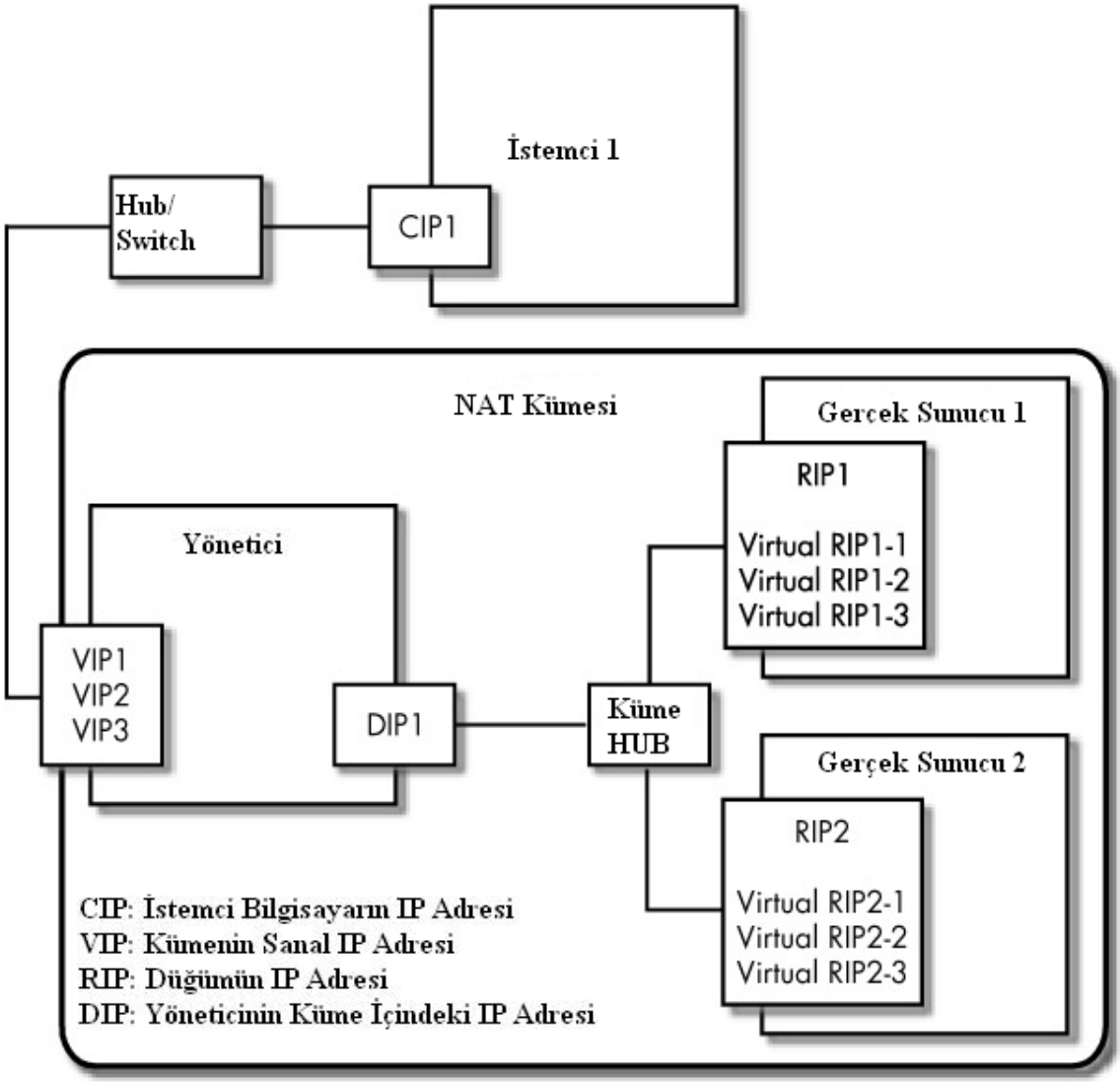
Paket 4 LVS-NAT kümesinden gelen yanıt paketidir. Paket 4 kaynak adresi olarak VIP1, hedef adresi olarak ise CIP1 içerir. Yönetici tek bir sanal IP adresinin arkasında küme düğümlerinin gerçek IP adreslerini gizler.

Bu iletişimde dikkat edilmesi gereken hususlar şunlardır;

- Yönetici küme ağında yönlendirici olarak görev üstlenir.
- Küme düğümleri Yöneticinin küme ağına bağlı olan ağ kartının IP adresi varsayılan ağ geçidi olarak kullanır.
- Yönetici kümeye hedeflenmiş tüm gelen paketleri alır.
- Yönetici küme düğümlerinin yerine yanıt vermektedir.
- Yönetici küme düğümlerinin ağ adreslerini gizlediği için kümeye ulaşmanın tek yöntemi yöneticinin dış ağa bağlı olan sanal IP adresidir.

4.7.1. LVS-NAT gerçek sunucularında sanal IP adresleri

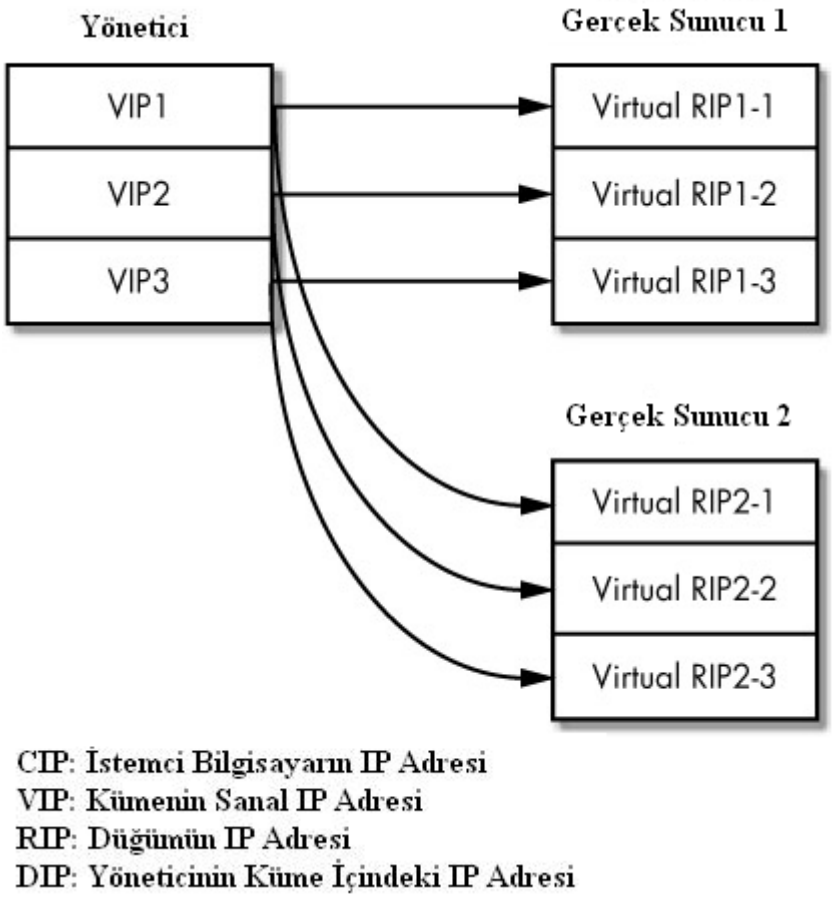
Eğer istemci bilgisayarlara farklı IP adreslerinde farklı servislere sunulmak isteniyorsa LVS-NAT kümesindeki küme düğümlerinin de birden fazla IP adresleri olacaktır. Aşağıdaki şekilde görüldüğü gibi Yönetici farklı sanal gerçek sunucu IP adresleri kullanarak paketleri küme düğümleri olan gerçek sunuculara gönderebilir. Bu durumda istemci bilgisayarlar Yönetici de konfigüre edilmiş olan sanal IP adresleri ile servislere ulaşacaktır. Yönetici uygun sanal gerçek sunucu IP adreslerini kullanarak paketlerini kümeye gönderecektir.



Şekil 4.15. Çoklu Sanal IP Adresli LVS-NAT Kümesi

Bu yapı farklı IP adreslerinde farklı kullanıcı tiplerini yük dengeleme işlemine tabi tutmak için kullanılabilir. Örneğin muhasebe bölümü kullanıcıları bir IP adresini, müşteri hizmetleri diğer IP adresini kullanabilir.

Aşağıdaki şekilde Yönetici üç sanal IP adresini kullanarak istemci bilgisayarlardan gelen ağ isteklerini almaktadır. Yönetici LVS-NAT yöntemini kullanarak kümedeki gerçek sunuculardaki sanal gerçek sunucu IP adreslerinden birisini seçecektir.



Şekil 4.16. Çoklu Sanal IP Adreslerinin Kullanımı

Yöneticideki sanal IP adresleri gibi sanal gerçek sunucu IP adresleri de IP aliases veya ikincil IP adresleri kullanılarak yaratılabilir. Yöneticideki VIP1 adresine gelen paket Yönetici tarafından gerçek sunucu 1 de bulunan RIP1-1 adresine veya gerçek sunucu 2 de bulunan RIP2-1 adresine iletilecektir. Gerçek sunucudan geriye dönen yanıt paketi Yönetici tarafından geri gönderilecektir. Geri dönen paket istemci tarafından VIP1 adresinden gönderilmiş olarak kabul edilecektir.

4.7.2. LVS-NAT web kümesi

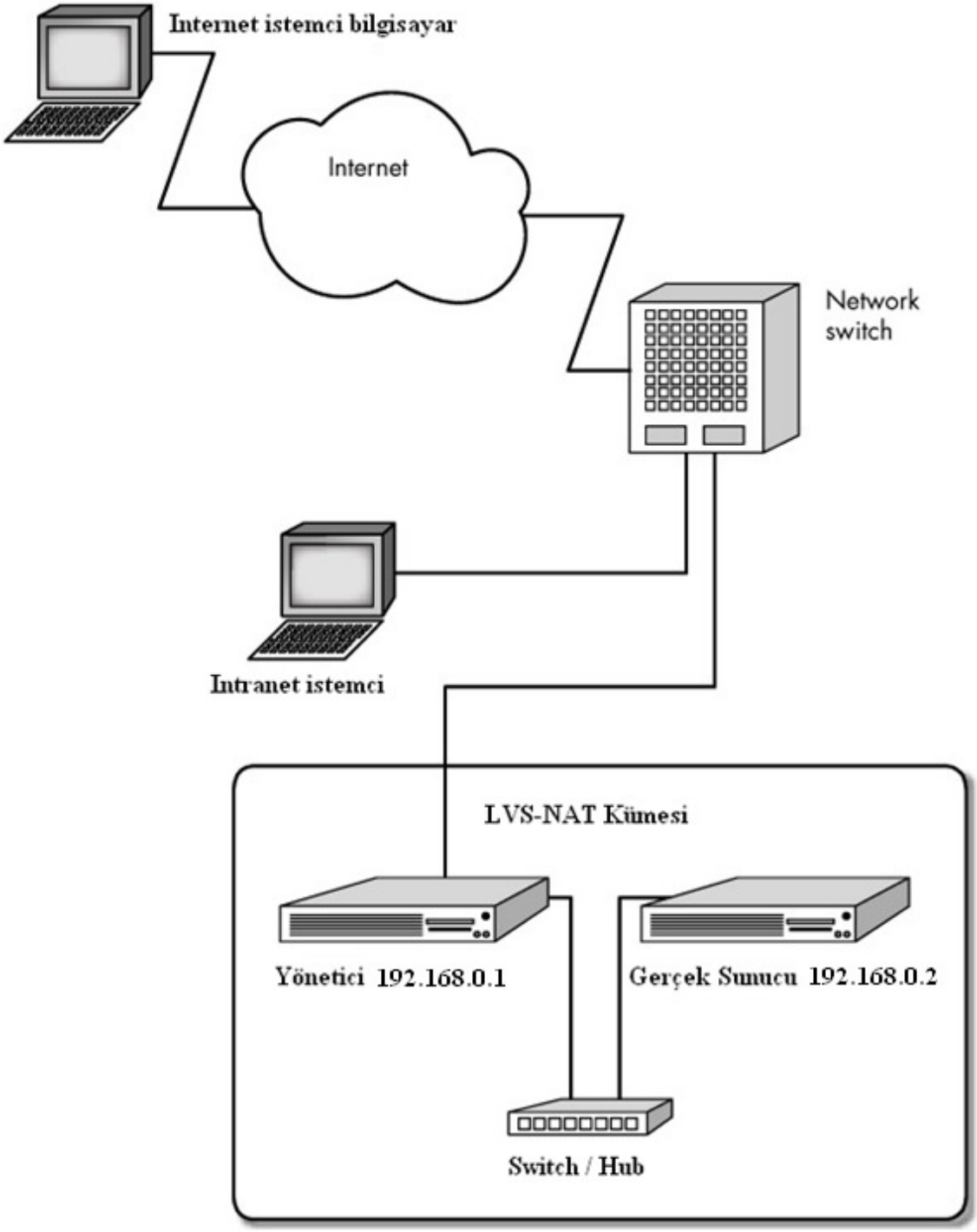
LVS-NAT Web kümesinin amacı web servislerinde yük dengeleme yöntemiyle yüksek performans ve hata toleransı elde etmektir.

LVS-NAT kümesi kurulumundan önce küme ağında kullanılacak IP adres aralığı belirlenmelidir. Küme ağında kullanılacak IP adreslerinin küme ağı dışında bir önemi yoktur. IP adres aralığı belirlenirken RFC 1918'de belirtilen adresler kullanılmalıdır.

192.168.0.0	LVS-NAT küme ağı
192.168.0.255	LVS-NAT küme broadcast adresi
255.255.255.0	LVS-NAT küme subnet mask

Tablo 4.1. LVS-NAT Web Kümesinde Kullanılacak Private Adres Aralığı

Küme Yöneticisinin kullandığı sanal IP adresi kümeye ulaşılırken kullanılacak IP adresidir. Bu IP adresi kümenin Internet bağlantısını sağlayan ağ kartına atanacak olan IP adresidir. Bu IP adresi Internet Servis Sağlayıcıdan alınabilir.



Şekil 4.17. LVS-NAT Web Kümesi

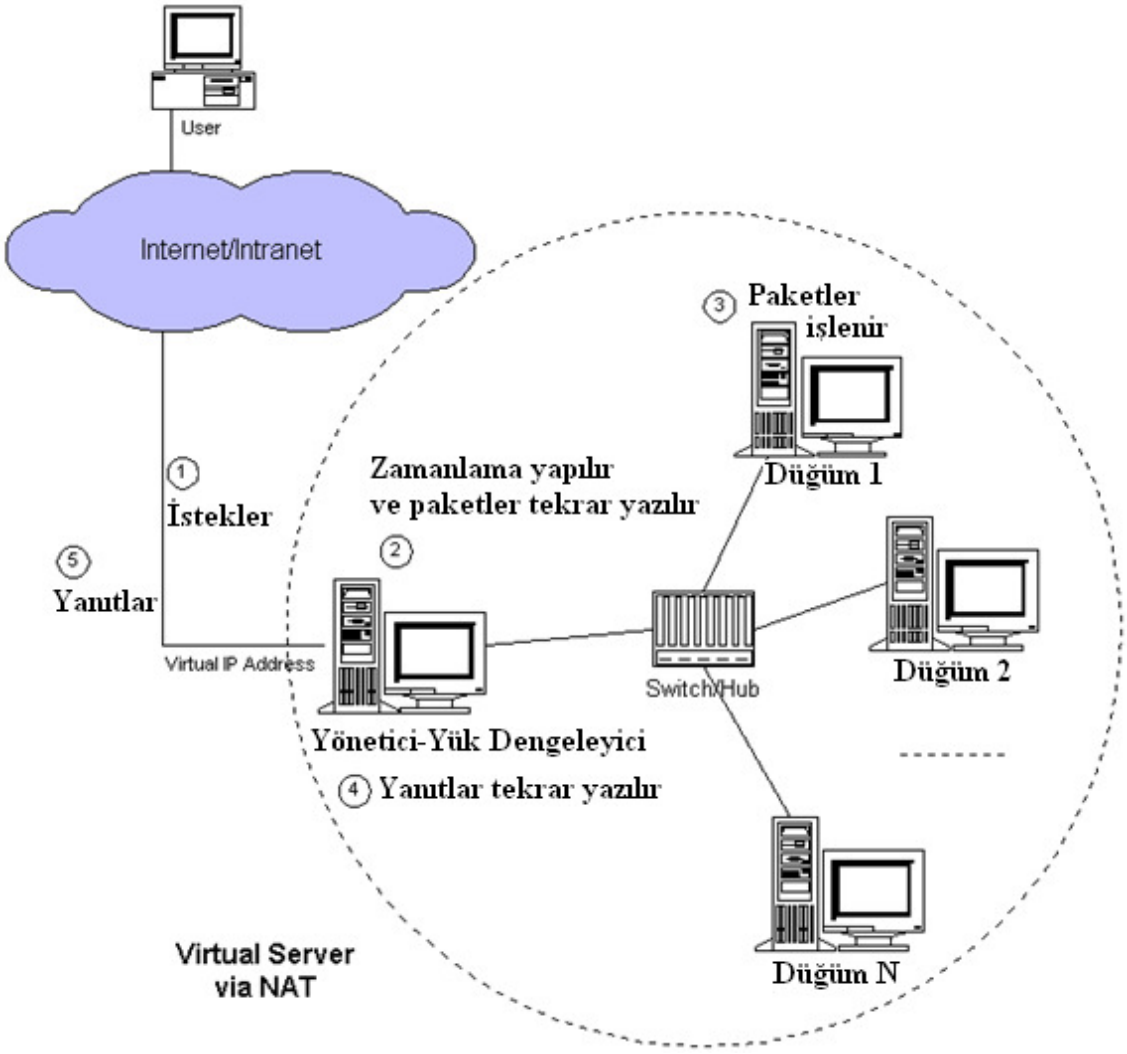
4.7.3. Virtual Server via NAT yönteminin detaylı incelenmesi ve konfigürasyonu

Virtual server via NAT yönteminde düğümlerin kullanabileceği IP adresleri private IP adres aralıkları olan 10.0.0.0/255.0.0.0, 172.16.0.0/255.240.0.0 veya 192.168.0.0/255.255.0.0 aralıklarından olmalıdır.

NAT (Network address translation) yönteminde IP adresleri bir gruptan diğer bir gruba dönüştürülür. Adres dönüştürme N→N olduğunda bu yöntem static network address translation olarak adlandırılır. Adres dönüştürme M→N olduğunda yöntem dynamic network address translation olarak adlandırılır. Network address port translation temel NAT işlemine yapılan bir eklenti olup birçok IP adresi ve bu adreslere ait olan TCP/UDP portları tek bir IP adresine ve TCP/UDP portlarına dönüştürülür. Bu yöntem N→1 dönüştürmedir. Linux IP Masquerading tekniği bu yönteme göre çalışır.

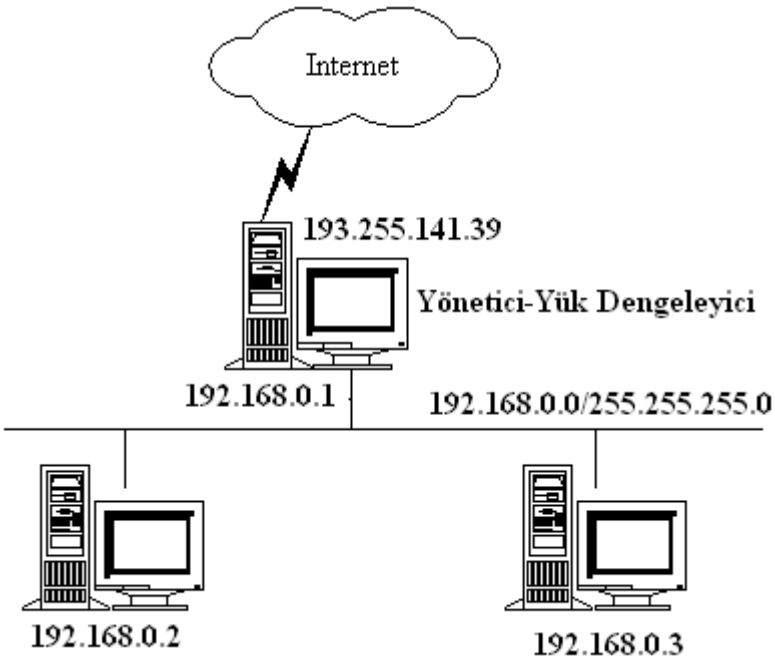
4.7.3.1. Virtual Server via NAT yöntemi nasıl çalışır?

Yönteme ait olan bir örnek konfigürasyon aşağıdaki şekilde görülmektedir.



Şekil 4.18. Virtual Server via NAT Yöntemiyle Küme Oluşturulması

Kullanıcı küme tarafından sunulan bir servise erişmek istediğinde sanal IP adresi için olan istek paketi yöneticiye geldiğinde yönetici paketin hedef adresine ve port numarasına bakar. Eğer paket sanal sunucu tablosundaki sanal servislerden birisine uyarsa zamanlama yöntemine göre sunucu düğümlerinden birisi seçilir. Bağlantı bağlantı izleme tablosuna eklenir. Daha sonra paketin hedef adres ve port numarası seçilen sunucuya göre tekrar yazılır ve paket sunucuya iletilir. Eğer içeriye doğru gelen paket bu bağlantıya ait ise paket tekrar yazılır ve sunucuya iletilir. Yanıt paketleri geldiğinde yönetici kaynak adresi ve port numarasına sanal servisinkileri yazar. Bağlantı sona erdiğinde veya timeout durumuna düştüğünde bağlantı izleme tablosundan silinir [5].



Şekil 4.19. LVS-NAT Yönteminde Yönetici-Düğüm Paket Aktarımları

Gerçek sunucu düğümünde TCP/IP protokolüni destekleyen herhangi bir işletim sistemi çalışabilir. Varsayılan ağ geçidi sanal yöneticinin iç IP adresi olan 192.168.0.1 olmalıdır. ipfwadm kullanılarak sanal sunucu olan yöneticinin sunucu düğümlerinden gelen paketleri kabul etmesi sağlanmalıdır. Bu işlem aşağıdaki şekilde yapılır.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipfwadm -F -a m -S 192.168.0.0/24 -D 0.0.0.0/0
```

Protocol	Virtual IP Address	Port	Real IP Address	Port	Weight
TCP	193.255.141.39	80	192.168.0.2	80	1
			192.168.0.3	8000	2
TCP	193.255.141.39	21	192.168.0.3	21	1

Tablo 4.2. Linux Virtual Server Yönetici-Düğüm İlişkileri

193.255.141.39 IP adresine Port 80 üzerinden gelen tüm istekler 192.168.0.2 IP adresli düğüme (Port 80) ve 192.168.0.3 IP adresli düğüme (Port 8000) gönderilir. 193.255.141.39 Port 21 için gelen tüm trafik ise 192.168.0.3 IP adresi Port 21 iletilir.

Paket yazım işlemi aşağıdaki şekilde gerçekleşir.

Web servisi için gelen paket aşağıdaki gibi kaynak ve hedef adreslerine sahiptir.

SOURCE 193.255.155.125:3456 DEST 193.255.141.39:80

Yönetici düğümlerden birisini seçer. Paket tekrar yazılır ve düğüme iletilir.

SOURCE 193.255.155.125:3456 DEST 192.168.0.3:8000

Yanıtlar yöneticiye geri döner.

SOURCE 192.168.0.3:8000 DEST 193.255.155.125:3456

Paketler yöneticiye ulaşır ve istemciye aşağıdaki şekilde gönderilir.

SOURCE 193.255.141.39:80 DEST 193.255.155.125:3456

4.7.3.2. Virtual Server işlemi için çekirdeğin yapılandırılması

Öncelikle Linux çekirdek sürümlerinden kararlı bir tanesi indirilir ve virtual server yaması çekirdeğe uygulanır. 2.6 ve üstü çekirdekler için yapılması gereken bir işlem yoktur.

2.0.36 ve sonrası çekirdekler için VS yamasının uygulanması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers
```

```
Networking options --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    ....
    [*] IP: masquerading
    ....
    [*] IP: ipportfw masq & virtual server support
```

Seçim yapıldıktan sonra zamanlama yöntemi seçilir.

```
    Virtual server scheduling algorithm
    (X) WeightedRoundRobin
    ( ) LeastConnection
    ( ) WeightedLeastConnection
```

Son olarak çekirdek tekrar oluşturulur, güncellenir ve sistem tekrar başlatılır.

Daha sonra ippfvsadm aracı kurulur ve aşağıdaki komutlar ile yapılandırılır.

```
   ippfvsadm -A -t 193.255.141.39:80 -R 192.168.0.2:80 -w 1
    ippfvsadm -A -t 193.255.141.39:80 -R 192.168.0.3:8000 -w 2
    ippfvsadm -A -t 193.255.141.39:21 -R 192.168.0.3:21
```

2.2.x çekirdeği için IPVS yaması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network firewalls
    ...
    [*] IP: forwarding/gatewaying
    ...
    [*] IP: firewalling
    ...
    [*] IP: masquerading
    ...
    [*] IP: masquerading virtual server support (EXPERIMENTAL)
    (12) IP masquerading table size (the Nth power of 2)
    <M> IPVS: round-robin scheduling
    <M> IPVS: weighted round-robin scheduling
    <M> IPVS: least-connection scheduling
    <M> IPVS: weighted least-connection scheduling
    <M> IPVS: locality-based least-connection scheduling
    <M> IPVS: locality-based least-connection with replication
scheduling
```

Çekirdek oluşturulduktan sonra sistem çekideği güncellenir ve sistemtekrar başlatılır. Daha sonra ipvsadm programı kurulur.

Yöneticinin maskelenmiş paketleri iletmesi için aşağıdaki komutlar kullanılır.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipchains -A forward -j MASQ -s 192.168.0.0/24 -d 0.0.0.0/0
```

Sanal servis eklenir ve zamanlayıcıya bağlanır.

```
ipvsadm -A -t 193.255.141.39:80 -s wlc  (Weighted Least-
Connection scheduling)
ipvsadm -A -t 193.255.141.39:21 -s wrr  (Weighted Round
Robing scheduling )
```

Gerçek sunucu düğümü eklenir ve iletim yöntemi seçilir.

```
ipvsadm -a -t 193.255.141.39:80 -r 192.168.0.2:80 -m
ipvsadm -a -t 193.255.141.39:80 -r 192.168.0.3:8000 -m -w 2
ipvsadm -a -t 193.255.141.39:21 -r 192.168.0.2:21 -m
```

2.4.x çekirdeği için IPVS yaması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network packet filtering (replaces ipchains)
    [ ]   Network packet filtering debugging
    ...
      IP: Netfilter Configuration  --->
      IP: Virtual Server Configuration  --->
       <M> virtual server support (EXPERIMENTAL)
       [*]   IP virtual server debugging
       (12)   IPVS connection table size (the Nth power of 2)
       --- IPVS scheduler
       <M>   round-robin scheduling
       <M>   weighted round-robin scheduling
       <M>   least-connection scheduling scheduling
       <M>   weighted least-connection scheduling
       <M>   locality-based least-connection scheduling
       <M>   locality-based least-connection with replication
scheduling
       <M>   destination hashing scheduling
       <M>   source hashing scheduling
```

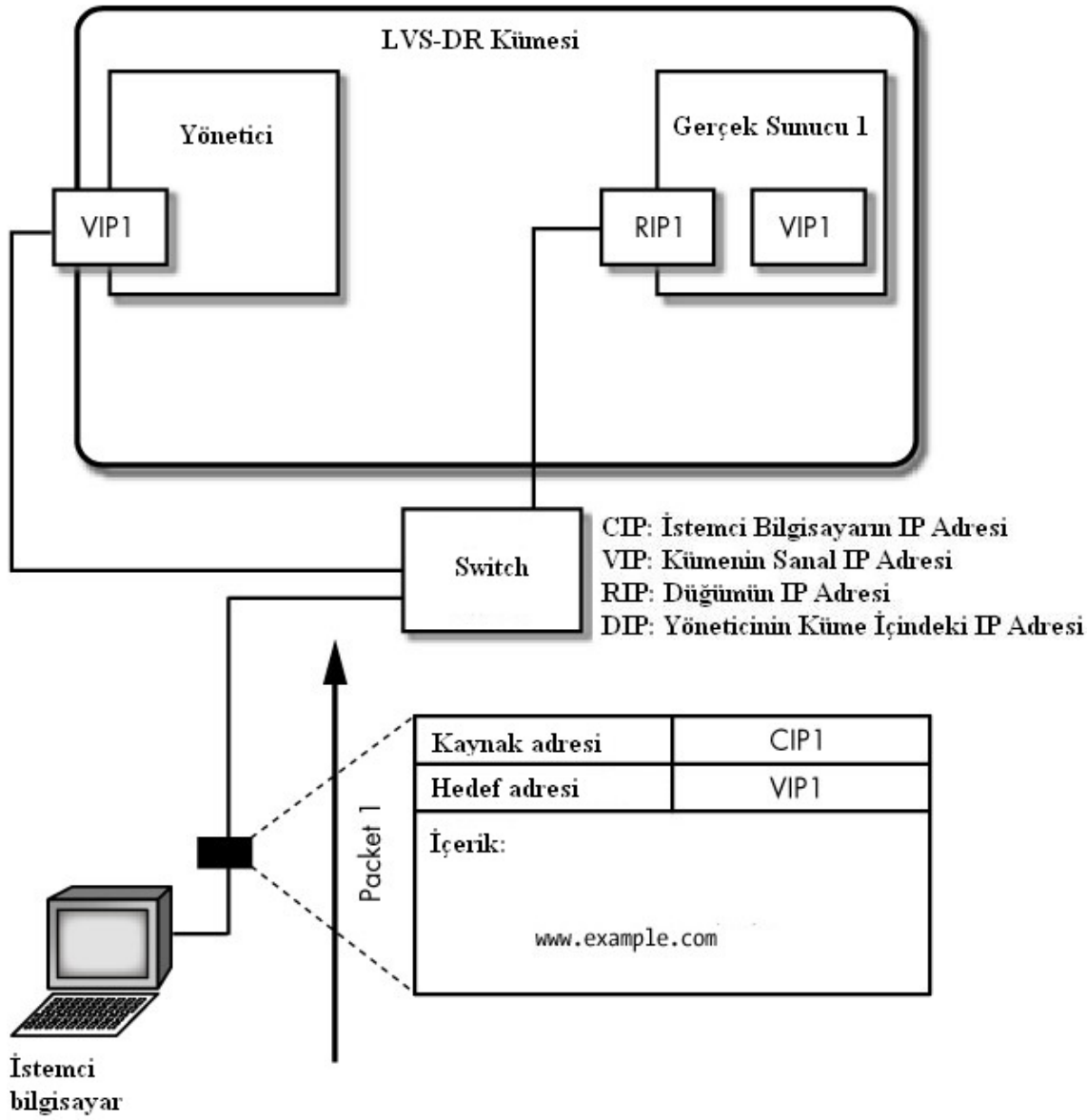
```
--- IPVS application helper
<M>   FTP protocol helper
```

Diğer komutlar 2.2 sürümü çekirdeklerle aynıdır.

4.8. LVS-DR Kümesi

LVS-DR kümesinde kümedeki tüm düğümler ve yönetici aynı sanal IP adresi ile konfigüre edilir. LVS-DR kümesinde de paketler önce Yöneticiye gelir. Yönetici istemci bilgisayarlardan gelen yükü LVS zamanlama yöntemlerinden birisini kullanarak düğümler arasında dağıtır.

LVS-DR yapısı aşağıdaki şekilde görülmektedir.



Şekil 4.20. İstemcinin LVS-DR kümesine istek göndermesi

LVS-DR kümesinde LVS-NAT kümeleri gibi birden fazla sanal IP kullanabilir.

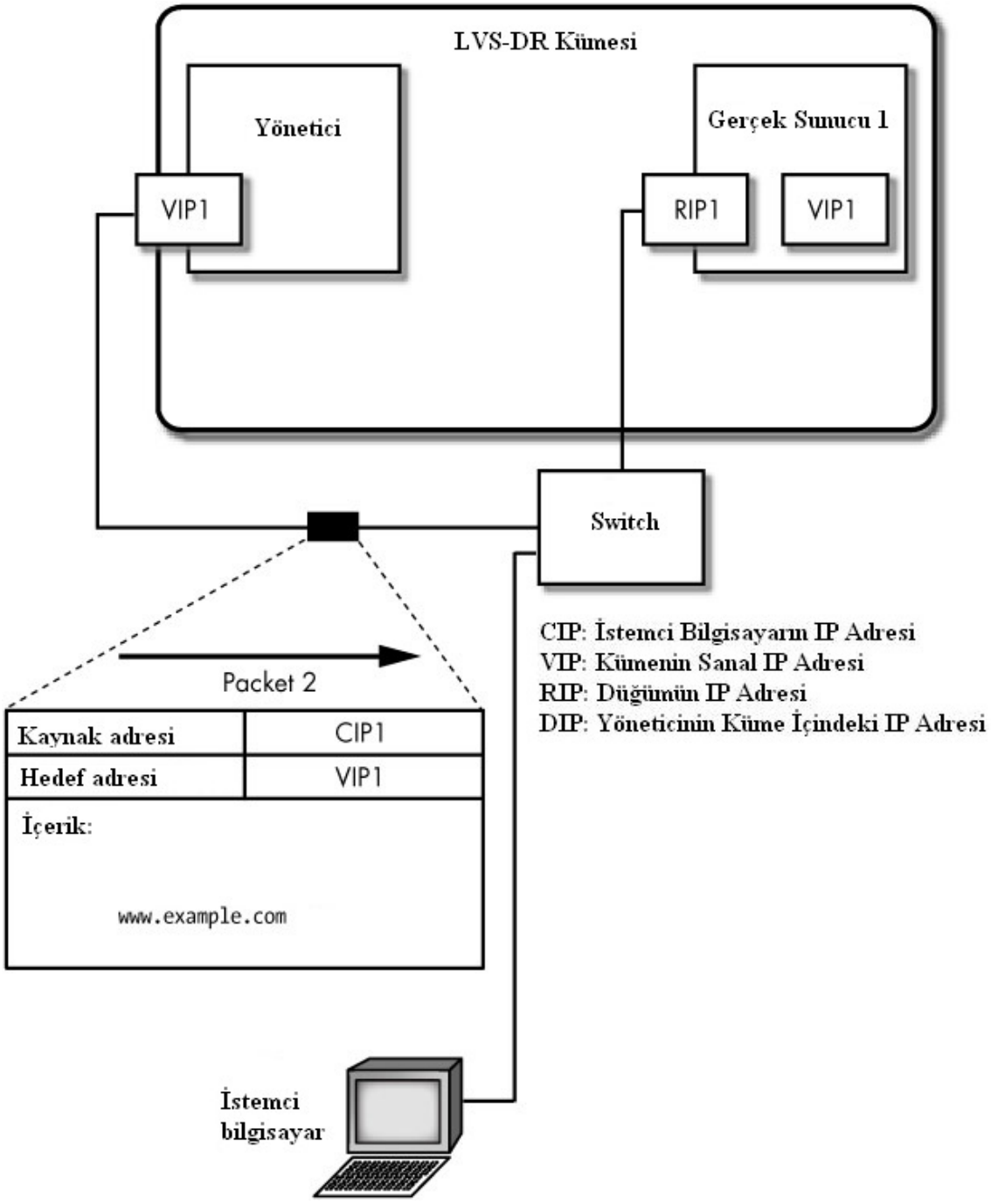
LVS-DR kümesinde Yönetici, küme yapısındaki düğümler ve istemci bilgisayarlar aynı switch veya hub üzerinden bağlıdır. Tüm bileşenler aynı fiziksel ağda bulunmaktadır.

LVS-DR kümesinde önce istemci bilgisayar VIP adresini kullanarak kümeye bir istek gönderir. Örneğimizde paket içeriği bir web sayfası görüntülemek için HTTP isteğidir. İstemci bilgisayardan gelen pakette ARP ile öğrenilen Yönetici MAC adresi bulunmaktadır.

İstemci bilgisayardan gönderilen ARP broadcast paketinde istemci VIP1 IP adresinin kime ait olduğunu sormaktadır. Bu soruya Yönetici kendi MAC adresiyle yanıt vermektedir. İstemci MAC adresini öğrendiğinde bu MAC adresini paket 1 içine hedef adres olarak girmektedir. Böylece paket Yöneticiye ulaşmaktadır.

Eğer küme Internete bağlıysa ve istemci bilgisayar kümeye Internet üzerinden ulaşıyorsa bu durumda ARP broadcast gönderilmeyecektir. İstemci tarafından gönderilen paket yönlendiriciye ulaştığında yönlendirici ARP broadcast göndererek MAC adresini bulacaktır.

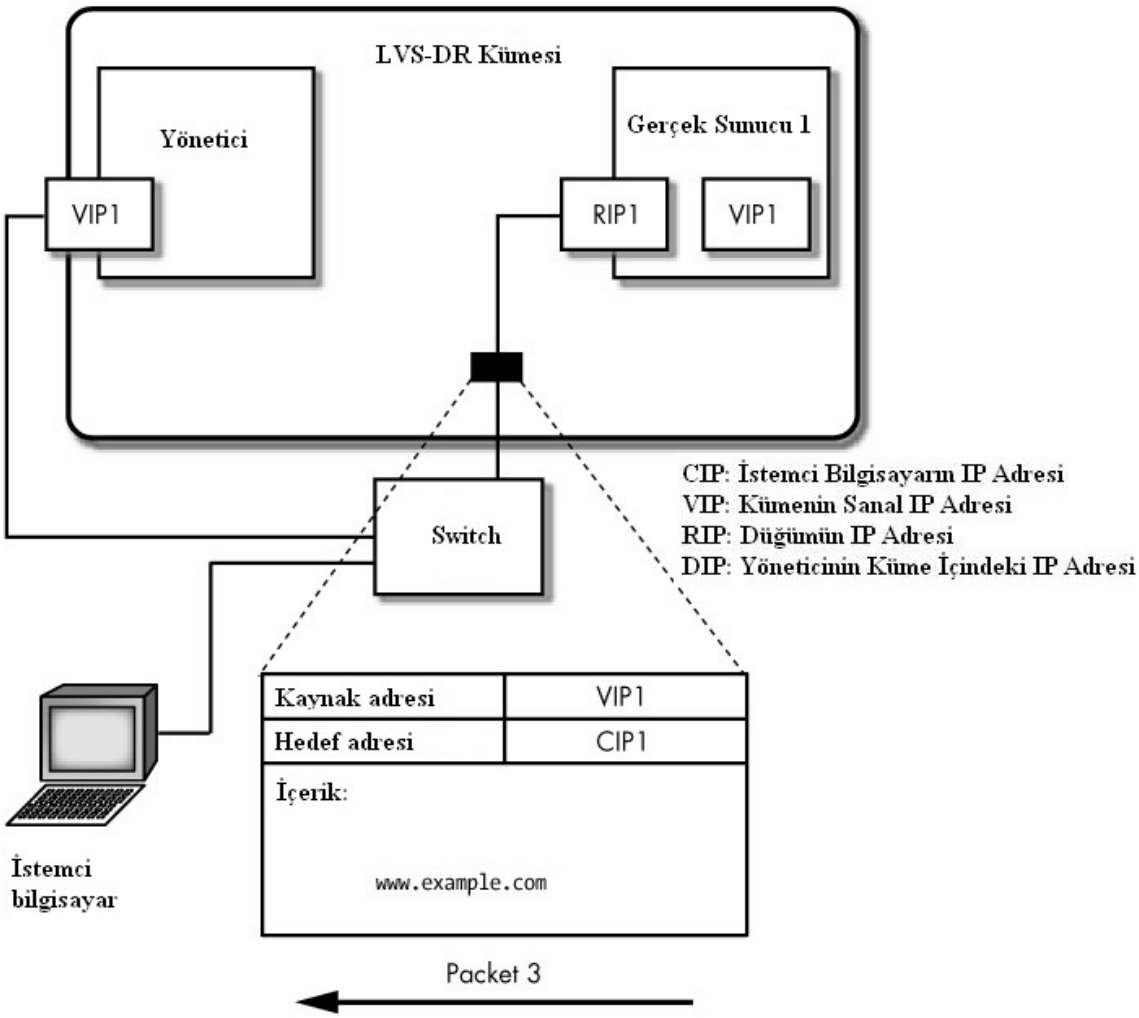
Paket 1 Yöneticiye vardığında Yönetici kaynak ve hedef adreslerini değiştirmeden paketi düğüme iletir. Gönderilen Paket 2 de sadece Yöneticinin MAC adresinin yerine düğümün MAC adresi yazılmaktadır.



Şekil 4.21. Yöneticinin istemciden gelen isteği küme düğümüne iletmesi

Paket kümedeki hedef düğüme ulaştığında paket düğümdeki loopback cihazına yönlendirilir. Bunun nedeni çekirdeğin içinde bulunan yönlendirme tablosudur. Çünkü düğümdeki VIP1 adresi loopback adresine aittir. Paket daha sonra VIP1 adresini dinleyen ve yerel olarak düğümde çalışan daemon tarafından alınır. Gelen istek bir web sayfası görüntüleme isteği olduğu için buradaki daemon Apache HTTPd web

sunucusudur. HTTPd daemon bir yanıt paketi hazırlar ve bu paketi RIP1 arabiriminden VIP1 kaynak adresi ile gönderir.



Şekil 4.22. Küme düğümünün yanıt paketi göndermesi

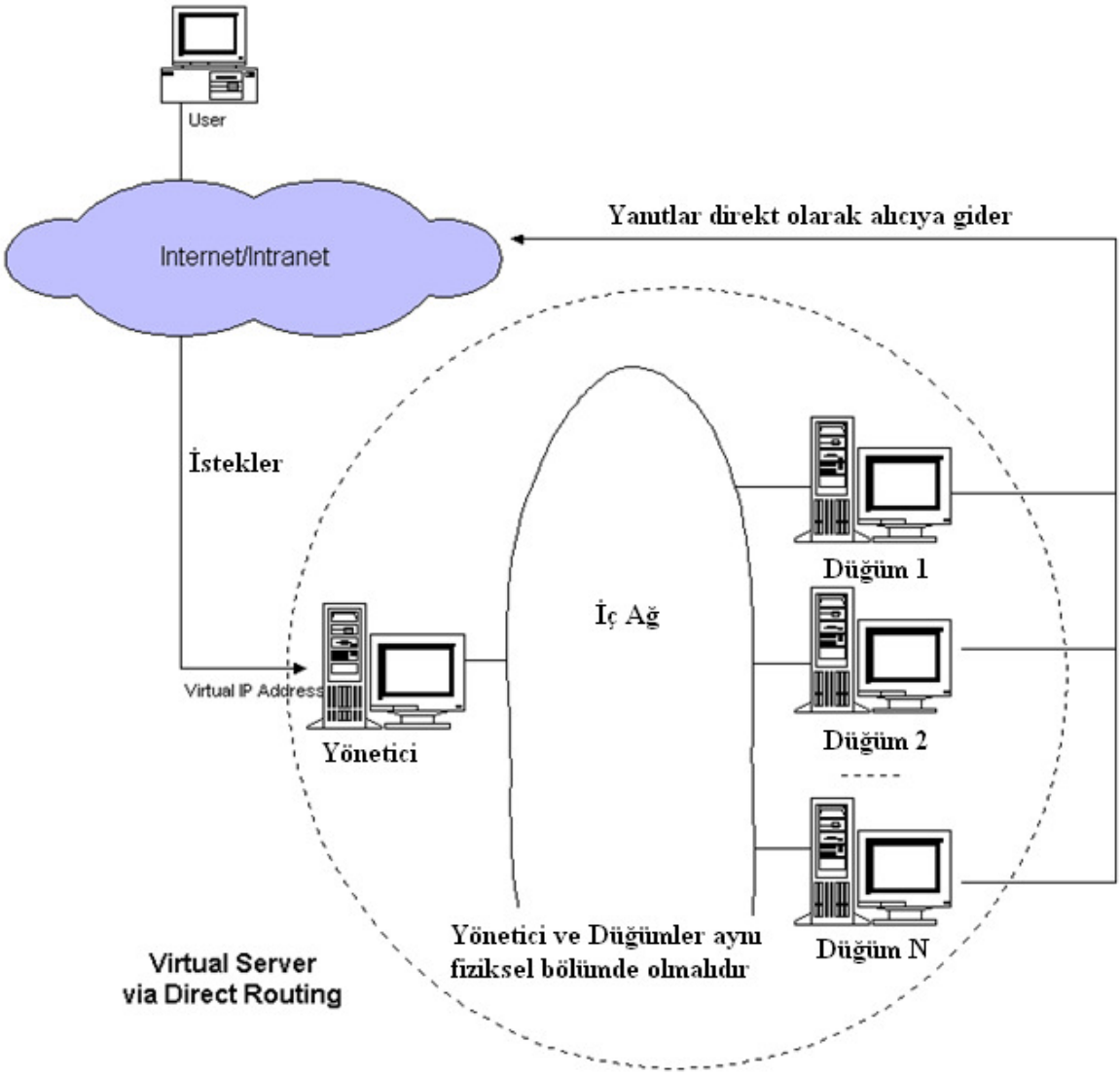
Yanıt paketi geriye Yönetici üzerinden gitmeyecektir. Çünkü LVS-DR kümesindeki düğümler Yöneticiyi varsayılan ağ geçidi olarak kullanmazlar.

Bu iletişimde dikkat edilmesi gereken hususlar şunlardır;

- Yönetici kümeye doğru gelen tüm paketleri almaktadır.
- Yönetici sadece küme içine doğru olan iletişimi almaktadır.
- Kümedeki düğümler, Yönetici ve istemci bilgisayarlar aynı fiziksel ağda bulunurlar.
- Düğümler yönlendiriciyi varsayılan ağ geçidi olarak kullanırlar.

4.8.1. Virtual Server via Direct Routing yönteminin detaylı incelenmesi ve konfigürasyonu

Direct Routing tekniğinin çalışmasına ait mimari aşağıdaki şekildeki gibidir.

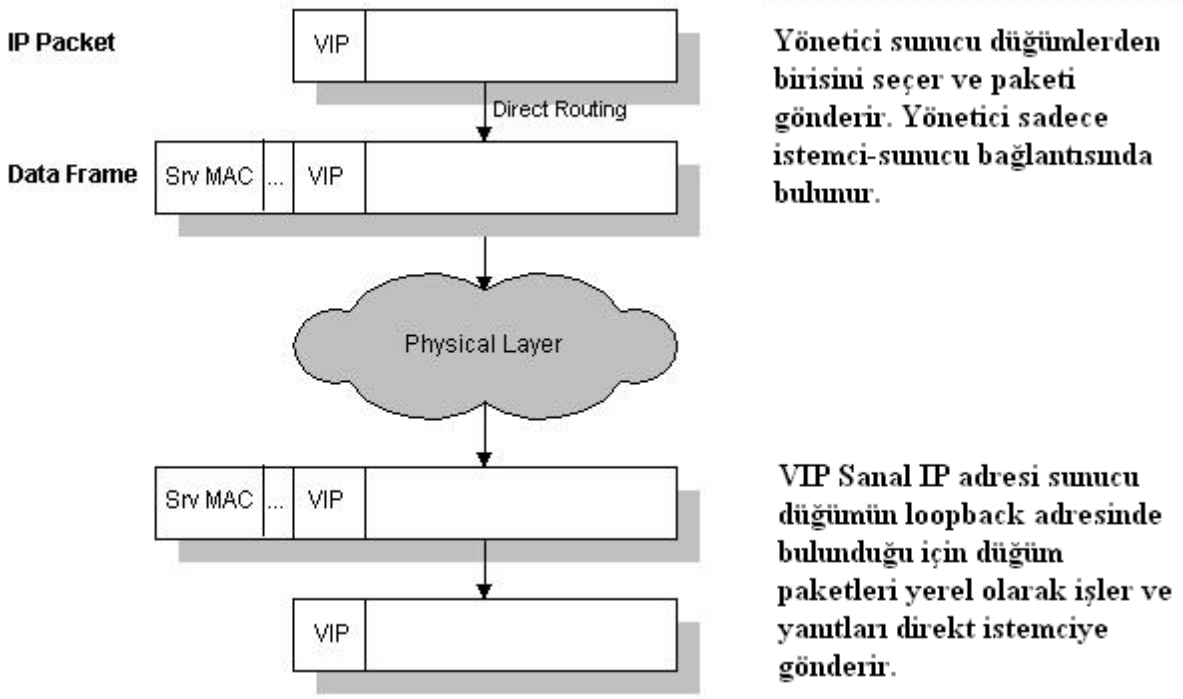


Şekil 4.23. Virtual Server via Direct Routing Yöntemiyle Küme Oluşturulması

Kullanıcı küme tarafından sunulan bir servise erişmek istediğinde sanal IP adresi için olan istek paketi yöneticiye geldiğinde yönetici paketin hedef adresine ve port numarasına bakar. Eğer paket sanal sunucu tablosundaki sanal servislerden birisine uyarsa zamanlama yöntemine göre sunucu düğümlerinden birisi seçilir. Bağlantı bağlantı izleme tablosuna eklenir. Daha sonra paketin hedef adres ve port numarası

seçilen sunucuya göre tekrar yazılır ve paket sunucuya iletilir. Eğer içeriye doğru gelen paket bu bağlantıya ait ise paket tekrar yazılır ve sunucuya iletilir. Yanıt paketleri geldiğinde yönetici kaynak adresi ve port numarasına sanal servisinkileri yazar. Bağlantı sona erdiğinde veya timeout durumuna düştüğünde bağlantı izleme tablosundan silinir [5].

Direct routing yönteminde akış aşağıdaki şekildeki gibi gerçekleşir.



Şekil 4.24. Direct Routing Yönteminde Akış

Yönetici paketin MAC adresini seçili olan düğümün MA adresiyle değiştirir ve paketi ağa tekrar gönderir.

4.8.2. Virtual Server işlemi için çekirdeğin yapılandırılması

Öncelikle Linux çekirdek sürümlerinden kararlı bir tanesi indirilir ve virtual server yaması çekirdeğe uygulanır. 2.6 ve üstü çekirdekler için yapılması gereken bir işlem yoktur.

2.0.36 sürümü çekirdekler için VS yamasının uygulanması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network firewalls
    ...
    [*] IP: forwarding/gatewaying
    ...
    [*] IP: firewalling
    ...
    [*] IP: masquerading
    ...
    [*] IP: ippfvs(LinuxDirector) masquerading (EXPERIMENTAL)
    Virtual server request dispatching technique---
    ( ) VS-NAT
    ( ) VS-Tunneling
    (X) VS-DRouting
```

Seçim yapıldıktan sonra zamanlama yöntemi seçilir.

```
   Virtual server scheduling algorithm
    (X) WeightedRoundRobin
    ( ) LeastConnection
    ( ) WeightedLeastConnection
    [ ] IP: enabling ippfvs with the local node feature
```

Son olarak sisteme ippfvsadm kurulumu yapılır.

2.2.x sürümü çekirdekler için IPVS yamasının uygulanması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network firewalls
    ...
    [*] IP: forwarding/gatewaying
    ...
    [*] IP: firewalling
    ...
    [*] IP: masquerading
    ...
    [*] IP: masquerading virtual server support (EXPERIMENTAL)
    (12) IP masquerading table size (the Nth power of 2)
    <M> IPVS: round-robin scheduling
```

```
    <M> IPVS: weighted round-robin scheduling
    <M> IPVS: least-connection scheduling
    <M> IPVS: weighted least-connection scheduling
    <M> IPVS: locality-based least-connection scheduling
    <M> IPVS: locality-based least-connection with replication
scheduling
```

Son olarak sisteme ipvsadm kurulumu yapılır.

2.4.x sürümü çekirdekler için IPVS yamasının uygulanması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network packet filtering (replaces ipchains)
    [ ]   Network packet filtering debugging
    ...
      IP: Netfilter Configuration  --->
      IP: Virtual Server Configuration  --->
       <M> virtual server support (EXPERIMENTAL)
       [*]   IP virtual server debugging
       (12)   IPVS connection table size (the Nth power of 2)
       --- IPVS scheduler
       <M>   round-robin scheduling
       <M>   weighted round-robin scheduling
       <M>   least-connection scheduling scheduling
       <M>   weighted least-connection scheduling
       <M>   locality-based least-connection scheduling
       <M>   locality-based least-connection with replication
scheduling
       <M>   destination hashing scheduling
       <M>   source hashing scheduling
       --- IPVS application helper
       <M>   FTP protocol helper
```

4.8.3. Virtual Server via Direct Routing yönteminin test edilmesi

Virtual server via direct routing yöntemin test edilmesi için kullanılacak bileşenler aşağıdaki özelliklere sahiptir.

Yönetici IP adresi: 192.168.0.111

Sunucu düğüm IP adresi: 192.168.0.112

Sanal IP adresi: 192.168.0.110

2.0.x sürümü çekirdekler için:

Yönetici, çekirdek 2.0.36

```
ifconfig eth0 192.168.0.111 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0
ifconfig eth0:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
route add –host 192.168.0.110 dev eth0:0
ippfvsadm –A –t 192.168.0.110:23 –R 192.168.0.112
```

Sunucu düğüm, çekirdek 2.0.36 (IP forwarding devrede)

```
ifconfig eth0 192.168.0.112 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0
ifconfig lo:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
route add –host 192.168.0.110 dev lo:0
```

2.2.x sürümü çekirdekler için:

Yönetici, çekirdek 2.2.14

```
ifconfig eth0 192.168.0.111 netmask 255.255.255.0 broadcast
192.168.0.255 up
ifconfig eth0:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
echo 1 > /proc/sys/net/ipv4/ip_forward
ipvsadm –A –t 192.168.0.110:23 –s wlc
ipvsadm –a –t 192.168.0.110:23 –r 192.168.0.112 –g
```

Sunucu düğüm, çekirdek 2.0.36 (IP forwarding devrede)

```
ifconfig eth0 192.168.0.112 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0
ifconfig lo:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
route add –host 192.168.0.110 dev lo:0
```

2.2.14 ve sonrası sürümler çekirdekler için (Gizli Aygıt):

Yönetici, çekirdek 2.2.14

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipvsadm –A –t 192.168.0.110:23 –s wlc
ipvsadm –a –t 192.168.0.110:23 –r 192.168.0.112 –g
```

Sunucu düğüm, çekirdek 2.2.14

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
echo 1 > /proc/sys/net/ipv4/conf/lo/hidden
ifconfig lo:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
```

2.2.x sürümü çekirdekler için (Redirect yaklaşımı):

Yöneticinin konfigürasyonu diğer sürümlerle aynıdır. Sunucu düğümlerinin konfigürasyonu aşağıdaki gibidir.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipchains –A input –j REDIRECT 23 –d 192.168.0.110 23 –p tcp
...
```

ipchains redirect komutlarıyla 192.168.0.110 port 23 için gelen paketler yerel bir sokete tekrar yönlendirilir.

Farklı ağ yollarına sahip düğümler için:

Virtual server via direct routing yönteminde sunucu düğümler istemcilere giderken farklı ağ yollarını takip edebilirler.

Yönetici, çekirdek 2.2.14

```
ifconfig eth0 <an IP address> ...
...
ifconfig eth0:0 <VIP> netmask 255.255.255.255 broadcast <VIP> up
ifconfig eth1 192.168.0.1 netmask 255.255.255.0 broadcast
192.168.0.255 up
ipvsadm –A –t <VIP>:23
ipvsadm –A –t <VIP>:23 –r 192.168.0.2 –g
...
```

Sunucu düğüm, çekirdek 2.0.36

```
ifconfig eth0 <a seperate IP address> ...
```

```
# Follow the different network route
...
ifconfig eth1 192.168.0.2 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth1
ifconfig lo:0 <VIP> netmask 255.255.255.255 broadcast <VIP> up
route add –host <VIP> dev lo:0
```

4.9. LVS-TUN Kümesi

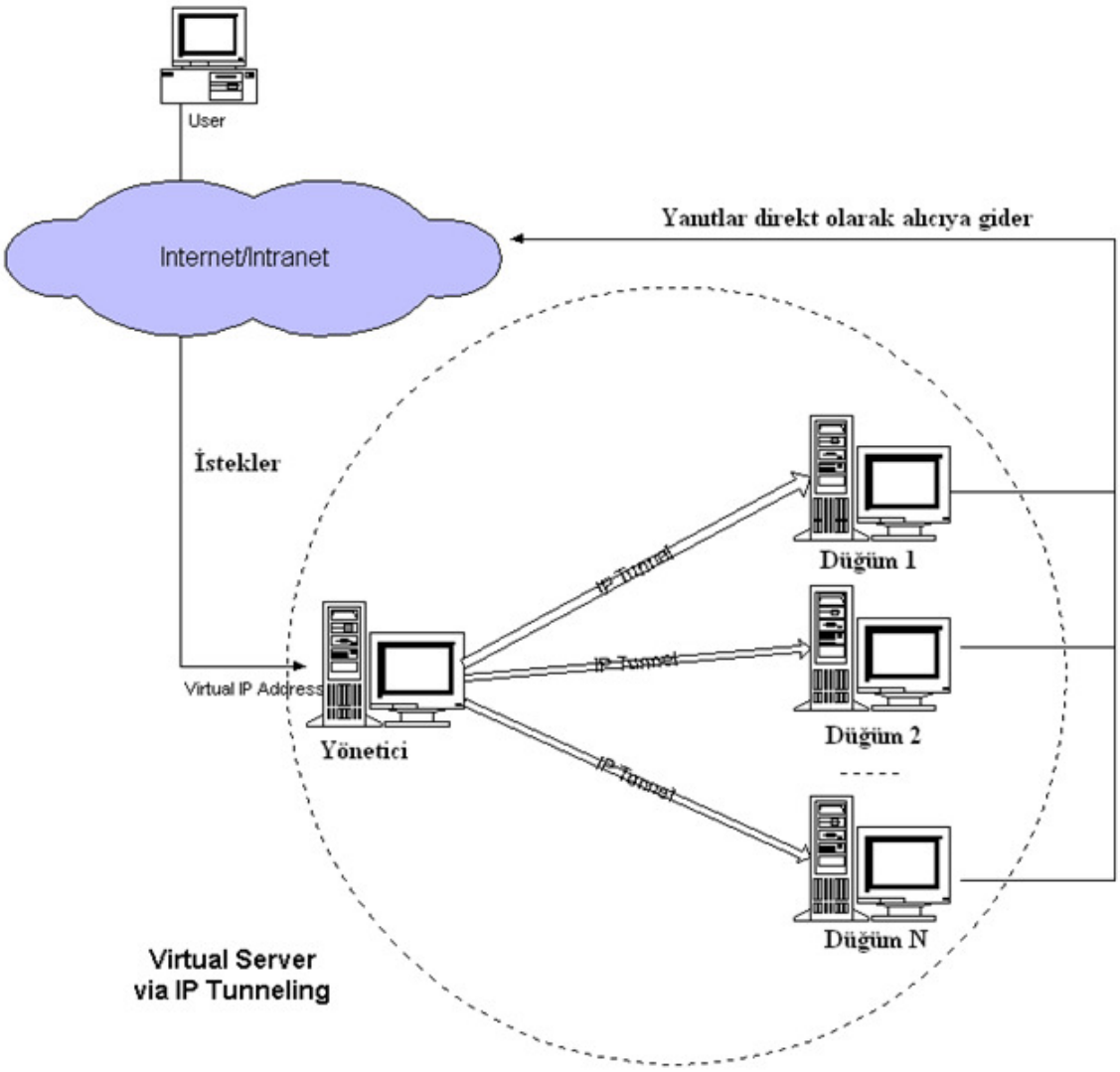
LVS-TUN (IP Tünelleme) bir alt ağdan veya VLAN (sanal ağ) den diğer bir alt ağa veya sanal ağa paketleri iletirken kullanılır. Bu durum paketler başka bir ağdan veya Internetten geçerken de geçerlidir. IP tünelleme yeteneği Linux çekirdeğinden gelir. LVS-TUN iletme yöntemi Yönetici ile aynı ağ bölümünde olmayan küme düğümlerinin de küme ağına yerleştirilmesine izin verir.

LVS-TUN Kümesi konfigürasyonu LVS-DR yönteminin paket iletme yöntemini iyileştirmektedir. İstemci bilgisayarlarından küme servisleri için gelen istekler paketlenir ve böylece paketler Yönetici ile aynı fiziksel ağ bölümünde olmayan küme düğümlerine iletilebilir.

4.9.1. Virtual Server via IP Tunneling yönteminin detaylı incelenmesi ve konfigürasyonu

IP tunneling (IP encapsulation) yöntemi IP datagramlarını IP datagram ları içinde encapsulate etmek için kullanılan bir yöntemdir. Bir IP adresini hedefleyen datagramların sarmalanmasını ve başka bir IP adresine tekrar yönlendirilmesini sağlar. IP encapsulation yöntemi Extranet, Mobile-IP, IP-Multicast, tunneled host ve ağ teknolojilerinde yaygın olarak kullanılmaktadır. IP tunneling tekniği sanal sunucu kümelerinde yüksek performans sağlamaktadır [5].

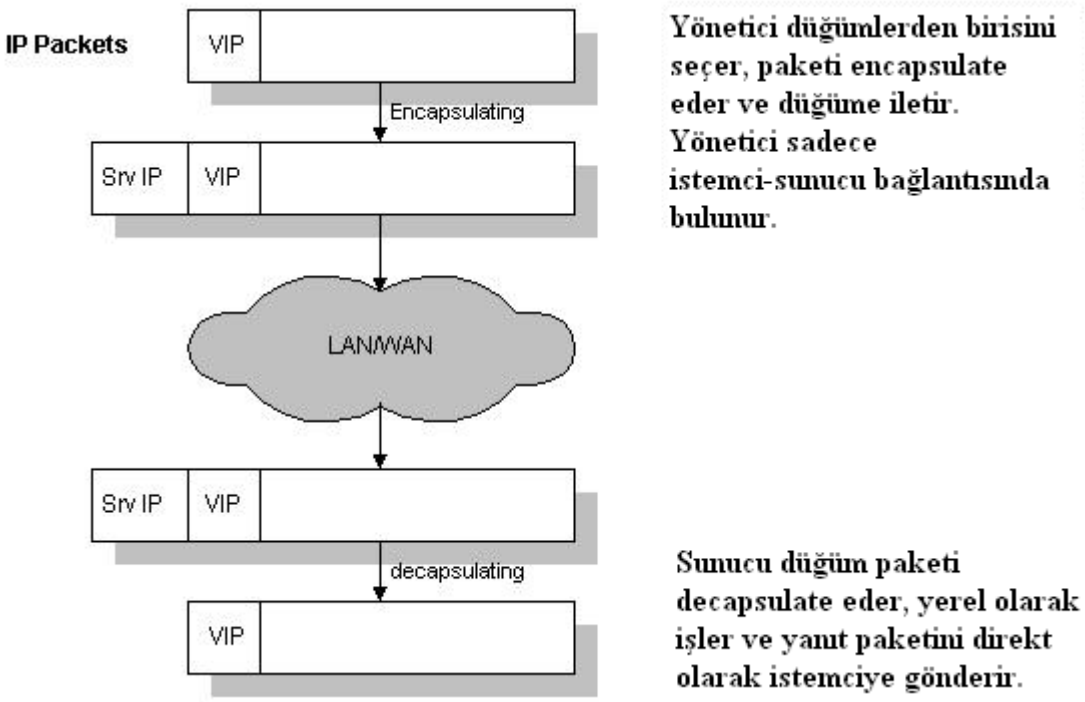
IP Tunneling tekniğinin çalışmasına ait mimari aşağıdaki şekildeki gibidir.



Şekil 4.25. Virtual Server via IP Tunneling Yöntemiyle Küme Oluşturulması

Kullanıcı küme tarafından sunulan bir servise erişmek istediğinde sanal IP adresi için olan istek paketi yöneticiye geldiğinde yönetici paketin hedef adresine ve port numarasına bakar. Eğer paket sanal sunucu tablosundaki sanal servislerden birisine uyarsa zamanlama yöntemine göre sunucu düğümlerinden birisi seçilir. Bağlantı bağlantı izleme tablosuna eklenir. Daha sonra yönetici paketi IP datagram içine encapsulate eder ve seçilmiş düğüme iletir. Eğer içeriye doğru gelen paket bu bağlantıya ait ise paket tekrar encapsulate edilir ve sunucuya iletilir. Sunucu encapsulate edilmiş paketi aldığında paketi decapsulate eder ve isteği işler. Bağlantı sona erdiğinde veya timeout durumuna düştüğünde bağlantı izleme tablosundan silinir [5].

IP Tunneling yönteminde akış aşağıdaki şekildeki gibi gerçekleşir.



Şekil 4.26. IP Tunneling Yönteminde Akış

Burada dikkat edilmesi gereken nokta düğümlerin birbirlerinden uzak noktalarda herhangi bir ağda bulunabileceğidir. Düğümler farklı IP adres gruplarına da sahip olabilirler. Tek zorunluluk IP encapsulation protokolünü desteklemedir. Tünellemeden sorumlu olan cihazlar sistemlerin aldıkları encapsulate edilmiş paketleri düzgün olarak decapsulate etmelerine izin vermelidirler. Ayrıca sistem sanal IP adresine gelmiş olan paketleri bir yerel sokete yönlendirmelidir.

Son olarak encapsulate edilmiş paketler geldiğinde gerçek sunucu düğümü paketi decapsulate eder ve paketin küme sanal IP adresini hedeflediğini gördüğünde isteği işler ve sonucu direkt olarak kullanıcıya döndürür.

4.9.2. Virtual Server işlemi için çekirdeğin yapılandırılması

Öncelikle Linux çekirdek sürümlerinden kararlı bir tanesi indirilir ve virtual server yaması çekirdeğe uygulanır. 2.6 ve üstü çekirdekler için yapılması gereken bir işlem yoktur.

2.0.36 sürümü çekirdekler için VS yamasının uygulanması:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network firewalls
    ...
    [*] IP: forwarding/gatewaying
    ...
    [*] IP: firewalling
    ...
    [*] IP: masquerading
    ...
    [*] IP: ippfvs(LinuxDirector) masquerading (EXPERIMENTAL)
    Virtual server request dispatching technique---
    ( ) VS-NAT
    (X) VS-Tunneling
    ( ) VS-DRouting
```

Seçim yapıldıktan sonra zamanlama yöntemi seçilir.

```
    Virtual server scheduling algorithm
    (X) WeightedRoundRobin
    ( ) LeastConnection
    ( ) WeightedLeastConnection
    [ ] IP: enabling ippfvs with the local node feature
```

Son olarak sisteme ippfvsadm kurulumu yapılır.

2.2.x sürümü çekirdekler için VS yamasının kurulumu:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network firewalls
```

```
    ...
    [*] IP: forwarding/gatewaying
    ...
    [*] IP: firewalling
    ...
    [*] IP: masquerading
    ...
    [*] IP: masquerading virtual server support (EXPERIMENTAL)
    (12) IP masquerading table size (the Nth power of 2)
    <M> IPVS: round-robin scheduling
    <M> IPVS: weighted round-robin scheduling
    <M> IPVS: least-connection scheduling
    <M> IPVS: weighted least-connection scheduling
    <M> IPVS: locality-based least-connection scheduling
    <M> IPVS: locality-based least-connection with replication
scheduling
```

Son olarak sisteme ipvsadm kurulumu yapılır.

2.4.x sürümü çekirdekler için IPVS yamasının kurulumu:

Çekirdek derleme seçenekleri:

```
Code maturity level options --->
    [*] Prompt for development and/or incomplete code/drivers

Networking options --->
    [*] Network packet filtering (replaces ipchains)
    [ ]   Network packet filtering debugging
    ...
      IP: Netfilter Configuration  --->
      IP: Virtual Server Configuration  --->
       <M> virtual server support (EXPERIMENTAL)
       [*]   IP virtual server debugging
       (12)   IPVS connection table size (the Nth power of 2)
       --- IPVS scheduler
       <M>   round-robin scheduling
       <M>   weighted round-robin scheduling
       <M>   least-connection scheduling scheduling
       <M>   weighted least-connection scheduling
       <M>   locality-based least-connection scheduling
       <M>   locality-based least-connection with replication
scheduling
       <M>   destination hashing scheduling
       <M>   source hashing scheduling
       --- IPVS application helper
       <M>   FTP protocol helper
```

4.9.3. IP Tunneling örneği

Aşağıdaki tablo IP tunneling kullanılan bir küme yapısında uygulanan kuralları göstermektedir.

Protocol	Virtual IP Address	Port	Real IP Address	Weight
TCP	193.255.141.39	80	193.255.141.40	1
			193.255.141.41	2

Tablo 4.3. Linux Virtual Server Yönetici-Düğüm İlişkileri

193.255.141.39 IP adresi Port 80 hedefine giden tüm trafik için 193.255.141.40 Port 80 ve 193.255.141.41 Port 80 sunucu düğümleri üzerinden yük dengeleme yapılmaktadır.

Konfigürasyonun Yapılması

2.0.x sürümü çekirdekler için:

```
ippfvsadm –A –t 193.255.141.39:80 –R 193.255.141.40 –w 1
ippfvsadm –A –t 193.255.141.39:80 –R 193.255.141.41 –w 2
```

2.2.x , 2.4.x ve 2.6.x sürümleri çekirdekler için:

```
ipvsadm –A –t 193.255.141.39:80 –s wlc
ipvsadm –a –t 193.255.141.39:80 –r 193.255.141.40 –i –w 1
ipvsadm –a –t 193.255.141.39:80 –r 193.255.141.41 –i –w 2
```

4.9.3.1. Virtual Server via IP Tunneling yönteminin test edilmesi

Virtual server via IP Tunneling yöntemin test edilmesi için kullanılacak bileşenler aşağıdaki özelliklere sahiptir.

Yönetici IP adresi: 192.168.0.111

Sunucu düğüm IP adresi: 192.168.0.112

Sanal IP adresi: 192.168.0.110

2.0.x sürümü çekirdekler için:

Yönetici, çekirdek 2.0.36

```
ifconfig eth0 192.168.0.111 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0
ifconfig eth0:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
route add –host 192.168.0.110 dev eth0:0
ippfvsadm –A –t 192.168.0.110:23 –R 192.168.0.112
```

Sunucu düğüm, çekirdek 2.0.36 (IP forwarding devrede)

```
ifconfig eth0 192.168.0.112 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0
ifconfig tunl0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
route add –host 192.168.0.110 dev tunl0
```

2.2.x sürümü çekirdekler için:

Yönetici, çekirdek 2.2.14

```
ifconfig eth0 192.168.0.111 netmask 255.255.255.0 broadcast
192.168.0.255 up
ifconfig eth0:0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
echo 1 > /proc/sys/net/ipv4/ip_forward
ipvsadm –A –t 192.168.0.110:23 –s wlc
ipvsadm –a –t 192.168.0.110:23 –r 192.168.0.112 –i
```

Sunucu düğüm, çekirdek 2.0.36 (IP forwarding devrede)

```
ifconfig eth0 192.168.0.112 netmask 255.255.255.0 broadcast
192.168.0.255 up
route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0
ifconfig tunl0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
route add –host 192.168.0.110 dev tunl0
```

2.2.14 ve sonrası sürümler çekirdek için (Gizli aygıt):

Yönetici, çekirdek 2.2.14

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipvsadm –A –t 192.168.0.110:23 –s wlc
ipvsadm –a –t 192.168.0.110:23 –r 192.168.0.112 –i
```

Sunucu düğüm, çekirdek 2.2.14

```
echo 1 > /proc/sys/net/ipv4/ip_forward
# insert it if it is compiled as module
modprobe ipip
ifconfig tunl0 0.0.0.0 up
echo 1 > /proc/sys/net/ipv4/conf/all/hidden
echo 1 > /proc/sys/net/ipv4/conf/tunl0/hidden
ifconfig tunl0 192.168.0.110 netmask 255.255.255.255 broadcast
192.168.0.110 up
```

2.2.x sürümler çekirdek için (Redirect yaklaşımı):

Yöneticinin konfigürasyonu diğer sürümlerdeki çekirdeklerle aynıdır. Düğümlerin konfigürasyonu aşağıdaki gibidir.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
# insert it if it is compiled as module
modprobe ipip
ifconfig tunl0 0.0.0.0 up
ipchains –A input –j REDIRECT 23 –d 192.168.0.110 23 –p tcp
...
```

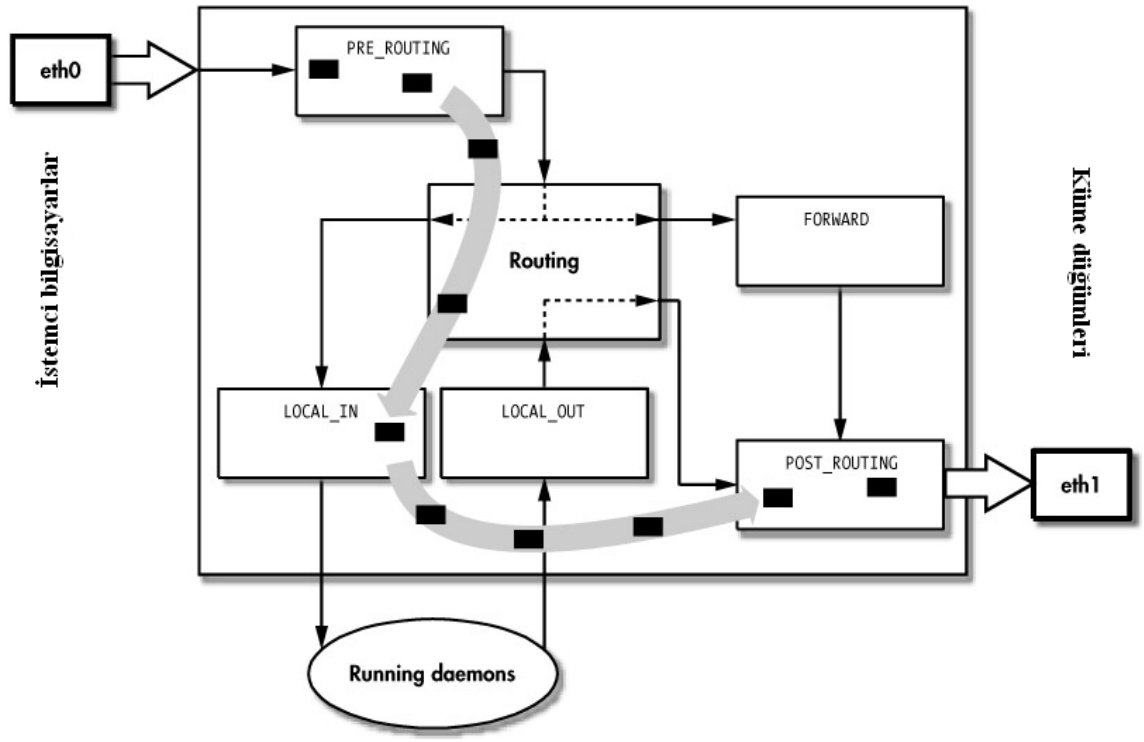
4.10. Yük Dengeleyicinin Çalışma Prensipleri

4.10.1. LVS ve Netfilter

İçeriye doğru gelen LVS paketleri beş Netfilter kancasından üç tanesine uğrar. Bunlar *PRE_ROUTING*, *LOCAL_IN*, ve *POST_ROUTING* dir. Aşağıdaki şekildeki ilk gri ok çekirdek içinde *PRE_ROUTING* kancasından *LOCAL_IN* kancasına geçen bir

paketi göstermektedir. Yönetici tarafından alınmış olan ve bir küme servisini hedefleyen her paket LVS iletim yöntemi ne olursa olsun Yöneticideki çekirdeğin içinde *PRE_ROUTING* kancasından *LOCAL_IN* kancasına geçmek zorundadır. Küme servisleri için bunu yapmak zor değildir. Zaten küme servislerine ulaşmak isteyen her istemci hedef IP adresi olarak Yöneticinin sanal IP adresini kullanmak zorundadır. Sanal IP adresi Yöneticinin yerel IP adreslerinden birisi olduğu için Yöneticinin çekirdeğindeki yönlendirme tablosu daima paketleri yerel olarak teslim etmeyi deneyecektir. Dolayısıyla Yönetici tarafından alınan tüm paketler *LOCAL_IN* kancasına uğrarlar.

Aşağıdaki şekildeki ikinci gri ok çekirdeğin paketin sanal bir servis için bir istek olduğunu anlamasından sonra içeriye doğru gelen paketlerin izlediği yolu göstermektedir. Herhangi bir paket *LOCAL_IN* kancasına geldiğinde çekirdek içinde çalışan LVS yazılımı paketin küme servisi isteği olduğunu bilmektedir. Küme yapısı oluşturulduktan sonra *ipvsadm* kullanılarak sanal bir servis için sanal IP adresleri çekirdeğe eklenebilir. Böylece LVS yazılımı *LOCAL_IN* kancasında sanal IP adreslerine gönderilen paketleri tanıyabilir. Eğer sanal IP adresi LVS sanal servis tablosuna eklenmemişse sanal IP adresine hedeflenmiş paketler Yöneticide yerel olarak çalışan daemonlara teslim edilecektir. Fakat LVS sanal IP adresini bildiği için *LOCAL_IN* kancasına gelen her paket LVS tarafından bir küme servisi isteği olup olmadığı kontrol edilir. Böylece LVS Yöneticide yerel olarak çalışan daemonlara gelen paketin kaderini değiştirebilir. LVS çekirdeğin paketin hedefini değiştirimesini ve paketin küme düğümüne gönderilmesini sağlar. Böylece paket aşağıdaki şekilde ikinci gri okta görüldüğü gibi *POST_ROUTING* kancasına gönderilir.



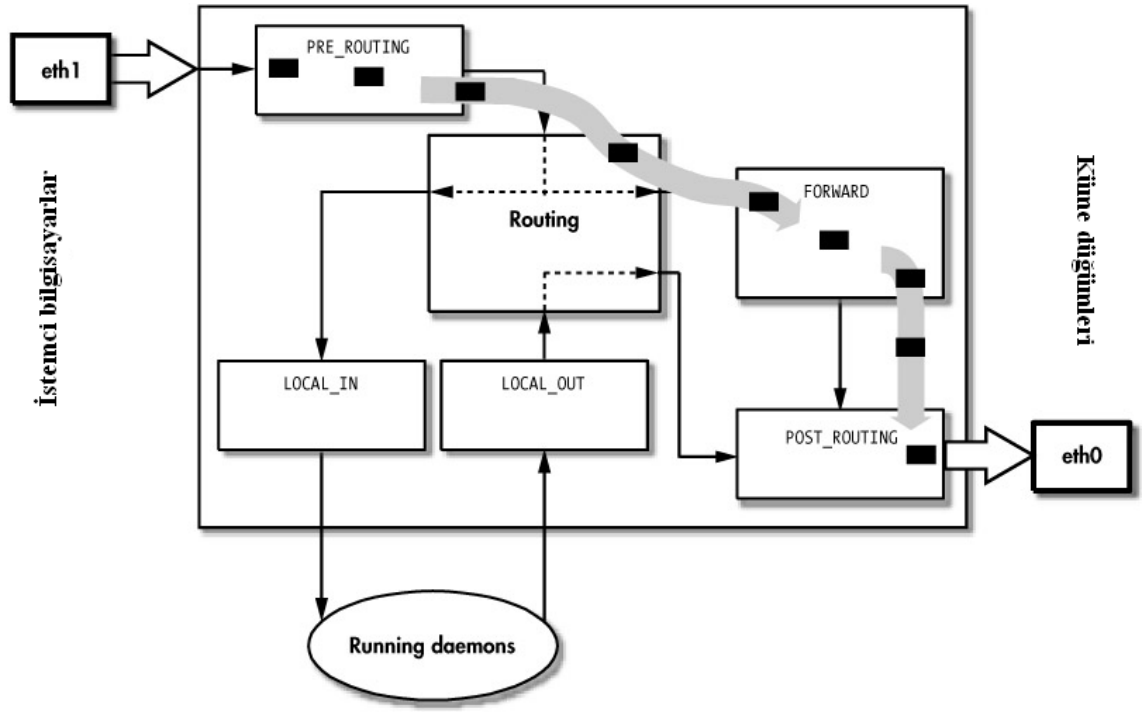
Şekil 4.27. Yöneticinin içinde gelen paketlerin hareketi

LVS Yöneticisi paketlerin kaderini Netfilter kancaları ve *ipvsadm* ile oluşturulabilecek IPVS tablosuyla değiştirebilir. Yönetici küme servisleri için gelen paketleri küme düğümleri arasında dağıtabilir. Bunu yapabilmesi için Yönetici her bir istemciye atanan küme düğümlerini takip eder ve istemci sürekli olarak aynı düğümle konuşur. Yönetici bellekte "connection tracking table" olarak isimlendirilen bir tablo tutarak bunu gerçekleştirir.

LVS-NAT kümesinde küme düğümleri istemci bilgisayarlara yanıt gönderirken paketler Yöneticiden geçmelidir. Bu yanıt paketlerinin LVS-NAT kümesinde Yöneticiden geçmesinin nedeni Yöneticinin NAT (Network Address Translation) yaparak küme düğümlerinden gelen paketlerdeki kaynak IP adresinlerini Yöneticinin sanal IP adresine çevirmesi zorunluluğudur.

Yöneticide küme düğümlerinden gelen yanıt paketlerindeki kaynak IP adreslerinin NAT işlemine tabi tutulması *FORWARD* kancası sayesinde olur. Paketler Yöneticiden istemci bilgisayara geri dönerken Netfilter kancalarından geçer ve LVS bağlantı izleme tablosuna bakarak paket başlığında hangi düğümlerin IP adreslerini sanal IP adresleri ile değiştirileceğini bulur. Aşağıdaki şekilde LVS-NAT yöntemi

kullanılan kümede yöneticiden geçerek geri dönen yanıt paketlerinin izlediği yol görülmektedir.



Şekil 4.28. Yöneticinin içinde dışarıya doğru giden LVS-NAT paketlerinin hareketi

Dışarıya doğru giden paketler içeriye doğru giden paketlerle aynı yöntemlerle işlenir. Ancak bu durumda paketler için yönlendirme tablosunda uygun bir yönlendirme kuralı bulunmadığı için çekirdek paketi *FORWARD* kancası ile ağa iletir. Paket *FORWARD* kancasından sonra *POST_ROUTING* kancası üzerinden ağa aktarılır.

4.10.2. Connection Tracking Table (Bağlantı İzleme Tablosu)

Yöneticide bulunan Connection Tracking Table (IPVS Connection Tracking Table) istemcilerden gelen her bir bağlantı için 128 byte'lık bir bilgi saklar. Bu bilginin amacı aynı istemciden gelen paketlerin aynı sunucuya ulaşmasıdır.

Tablo satır ve sütunlardan oluşur. Tablodaki her satır " hash bucket " olarak, her sütun ise " connection tracking record (bağlantı takip kaydı) " olarak isimlendirilir.

Tablodaki her kayıt zaman sayacı bilgisi, paket tarafından kullanılan protokol, istemcinin IP adresi, istemcinin port numarası, sanal IP adresi, sanal port numarası ve bazı ek kontrol bilgisi içerir. Her satır ise bu kayıtlardan sınırsız sayıda içerebilir.

Yönetici her paket aldığında tabloda eşleşen bir kayıt aradığı için tabloya mümkün olduğunca hızlı bakabilmelidir.LVS tabloya bakarken bir hashing tekniği kullanarak ilk olarak hangi satırdan başlayacağına karar verir. Tablo ne kadar küçükse arama süreside o kadar kısadır. LVS uzmanları her satır için ortalama 16-20 arası kayıt önermektedir.

Connection Tracking Table içeriğini görüntülemek için aşağıdaki komut kullanılır.

```
#ipvsadm –lcn
```

ipvsadm komutu kullanıldığında connection tracking table boyutu ekrana yazdırılır. Değer byte büyüklüğündedir.

```
#ipvsadm
IP Virtual Server version 1.2.4 (size=4096)
```

2.2 serisi çekirdeklerde bu tablonun içeriği aşağıdaki komutla görülebilir.

```
#netstat –Mn
```

Bir istemci bilgisayar ile bir küme düğümü arasındaki ağ iletişimi daha fazla aktif değilse ve timeout süresi dolduysa kayıt tablodan silinir. Örneğin telnet servisi TCP protokolünü kullanır. Telnet servisinde Yönetici bağlantı izleme tablosundaki kayıdı TCP bağlantısı "ESTABLISHED" durumunda kaldığı ve istemciden paketler aldığı sürece tutacaktır. TCP bağlantısı koptuğunda bu kayıt tablodan silinir.

LVS tablodaki kayıtların süresinin dolması için 3 önemli timeout değeri kullanır.

- Tüm işlem yapılmayan TCP oturumları için bir timeout değeri
- İstemci bilgisayar bağlantıyı kopardıktan sonra TCP oturumları için bir timeout değeri (İstemci bilgisayardan FIN paketi alındıktan sonra)
- UDP paketleri için bir timeout değeri

Timeout değerleri her bir zaman sayacı için saniye olarak belirlenebilir. Bu işlem *ipvsadm* komutunun "--set " argümanı kullanılarak yapılır. Örneğin hareketsiz TCP oturumları için timeout değerini 4 saat, FIN paketi alındıktan sonraki TCP oturumları için 15 saniye ve UDP paketleri için 10 dakika belirlemek için aşağıdaki komut kullanılabilir.

```
#ipvsadm --set 14400 15 36000
```

Bu timeout değerlerini uygulamak için LVS iki tablo kullanır. Birincisi *ip_vs_timeout_table* olarak isimlendirilen bağlantı timeout tablosu, ikincisi ise *tcp_states* olarak isimlendirilen bağlantı durum tablosudur. *ipvsadm* komutuyla değiştirilen timeout değerleri LVS tarafından izlenen o anki ve gelecekteki bağlantıları etkiler.

4.10.3. Devamlı olan – Devamlı olmayan LVS

LVS kümeleri genelde devamlı olmayan LVS kümeleridir. Bu durumda kümeye gelen her bir bağlantı kümedeki en az yüklü olan sunucuya gönderilir. Buradaki amaç iş yükünün en uygun dağıtımıdır. Küme yapısında kullanılan IP Virtual Server tablosu aşağıdaki komutlarla oluşturulabilir. Kümenin Internet üzerinden erişilebilen sanal IP adresi 193.255.141.39 olarak kullanılmıştır.

```
/sbin/ipvsadm -A -t 193.255.141.39:80 -s wrr
/sbin/ipvsadm -a -t 193.255.141.39:80 -r 192.168.0.2 -m
/sbin/ipvsadm -a -t 193.255.141.39:80 -r 192.168.0.3 -m
```

İlk komut port 80 için 193.255.141.39 IP adresi için LVS sanal servisi yaratır. Sonraki iki komut ise bu sanal IP adresi için port 80 üzerinden iletilen isteklerin iletileceği düğümleri tanımlar. Yönetici her ağ bağlantısı isteği aldığında wrr (weighted round robin) zamanlama yöntemi kullanılarak en az bağlantısı olan düğüm seçilir. Eğer bir istemci kümeye iki adet telnet isteği gönderirse bu istekler kümedeki farklı düğümlere gönderilebilir.

Devamlı olan LVS yöntemi kullanmanın bir sakıncası vardır. Eğer birçok istemci aynı IP adresini kullanıyorsa (NAT işlemi yapılan bir firewall veya router arkasında ise) bu durumda yük dengeleme işleminde sorunlar olacaktır. Bu durumda ayrıca log işleminde sorunlar oluşacaktır.

Kullanılan LVS yöntemi ne olursa olsun eğer bir istemciden gelen tüm bağlantıların aynı düğüme gelmesi gerekiyorsa Devamlı Olan LVS yöntemi kullanılmalıdır. Devamlı Olan LVS yönteminin kullanılmasının amacı genelde aynı kullanıcının aynı uygulamayı çoklu çalıştırması zorunluluğu olduğunda program lisans ihlalleri olması ve lisansların boşa harcanmasıdır. Devamlılık SSL'de de gereklidir. Bunun nedeni bir SSL bağlantısı sırasında anahtar değişim işleminin gerekli olmasıdır.

Devamlı Olan LVS yönteminde Yönetici "persistent connection template (devamlı bağlantı şablonu şablonu)" adı verilen bir bağlantı takip kayıdı tutarak bir istemciden gelen tüm bağlantıların aynı düğüme atanmasını garanti eder. İstemci bilgisayar kümeye bağlantı istekleri gönderirken Yönetici her bir bağlantı için bağlantı takip kayıdı yaratır ve bu devamlı bağlantı şablonuna bakarak bu istemciden gelen bağlantılara daha önceden hangi düğümün atandığına karar verir.

Devamlı Olan LVS yönteminde Devamlı Olmayan LVS yöntemi gibi bir timeout değeri bulunmaktadır. Yöneticide *ipvsadm* komutu kullanılarak devamlı bağlantı şablonu için timeout değerleri belirlenir. Timeout değeri her devamlı bağlantı şablonu için bir zaman sayacı tanımlamak için kullanılır. Zaman sayacı değeri "ipvsadm –L –c" komutu kullanılarak görülebilir. Eğer sayaç sıfıra ulaşırsa ve bağlantı aktifse sayaç varsayılan değere getirilir ve sıfıra doğru tekrar azalmaya devam eder.

Devamlı Olan Bağlantı Türleri: Devamlı olan bağlantı türleri şunlardır;

1. PCC (Persistent Client Connections) / Zero Port Connections
2. PPC (Persistent Port Connections)
3. Persistent Netfilter Marked Packet Persistence
4. FTP Connections
5. Expired Persistence

En yaygın kullanılan Devamlı Olan Bağlantı türleri PPC, PCC ve Netfilter Marked Packet Persistence'dır.

PCC (Persistent Client Connection): PCC bir istemci bilgisayardan gelen tüm bağlantıların aynı düğümde sonlanmasını sağlar. PCC port numarası olmaksızın yaratılan bir sanal servistir. Bir istemci bilgisayardan gelen tüm bağlantıların aynı düğüme ulaşması isteniyorsa PCC kullanılır [3]. Örneğin HTTP protokolü kullanılan bir web kümesinde çalışan bir alışveriş sepetine müşteri ürünleri ekleyip "Ödeme" düğmesini tıkladığında şifrelenmiş HTTPS protokolü kullanılmaya başlanır. Dolayısıyla aynı istemciden gelen HTTP (port 80) ve HTTPS (port 443) bağlantıları aynı düğüme ulaşmalıdır. Kümenin Internet üzerinden erişilebilen sanal IP adresi 193.255.141.39 olarak kullanılmıştır.

```
/sbin/ipvsadm –A –t 193.255.141.39:0 –s rr –p
/sbin/ipvsadm –a –t 193.255.141.39:0 –r 192.168.0.2 –m
/sbin/ipvsadm –a –t 193.255.141.39:0 –r 192.168.0.3 –m
```

Yukarıdaki satırlar belirtilen 193.255.141.39 IP adresinde 192.168.0.2 ve 192.168.0.3 IP adreslerine sahip düğümlerden oluşan bir PCC sanal servisi oluşturur.

Varsayılan timeout değeri 360 saniyedir. Timeout değeri ile sadece belirtilen süre için geçerli olan bir PCC sanal servisi yaratılabilir. Örneğin aşağıdaki komutlar kullanılarak iki saat için geçerli olan PCC servisi yaratılabilir.

```
/sbin/ipvsadm –A –t 193.255.141.39:0 –s rr –p 7200
/sbin/ipvsadm –a –t 193.255.141.39:0 –r 192.168.0.2 –m
/sbin/ipvsadm –a –t 193.255.141.39:0 –r 192.168.0.3 –m
```

PPC (Persistent Port Connection): PPC bir istemci bilgisayardan gelen belirli bir portu hedefleyen tüm bağlantıların aynı düğümde sonlandırılmasını sağlar. Örneğin HTTP (port 80) ve TELNET (port 23) için PCC uygulamak için aşağıdaki komutlar kullanılabilir.

```
/sbin/ipvsadm –A –t 193.255.141.39:80 –s rr –p 7200
/sbin/ipvsadm –a –t 193.255.141.39:80 –r 192.168.0.2 –m
/sbin/ipvsadm –a –t 193.255.141.39:80 –r 192.168.0.3 –m
/sbin/ipvsadm –A –t 193.255.141.39:23 –s rr –p 7200
/sbin/ipvsadm –a –t 193.255.141.39:23 –r 192.168.0.2 –m
/sbin/ipvsadm –a –t 193.255.141.39:23 –r 192.168.0.3 –m
```

PCC ve PPC yöntemleri arasında temel bir fark vardır. PCC yönteminde aynı istemciden kümeye gelen bağlantılar aynı düğüme ulaşır. Dolayısıyla port 80 ile bağlı

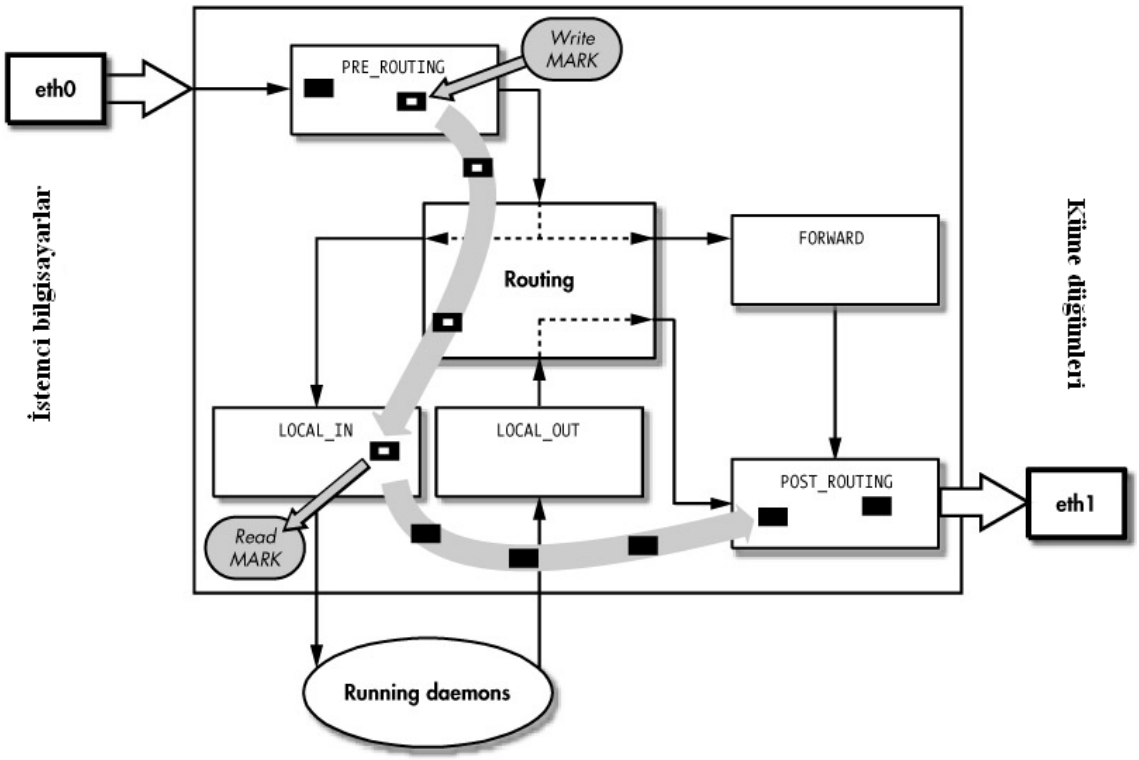
olan bir istemci port 443 üzerinden de bağlanmak istediğinde aynı sunucuya bağlanacaktır.

PPC yönteminde farklı portlar için IPVS tablosu yaratıldığında Yönetici istemcinin kullandığı her port için yeni bir bağlantı takip kayıdı yaratır [3].

Persistent Netfilter Marked Packet Persistence: Netfilter Marked Packets Yöneticideki ipchains veya iptables tarafından işaretlenir. Linux 2.2 çekirdeğinde firewall olarak ipchains, 2.4 ve 2.6 serisi çekirdeklerde ise firewall olarak iptables kullanılır. Netfilter tarafından işaretlenmesi paketi sadece Yönetici üzerindeyken etkiler. Paket Yöneticiyi terk ettiğinde işaret kalkar.

Bir paketin işaretlenmesi için bir kriter tanımlandığında bir işaret numarası atanır. Daha sonra bu numara *ipvsadm* kullanılarak bir IP sanal servisi ile ilişkilendirilir. Böylece işaretlenmiş paketler uygun olan düğüme gönderilecektir. Paket Yöneticideki çekirdek tarafıdan işaretlenirken Netfilter işaret numarası paket başlığında saklanmaz, "sk_buff" olarak adlandırılmış socket buffer da saklanır.

iptables kuralları paket *PRE_ROUTING* kancasında yönlendirme sürecinden geçmeden önce Netfilter işaret numarasının paket *sk_buff* girişinde saklanmasını sağlar. Paket yönlendirme sürecini tamamladıktan ve çekirdek hangi paketlerin yerel olarak teslim edilmesine karar verdikten sonra LOCAL_IN kancasına ulaşır. Bu noktada LVS paketi görür ve Netfilter işaret numarasını kontrol ederek hangi IP sanal servisinin kullanılacağına karar verir. Bu süreç aşağıdaki şekilde görülebilir.



Şekil 4.29. Netfilter tarafından işaretlenmiş paketler ve LVS

Yukarıdaki şekilde de görülebileceği gibi *PRE_ROUTING* kancasında Netfilter işareti gelen paket başlığına yerleştirilir. Daha sonra Netfilter işareti LOCAL_IN kancasındaki IPVS paketlerini araştıran LVS yazılımı tarafından okunabilir. Normal olarak *ipvsadm* tarafından sanal bir servis için ayarlanmış olan sanal servis adreslerini hedefleyen paketler LVS tarafından seçilir. Ayrıca işaret numarasına göre paket seçen sanal servisler kurulabilir.

iptables ile paketlerin işaretlenmesi: Örneğin küme sanal IP adresinde port 80 ve port 443 için iki saat devamlılığı olan bir PPC yaratmak için aşağıdaki komutlar kullanılmalıdır.

```
/sbin/iptables -F -t mangle
/sbin/iptables -A PREROUTING -i eth0 -t mangle -p tcp \
    -d 193.255.141.39/24 --dport 80 -j MARK --set-mark 96
/sbin/iptables -A PREROUTING -i eth0 -t mangle -p tcp \
    -d 193.255.141.39/24 --dport 443 -j MARK --set-mark 96
/sbin/ipvsadm -A -f 96 -s rr -p 7200
/sbin/ipvsadm -a -f 96 -r 192.168.0.2 -m
/sbin/ipvsadm -a -f 96 -r 192.168.0.3 -m
```

Yukarıdaki ilk komut iptables mangle tablosundaki tüm komutları siler. mangle tablosu *PRE_ROUTING* kancasına istenilen kuralların eklenebilmesini sağlar. İkinci ve üçüncü satırlardaki komutlar işaretleme için paketlerin seçilmesini sağlayan kriteri belirler. Küme sanal IP adresini hedefleyen paketler seçilmelidir. 80 ve 443 numaralı paketleri hedefleyen paketler 96 işaret numarası ile işaretlenmektedir. Son üç satır ise *ipvsadm* ile round robin scheduling zamanlama yöntemi ve LVS-NAT yöntemiyle paketleri 192.168.0.2 ve 192.168.0.3 IP adresli düğümlere göndermektedir.

Oluşturulmuş iptables içeriğini görüntülemek için aşağıdaki komut kullanılır.

```
#iptables –L –t mangle –n
Chain PREROUTING (policy ACCEPT)
target      prot opt source     destination
MARK        tcp  --  0.0.0.0/0  193.255.141.39    tcp dpt:80
MARK set 0x60
MARK        tcp  --  0.0.0.0/0  193.255.141.39    tcp dpt:443
MARK set 0x60

Chain OUTPUT (policy ACCEPT)
target      prot opt source     destination
```

iptables çekirdek Netfilter kurallarının yönetimini basitleştirmek için Netfilter kancalarının yerine tablo isimlerini kullanır. Örneğin *PREROUTING* olarak isimlendirilen zincir Netfilter *PRE_ROUTING* kancasıdır.

LVS IP Virtual Server yönlendirme kurallarını görüntülemek için aşağıdaki komut kullanılır.

```
#ipvsadm –L –n
IP Virtual Server version x.x.x (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port        Forward   Weight   ActiveConn
InActConn
FWM  96 rr persistent 7200
     -> 192.168.0.2:0             Masq      1        0
0
     -> 192.168.0.3:0             Masq      1        0
0
```

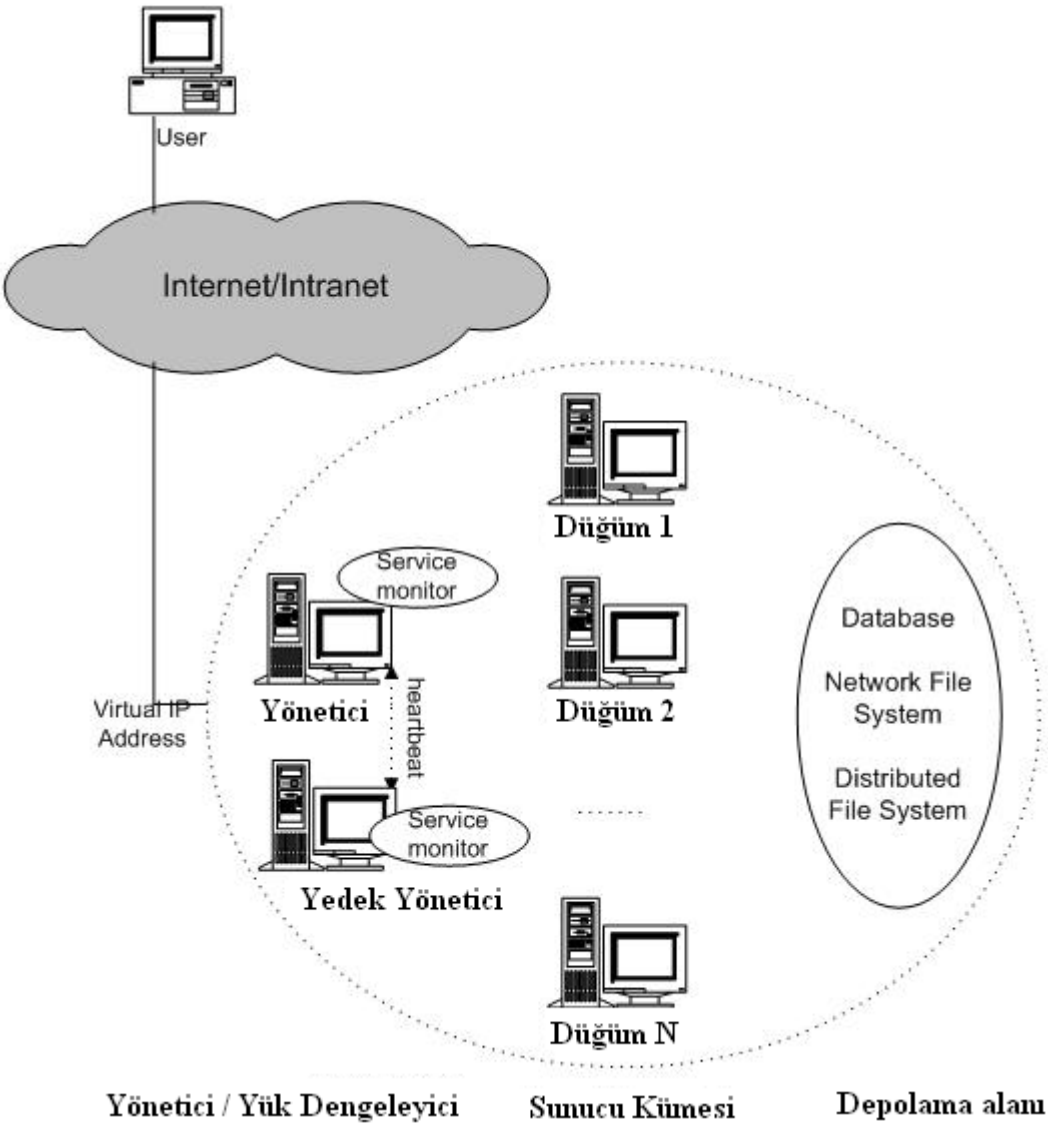
Yukarıdaki komutun çıktısında görülebileceği gibi 96 numarasıyla Netfilter tarafından işaretlenmiş paketler 192.168.0.2 ve 192.168.0.3 IP adresli düğümlere gönderilecektir.

4.11. Yüksek Erişilebilir Küme Yapıları

Yüksek Erişilebilir Küme Yapılarının amacı bir adet Yönetici düğüm kullanılmasından kaynaklanan tek hata noktası problemini ortadan kaldırmaktır. Yüksek Erişilebilirlik elde etmek için LVS yazılımı ile birlikte Heartbeat veya Keepalived yazılımları kurulmalıdır. Yüksek Erişilebilir Küme Yapıları elde etmek için ek yazılım paketlerinin yanında birden fazla LVS Yöneticisi kullanılmalıdır.

Linux dünyasında iki tip yüksek erişilebilir küme yapısı bulunmaktadır. Yüksek Erişilebilir IP Kümeleri ve Yüksek Erişilebilir Uygulama Kümeleri. Yüksek Erişilebilir IP Kümeleri ağ erişim noktalarının kullanılabilirliğini garantilemek için kullanılır. Bu kümeler ağ servislerine erişim isteyen IP adresleri kullanan istemcilere hizmet sunarlar. Yüksek Erişilebilir IP Kümeleri yüksek erişilebilirliği Linux Virtual Server (LVS) mekanizmasıyla gerçekleştirirler. Bu kümeler belirli uygulamalar için yük dengeleme işlemini de yerine getirebilirler. Yüksek Erişilebilir IP Kümelerininin yük dengeleme yapabileceği uygulamalardan bazıları Web, FTP ve video streaming servisleridir.

Yüksek Erişilebilir Uygulama Kümeleri ise veritabanı sunucuları, web uygulama sunucuları, dosya sunucuları ve yazdırma sunucuları gibi uygulamalar için daha uygundur. Bu kümeler erişilebilirliği sağlamak için uygulamaların failover özellikleri yanından uygulamaların ihtiyaç duyduğu disk, IP adresi ve yazılım modülleri gibi tüm bileşenleri de kullanırlar.



Şekil 4.30. Yüksek Erişilebilir Küme Çalışma Prensibi

Yüksek Erişilebilir Küme yapısında depolama alanı veritabanı sistemleri (database systems), ağ dosya sistemleri (network file systems) veya dağıtık dosya sistemleri (distributed file systems) olabilir. Sunucu kümesini oluşturan düğümlerin dinamik olarak güncellemesi gereken veriler veritabanlarında saklanmalıdır. Düğümlerin veritabanı sistemlerinden paralel olarak okuması veya yazması gerektiğinde veritabanı sistemleri verinin tutarlılığını garanti edebilir. Statik veriler ise genellikle NFS ve CIFS gibi ağ dosya sistemlerinde saklanır. Böylece veriler bütün sunucu düğümleri tarafından paylaşılabilir. Ancak ağ dosya sistemlerinin genişleyebilirliği sınırlıdır. Örneğin tek bir NFS/CIFS 4-8 arası erişimi destekler. Daha büyük küme

sistemlerinde ortak depolama alanı için GPFS, Coda ve GFS gibi dağıtık/küme dosya sistemleri kullanılabilir.

4.11.1. Yüksek erişilebilir IP kümeleri

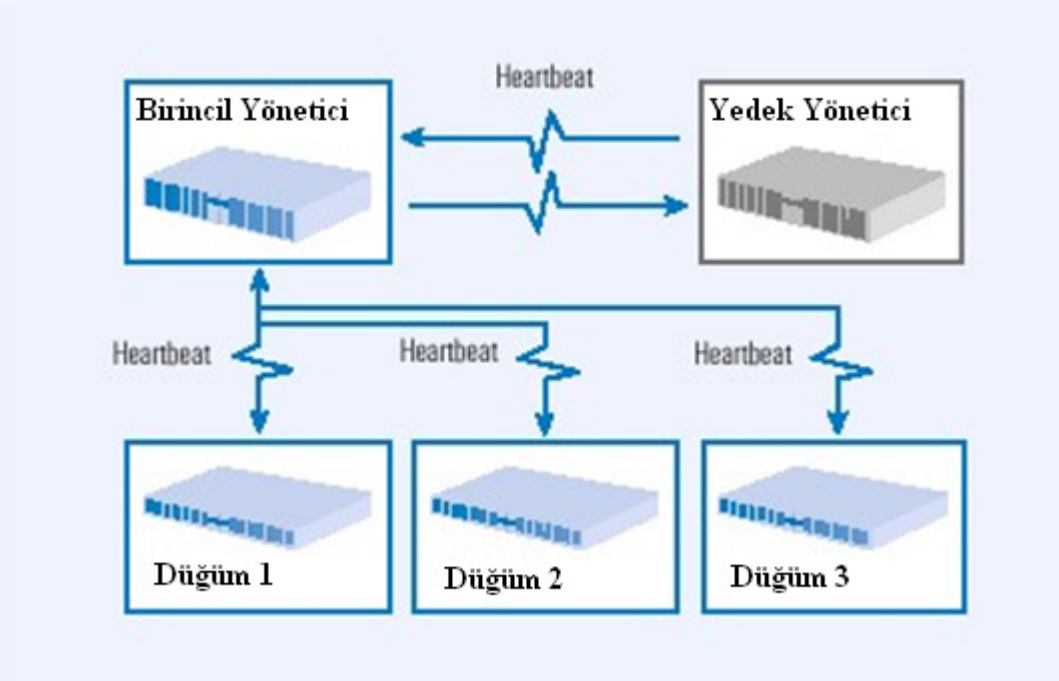
Yüksek Erişilebilir IP Kümeleri genellikle LVS mekanizmasını kullanırlar. Bu kümelerde kullanılan uygulamalara örnek olarak Piranha, Red Hat High Availability Server 1.0, TurboLinux® TurboCluster ve Ultra Monkey verilebilir.

LVS bir uygulama sunucusu kümesini istemcilere tek bir sistem olarak gösterir. LVS kümelenmiş servislere erişim için istemcilere sanal IP adresleri sunar. TCP veya UDP portları kullanan servisler sunabilir. LVS istemcilerden gelen istekleri sunucu kümesindeki düğümlere dağıtır. İstemciler küme yapısında bulunan düğümlerden ve bu düğümler tarafından kullanılan IP adreslerinden habersizdirler. Kümeye erişmenin tek yolu LVS tarafından kullanılan sanal IP adresidir.

Yöneticide çalışan LVS sunucu gelen istekleri küme yapısında bulunan düğümelere önceden belirlenmiş bir politikaya göre iletir. Eğer düğümlerden birisinde sorun meydana gelirse yeni gelecek istekler bu düğüme gönderilmez. Eğer yeni bir düğüm eklenirse bu düğümü de kümeye ilave eder. LVS sadece sanal IP adresleri üzerinden kümeye erişime izin verir.

Heartbeats Monitor servisi kesintisiz hizmet sağlayan özelliktir. LVS sürekli olarak uygulama sunucularının durumlarını izler. LVS sürekli olarak düğümlerin durumlarını izlediği için hatayı anında tespit eder ve küme üyelik durumunu değiştirir. Bu izleme işleminde hearbeat monitor servisi kullanılır. Heartbeat paketleri kümedeki düğümler arasında belirli periyodlarla gönderilir. Eğer belirli bir süre içinde heartbeat alınamazsa mesaj alınmayan düğümde sorun olduğu kabul edilir. Sorun meydana gelen düğüme LVS sunucu istemcilerden gelen istekleri yönlendirmez. Heartbeat protokolü TTY seri port veya Ethernet ağında UDP/IP üzerinden çalışabilir.

Kesintisiz hizmet sunulabilmesi için küme tarafından sunulan uygulama her bir düğüm üzerine kurulmalıdır. Bir düğüm üzerindeki uygulamada yapılacak düzenlemeler diğer düğümlere de aktarılmalıdır. Bu senkronizasyon işleminde genelde rsync kullanılır.



Şekil 4.31. Düğümlerin Durumlarının İzlenmesi

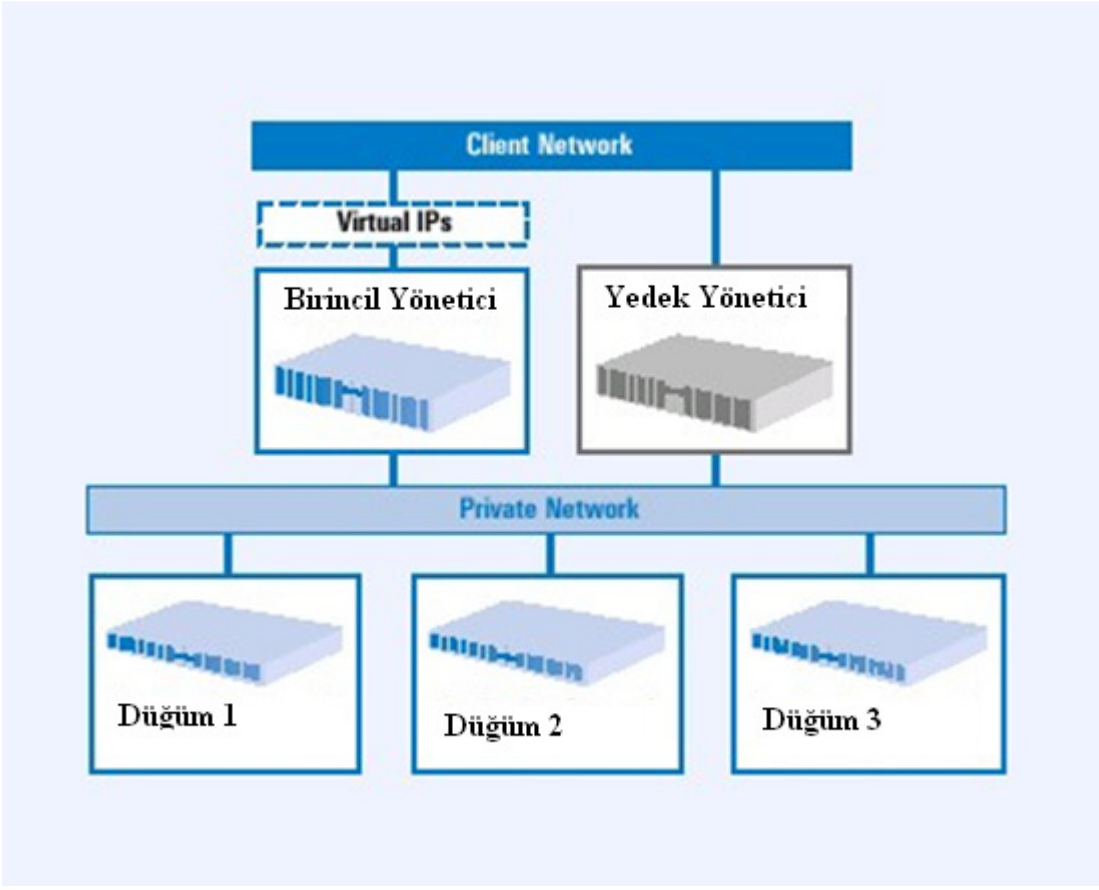
Yüksek Erişimli IP Kümeleri sadece bir IP adresinin erişilebilirliğini sağlamaz. Bunun yanında gelen istemci isteklerini küme yapısındaki düğümlere yük dengeleme yaparak gönderebilir. Yük dengeleme kullanıldığında eğer talep artışı olursa yapılması gereken küme yapısındaki düğümlere yeni düğümler ilave etmektir.

Gelen istemci isteklerini veya iş yükünü düğümler arasında dağıtabilmek için LVS zamanlama politikaları kullanır. En yaygın kullanılan zamanlama politikaları round robin ve least connections yöntemleridir. Round robin yöntemi gelen istekleri belirli bir sırada düğümlere dağıtır. Least connections yöntemi ise gelen isteği en az yükü olan sunucuya gönderir.

Yedekli LVS Yöneticileri: Yüksek Erişilebilir LVS Kümesinde Birincil ve Yedek LVS Yöneticileri olmak üzere iki adet LVS Yöneticisi bulunmalıdır. İki Yönetici de

boot ettiği anda, Birincil Yönetici kümenin yük dengeleme işlemlerini üzerine alır. Yedek Yönetici birincil Yöneticiden gelen hearbeat mesajlarını dinler. Eğer Yedek Yönetici Birincil Yöneticiden gelen heartbeat mesajlarını alamazsa devralma işlemini gerçekleştirir ve kümenin yük dengeleme işlemlerini üzerine alır. Bu devralma işlemi sırasında aşağıdaki işlemleri gerçekleştirir;

1. VIP adresini ağ kartlarının birisine ekler.
2. Ağdaki diğer makinelere GARP broadcast paketleri göndererek VIP adresine sahip olduğunu bildirir.
3. IPVS tablosu yaratarak sanal servisler için gelen isteklerin yük dengelemesini yapar.



Şekil 4.32. Yedekli LVS Mekanizması

4.11.2. Yüksek erişilebilir küme tasarımı

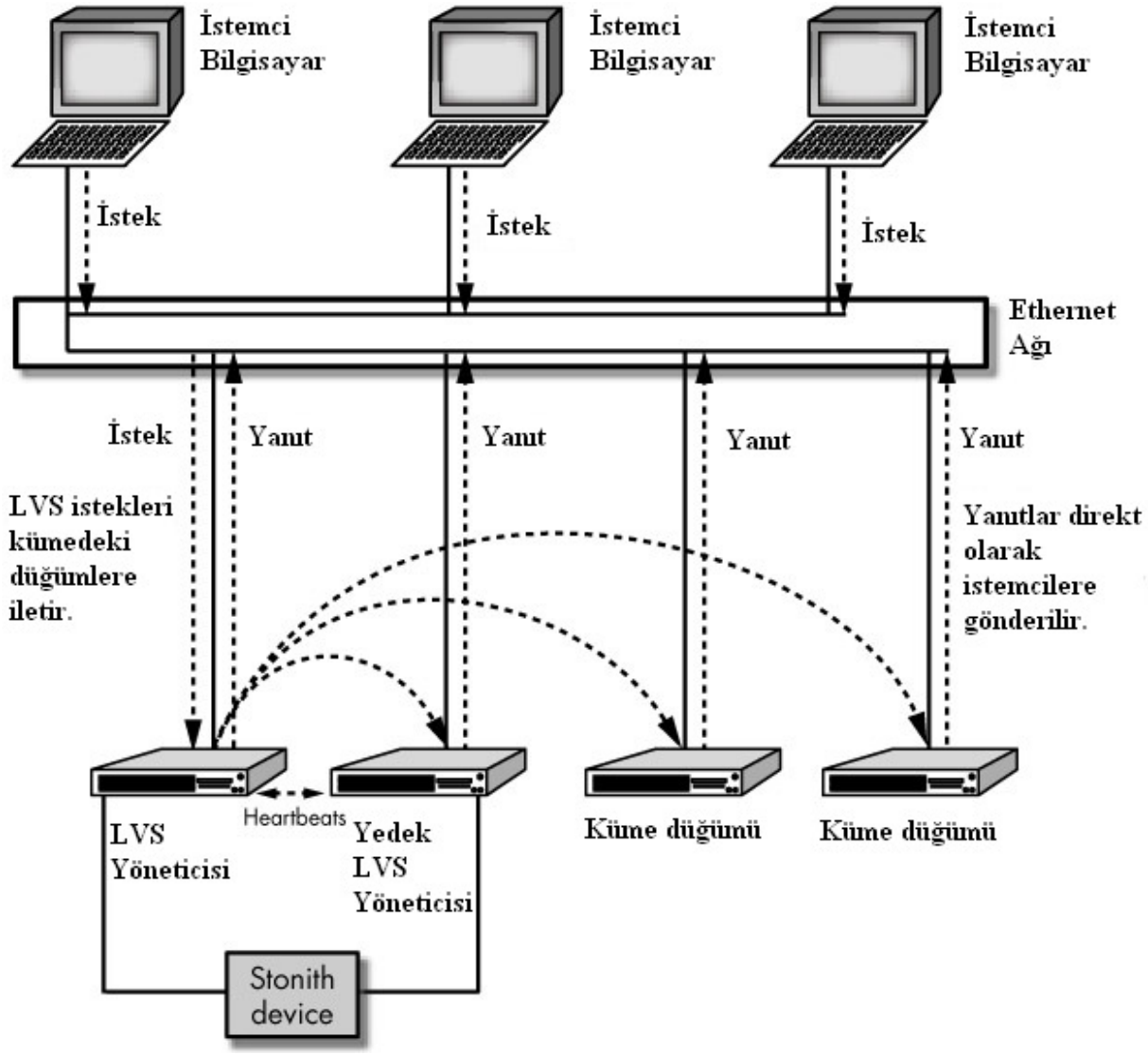
Tek hata noktası problemini ortadan kaldırarak yüksek erişilebilirliği sağlamak için küme şu durumlara cevap verebilmelidir;

- Eğer Birincil Yönetici dış ağa bağlı olan ağ kartında küme kaynakları için gelen isteklere yanıt veremezse Yedek Yönetici görevi devralmalı ve Birincil Yöneticiyi kapatmalı veya yeniden başlatmalıdır.
- Eğer küme düğümlerinden birisi servis isteklerine yanıt vermiyorsa düğüm kümeden çıkarılmalıdır.
- Birincil ve Yedek Yöneticiler ağ donanımından kaynaklanan tek hata noktası problemini aşmak için ayrı fiziksel switch / hub ile ağa bağlı olmalıdır.
- Paylaşılmış dosya sisteminden kaynaklanan tek hata noktası problemini aşmak için NAS kullanılmalıdır.

4.11.3. Yüksek erişilebilir LVS-DR kümesi

İki adet Yönetici bulunan Yüksek Erişilebilir LVS Kümesi aşağıdaki şekilde görülmektedir. Küme iki adet Yönetici, iki adet düğüm ve üç adet istemci bilgisayardan oluşmaktadır. Yöneticiler arasında haberleşmeyi sağlayan heartbeat paketleri Ethernet ağı üzerinden veya seri kablo bağlantısı üzerinden taşınabilir.

Şekilde görülen Stonith cihazı gerçek yüksek erişilebilirliği sağlar. Yedek LVS Yöneticisi Stonith cihazına seri bağlantı veya ağ bağlantısıyla bağlıdır. Böylece Heartbeat güç komutları cihaza iletilebilir. Birincil Yöneticinin güç kablosu Stonith cihazına bağlıdır.



Şekil 4.33. Yüksek Erişilebilir LVS-DR Kümesi

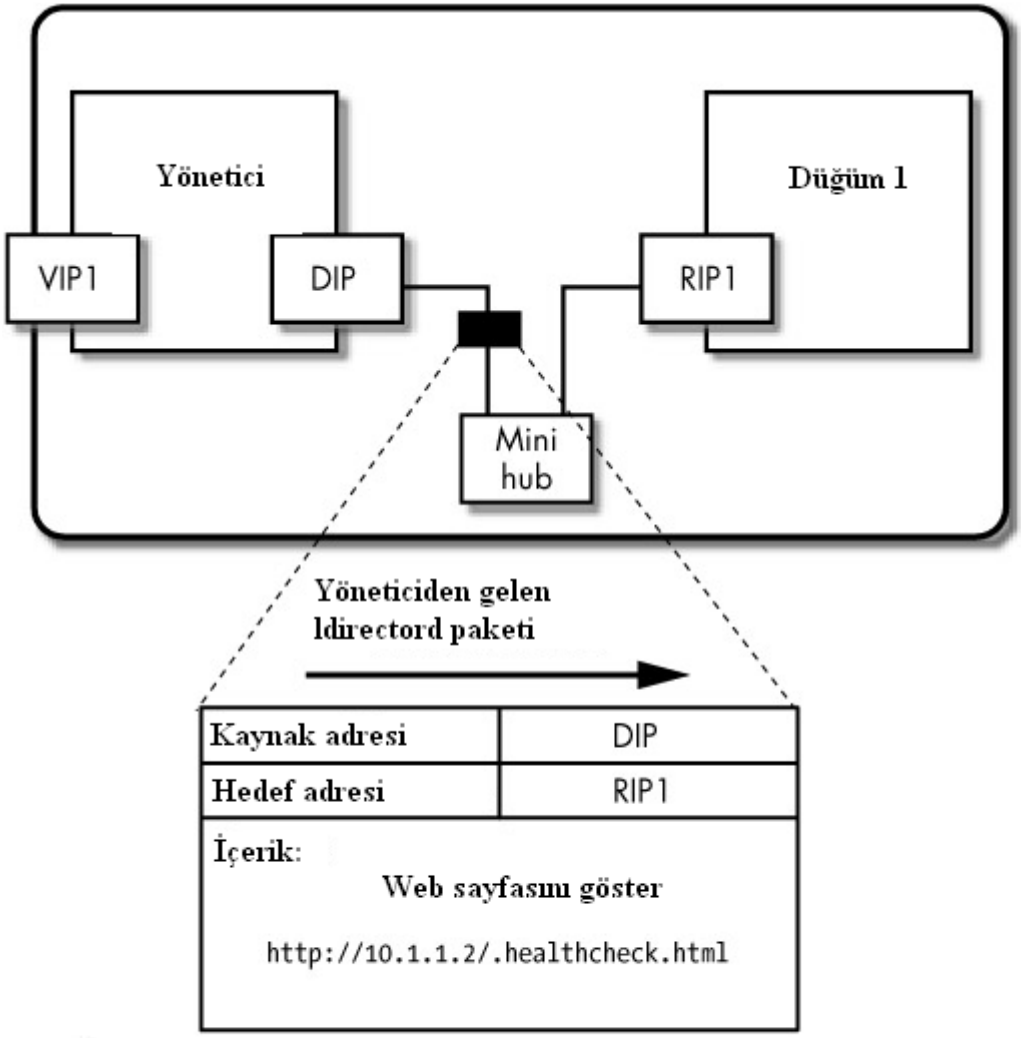
Yukarıdaki şekildeki LVS Yöneticisinden küme düğümlerine giden kesikli çizgili oklar istemci bilgisayarlardan gelen paketleri gösterir. LVS-DR Kümesi olduğu için küme düğümlerinden gelen yanıt paketleri istemci bilgisayarlara direkt olarak ulaşır. İşlem gücü gereksinimi arttığında kolaylıkla yeni düğümler kümeye eklenebilir.

4.11.3.1. ldirectord

LVS yük dengeleme işleminin Birincil Yöneticiden Yedek Yöneticiye geçmesi ve otomatik olarak düğümlerin kümeden çıkarılması için *ldirectord* programı kullanılır. Program ilk çalıştığında otomatik olarak IPVS tablosunu oluşturur ve sonra küme düğümlerinin durumunu izler. Düğümlerden birisinde sorun meydana gelirse düğüm IPVS tablosundan çıkarılır.

ldirectord daemonu düğümlerin durumunu sürekli olarak izler. Normal olarak Yöneticideki her bir sanal IP adresi için bir ldirectord daemonu çalışır. Düğümlerden birisi Yöneticide çalışan ldirectord daemonuna yanıt vermezse ldirectord ipvsadm komutu ile bu düğümü bu sanal IP adresi için IPVS tablosundan çıkarır. Eğer düğüm tekrar çalışmaya başlarsa ldirectord bu düğümü tekrar IPVS tablosuna ekler.

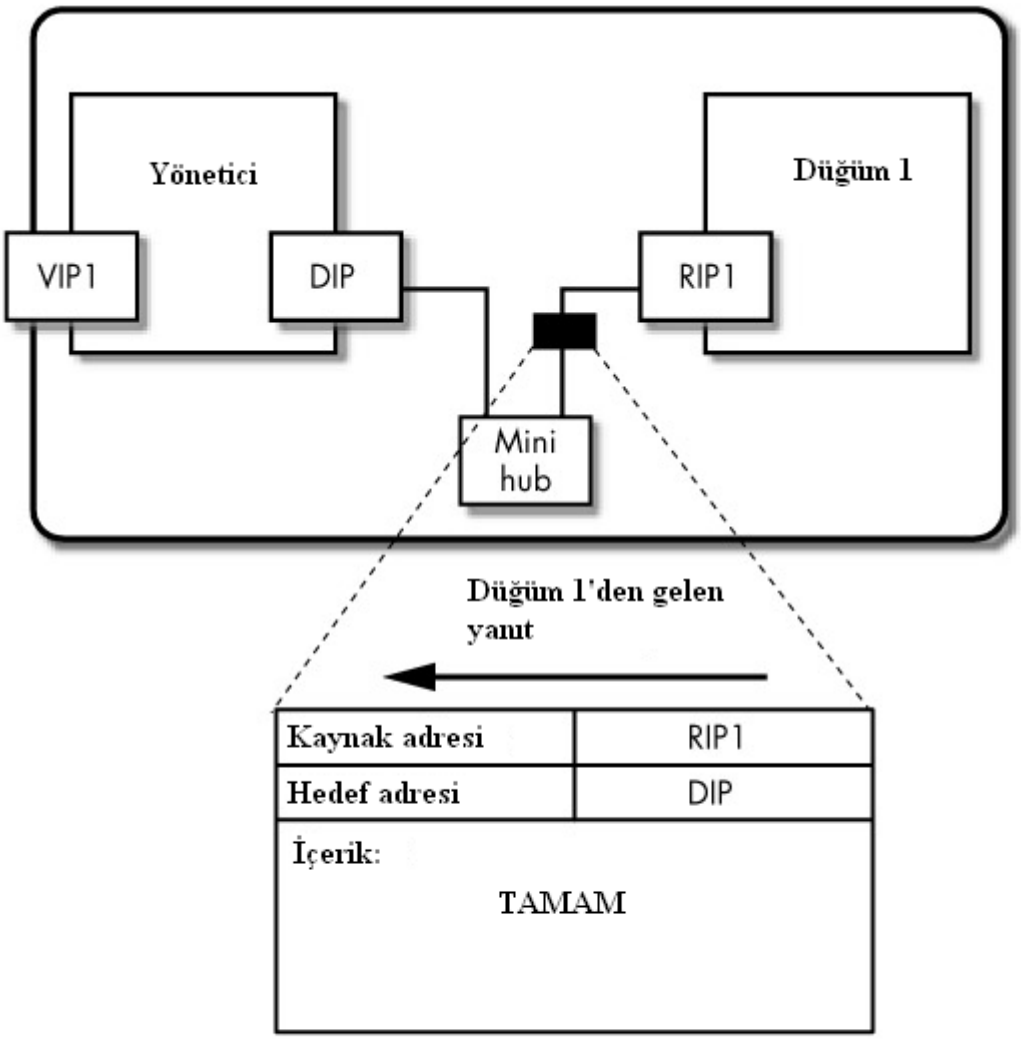
Bir web sunucu kümesindeki düğümleri izlemek için ldirectord daeomonu http protokolünü kullanarak özel bir web sayfasını düğümlerden ister. Yönetici düğümlerden yanıt aldığında düğüm sorunsuz olarak çalışmaktadır. Eğer yanıt dizisi veya web sayfasının düğümden gelmesi çok uzun zaman alırsa veya düğümden hiç yanıt alınamazsa ldirectord bu düğümü belirtilen sanal IP adresi için IPVS tablosundan çıkarır.



CIP: İstemci Bilgisayarın IP Adresi
VIP: Kümenin Sanal IP Adresi
RIP: Düğümün IP Adresi
DIP: Yöneticinin Küme İçindeki IP Adresi

Şekil 4.34. Düğümün healthcheck.html dosyası ile ldirectord tarafından izlenmesi

ldirectord düğümden dosyayı ister ve düğümde çalışan Apache web sunucusu ldirectord daemonuna yanıt verir.



Şekil 4.35. Düğüm 1'in geriye yanıt paketini göndermesi

Eğer düğüm ldirectord daemonuna yanıt vermezse IPVS tablosundan çıkarılır. Küme düğümlerinin izlenmesi işlemleri kullanılan LVS yöntemi ne olursa olsun düğümlerin RIP gerçek IP adresleri kullanılır.

4.11.3.2. Gereksiz Apache mesajlarının devre dışı bırakılması

Apache normal olarak düğümlerin durumunun takip edilmesi için kullanılan sayfaya her erişildiğinde bir kayıt yazar. Bu nedenle access_log ve disk sürücü kısa zamanda gereksiz bilgilerle dolacakır. Bu duruma engel olmak için httpd.conf dosyasına aşağıdaki satır eklenmelidir.

```
SetEnvIf Request_URI \.healthcheck\.html$ dontlog
```

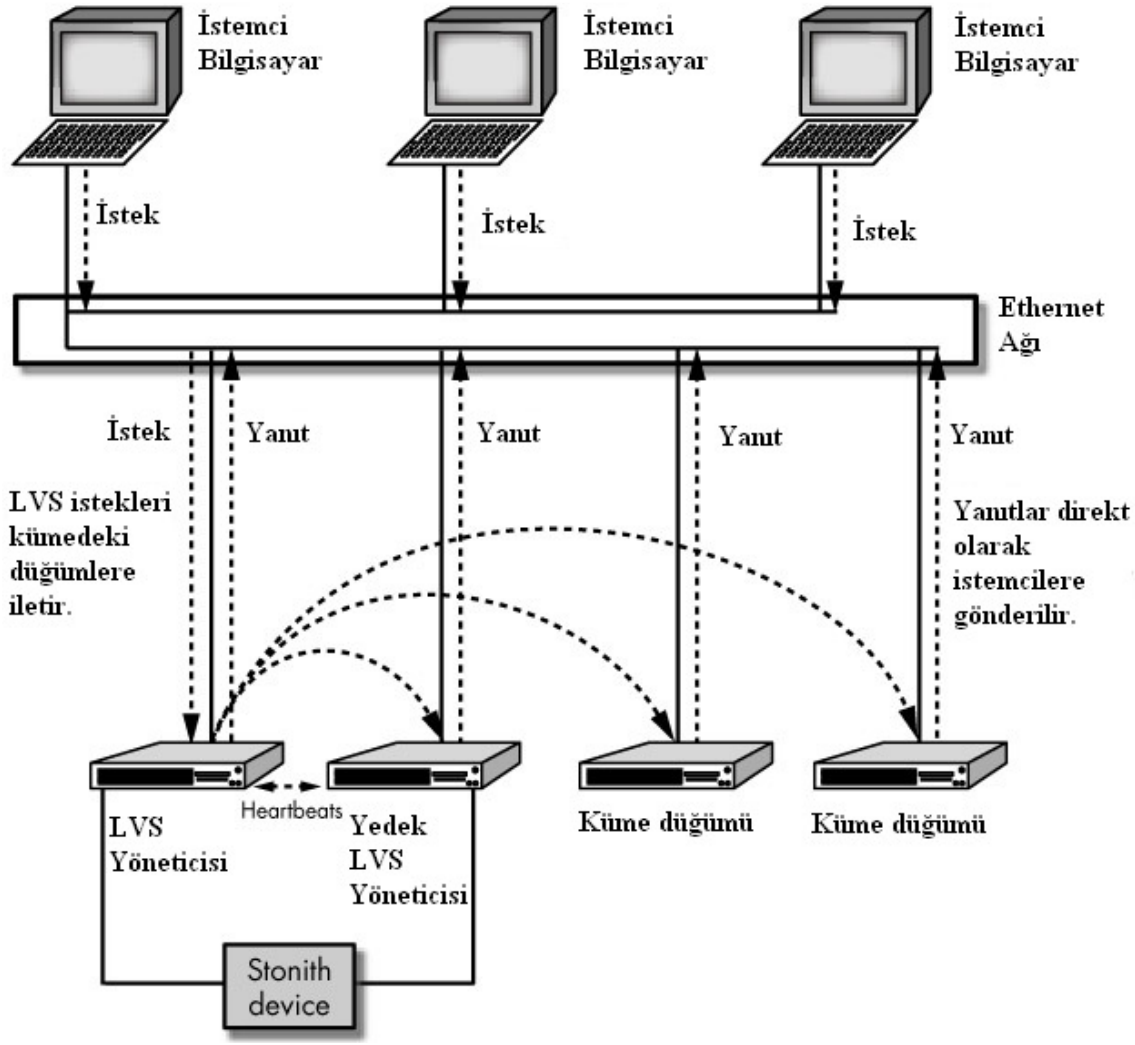
Yukarıdaki satır httpd.conf dosyasında access_log için CustomLog satırından önce yazılır. Daha sonra CustomLog satırı aşağıdaki şekilde değiştirilir.

```
CustomLog logs/access_log combined env=!dontlog
```

Son olarak Apache daemon'u tekrar başlatılır.

4.11.3.3. Yüksek erişilebilir LVS kümesinin kurulumu

Aşağıdaki şekil kurulacak LVS kümesini göstermektedir.



Şekil 4.36. Yüksek Erişilebilir LVS-DR Kümesi

Loopback Arabiriminin Gizlenmesi: İstemci bilgisayarlardan gelen ve VIP adresinin sahibini (MAC adresini) arayan ARP broadcast paketlerinin kümedeki düğümler tarafından önemsenmemesi için gerekli ayarlar yapılmalıdır.

Bunu gerçekleştirmek için iki yöntem vardır. Birinci yöntem sistem boot ettiğinde init programı tarafından başlatılacak bir script oluşturmaktır. İkinci yöntem `/etc/sysctrl.conf` dosyasını değiştirmek ve sistemin her boot edişinde "sysctl –p" komutunu çalıştırmaktır.

Birinci yöntemde kullanılabilecek script aşağıdaki işlemleri gerçekleştirir.

- VIP adresini gizli loopback aygıtına ekler.

- VIP adresi için bir yönlendirme tablosu satırı ekler.
- Paket iletmeyi devreye alır.
- Çekirdeğin VIP adresini gizlemesi sağlanır.

Bu script düğümlerde, Birincil ve Yedek Yöneticilerde kullanılabilir. Çünkü Heartbeat programı gizlenmiş loopback cihazındaki bir VIP adresinin */etc/ha.d/haresources* dosyasında tanılanmış bir IP adresi ile çakışma olduğunda nasıl kaldırılacağını bilir. Diğer bir deyişle Heartbeat programı Birincil Yöneticide ilk çalıştığında VIP adresini gizlenmiş loopback cihazından kaldırır. Birincil Yöneticide sorun olduğunda Yedek Yöneticinin VIP adresini alması gerektiğinde aynı işlem Yedek Yöneticide de gerçekleştirilir.

Script aşağıda görülmektedir.

```
#!/bin/bash
#
# lvsdrrs init script to hide loopback interfaces on LVS-DR
# Real servers. Modify this script to suit
# your needs—You at least need to set the correct VIP
address(es).
#
# Script to start LVS DR real server.
#
# chkconfig: 2345 20 80
# description: LVS DR real server
#
# You must set the VIP address to use here:
VIP=209.100.100.2
host=`/bin/hostname`
case "$1" in
start)
       # Start LVS-DR real server on this machine.
        /sbin/ifconfig lo down
        /sbin/ifconfig lo up
        echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore
        echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce
        echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
        echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

        /sbin/ifconfig lo:0 $VIP netmask 255.255.255.255 up
        /sbin/route add -host $VIP dev lo:0

;;
stop)

        # Stop LVS-DR real server loopback device(s).
        /sbin/ifconfig lo:0 down
```

```
        echo 0 > /proc/sys/net/ipv4/conf/lo/arp_ignore
        echo 0 > /proc/sys/net/ipv4/conf/lo/arp_announce
        echo 0 > /proc/sys/net/ipv4/conf/all/arp_ignore
        echo 0 > /proc/sys/net/ipv4/conf/all/arp_announce

;;
status)

        # Status of LVS-DR real server.
        islothere=`/sbin/ifconfig lo:0 | grep $VIP`
        isrothere=`netstat -rn | grep "lo:0" | grep $VIP`
        if [ ! "$islothere" -o ! "isrothere" ];then
            # Either the route or the lo:0 device
            # not found.
            echo "LVS-DR real server Stopped."
        else
            echo "LVS-DR Running."
        fi
;;
*)
            # Invalid entry.
            echo "$0: Usage: $0 {start|status|stop}"
            exit 1
;;
esac
```

Bu script tüm düğümlerde /etc/init.d dizinine kopyalanır ve chkconfig kullanılarak devreye alınır. Böylece sistem her boot ettiğinde çalışacaktır. Benzer bir init scripti oluşturacak installation scripti Linux Virtual Server web sitesinde (http://www.linuxvirtualserver.org) bulunmaktadır.

Birincil ve Yedek Yöneticilerde Heartbeat Programının Kurulumu: Heartbeat programı ldirectord programını başlatmak ve Birincil Yöneticideki VIP IP alias veya ikincil IP adresleri ayağa kaldırmak için kullanılır. ldirectord programı ve VIP Heartbeat programının kontrolü altında bulunan bir kaynak grubu olacaktır. Eğer Birincil Yöneticide sorun meydana gelirse Yedek Yöneticide çalışan Heartbeat programı bu kaynak grubunu devralacak ve istemci bilgisayarlar kümeye erişmeye devam edecektir.

ldirectord ve Zorunlu Olan Diğer Yazılım Bileşenlerinin Kurulumu: ldirectord bir Perl programı olup önceden yazılmış olan birçok Perl modülünü kullanır. Bu modüller HTTP, POP, TELNET gibi servislere bağlanmayı sağlar.

Gerekli olan Perl modülleri Internet ten indirilebilir. Örneğin Internetten indirilmiş ve tmp dizinine kopyalanmış olan modüller aşağıdaki komutlar kullanılarak açılıp kurulabilir.

```
#cp –r /tmp /usr/local/src/perldeps
#cd /usr/local/src/perldeps
#tar xzvf libnet*
#cd libnet-<version>
#perl Makefile.PL
#make
#make install
```

ldirectord kurulumu için aşağıdaki komutlar kullanılabilir. Örneğin Internet ten indirilip tmp dizinine kopyalanmış olan ldirectord aşağıdaki komutlarla kurulabilir.

```
#cp /tmp/ldirectord* /etc/ha.d/resource.d/ldirectord
#chmod 755 /usr/sbin/ldirectord
```

Daha sonra ipvsadm programı kurulmalıdır. ldirectord kurulumu tamamlandıktan sonra aşağıdaki komutla test edilebilir. Komut ldirectord yardım sayfasını ekrana getirecektir.

```
#/usr/sbin/ldirectord –h
```

Kurulumdan sonra ldirectord konfigürasyon dosyası düzenlenmelidir. ldirectord bu dosyayı IPVS tablosunu oluşturmak için kullanır. Dosya `/etc/ha.d/conf` dizinine kopyalanmalıdır. Örnek bir konfigürasyon dosyası aşağıda bulunmaktadır.

```
checktimeout=20
checkinterval=5
autoreload=yes
quiescent=no
logfile="info"
virtual=209.100.100.3:80
    real=127.0.0.1:80 gate 1 ".healthcheck.html", "TAMAM"
    real=209.100.100.100:80 gate 1 ".healthcheck.html", "TAMAM"
    service=http
    checkport=80
    protocol=tcp
    scheduler=wrr
    checktype=negotiate
    fallback=127.0.0.1
```

Dosyadaki ilk dört satır global ayarlardır ve çoklu sanal hostlar için uygulanabilen ayarlardır.

"checktimeout=20" satırı düğümlerin durumlarının kontrolünün tamamlanması için beklenmesi gereken süreyi saniye olarak belirler. Eğer kontrol belirtilen sürede tamamlanamazsa düğüm IPVS tablosundan çıkarılır.

"checkinterval=5" satırı kontroller arasında ldirectord'nin ne kadar beklemesi gerektiğini belirler.

"autoreload=yes" satırı ldirectord'nin periyodik olarak değişiklikler için konfigürasyon dosyasını kontrol etmesi ve md5sum değeri hesaplaması ve otomatik olarak dosya değiştiğinde bunları otomatik olarak uygulamasını sağlar. Aynı işlem ldirectord daemonuna kill komutuyla HUP sinyali gönderilmesi veya "ldirectord reload" komutuyla gerçekleştirilebilir.

"quiescent=no" satırı düğüm checktimeout süresinde cevap vermediğinde düğümü hareketsiz duruma getirir. Bu seçenek aktif ise ldirectord düğümü hareketsiz duruma getirmek yerine IPVS tablosundan çıkarır. Düğüm IPVS tablosundan çıkarıldığında tüm istemci bağlantıları koparılır ve LVS tüm bağlantı izleme kayıtlarını siler.

"logfile="info"" ldirectord'nin syslog ile hata mesajlarını kaydetmesini sağlar. Log mesajları /var/log/ldirectord.log dizinine kaydedilir.

"virtual=209.100.100.3:80" satırı LVS Yönetici üzerinde tanımlanacak VIP adresini ve port numarasını belirtir. Bu IP adresi istemci bilgisayarların kümeye ulaşırken kullanacağı IP adresidir.

Aşağıdaki satır küme içindeki hangi düğümlerin istemci bilgisayarlara hizmet vereceğini belirler. Loopback adresi olan 127.0.0.1 IP adresi Yöneticininde LocalNode modunda çalıştığını ve 209.100.100.3 VIP adresinde istemci bilgisayarlara hizmet vereceğini belirtir.

```
real=127.0.0.1:80 gate 1 ".healthcheck.html", "TAMAM"
```

Aşağıdaki satır ilk LVS-DR düğümünün 209.100.100.100 IP adresini kullanacağını belirtir.

```
real=209.100.100.100:80 gate 1 ".healthcheck.html", "TAMAM"
```

"real" ifadesi kullanılan her satır aşağıdaki dizilişi kullanır.

```
real=RIP:port gate|masq|ipip [ağırlık] "İstenen URL", "Beklenen
cevap"
```

Bu dizilimde kullanılabilecek gate, ipip ve masq seçenekleri düğümlere iletimde kullanılacak LVS iletim yöntemini belirler.

ldirectord Konfigürasyon Seçeneği	**ipvsadm İle Kullanılan Seçenek**	**ipvsadm -L Komutunun Çıktısı**	**LVS İletim Yöntemi**
gate	-g	Route	LVS-DR
ipip	-i	Tunnel	LVS-TUN
masq	-m	Masq	LVS-NAT

Tablo 4.4. LVS İletim Yöntemleri

LVS iletim yönteminden sonraki weight parametresi sadece ağırlık parametresi kullanan zamanlama yöntemleri için geçerlidir. Son iki parametre ldirectord'nin düğümün durumunu kontrol ederken kullanacağı web sayfasını ve ldirectord'nin düğümden hangi cevabı beklediğini belirtir.

"`service=http`" satırı ldirectord'nin düğümün durumunu izlerken kullanacağı servisi belirler.

"`checkport=80`" satırı http servisinin port 80'i kullanacağını belirler.

"`protocol=tcp`" satırı sanal servis tarafından kullanılacak protokolü belirler. Kullanılabilecek protokol *tcp*, *udp* veya *fwm* olabilir. Eğer *fwm* kullanılırsa virtual=line satırında IP adresi yerine Netfilter işaret numarası kullanılmalıdır.

"`scheduler=wrr`" weighted round-robin yük dengeleme kullanılacağını belirtir. wrr dışında farklı zamanlama yöntemleri de bulunmaktadır.

"checktype=negotiate" satırı ldirectord'nin düğümleri izlerken kullanacağı yöntemi belirler.

Checktype seçeneği aşağıdakilerden birisi olabilir.

negotiate: Bu yöntem düğüme bağlanır ve belirtilen *request* ifadesini gönderir. Eğer tanımlanan *reply* ifadesi düğümden belirtilen checktimeout süresi içinde alınamazsa düğüm ölü olarak değerlendirilir.

connect: Bu yöntem belirtilen checkport u kullanarak düğüme bağlanır. Eğer düğüm basit TCP/IP bağlantısını kabul ederse her şeyin sorunsuz olduğu kabul edilir.

numara: Eğer negotiate veya connect ifadesi yerine bir numara tanımlanırsa ldirectord tanımlanan kadar sayıda bağlantıyı test eder ve daha sonra negotiate testini yapar.

off: Bu yöntem kullanılırsa ldirectord düğümlerin durumunu izlemez.

"fallback=127.0.0.1" satırı IPVS tablosunda düğüm kalmadığında servisin yönlendirileceği IP adresini ve port numarasını belirler. Port numarasıda aşağıdaki gibi kullanılabilir.

```
fallback=127.0.0.1:9999
```

Düğümlerin Durumlarının İzlenmesi İçin Kullanılan healthcheck.html Dosyası: Düğümlerde ve Yöneticide aşağıdaki komut kullanılarak izleme işlemi için kullanılabilecek basit bir web sayfası oluşturulabilir.

```
#echo "TAMAM" > /www/htdocs/.healthcheck.html
```

Eğer httpd.conf dosyasında DocumentRoot olarak belirtilen dizin /www/htdocs/ dizininden farklı ise bu dizin adı kullanılmalıdır.

Düğümlerde aşağıdaki komut kullanılarak oluşturulmuş olan sayfa kontrol edilebilir.

```
#lynx –dump 127.0.0.1/.healthcheck.html
```

ldirectord'nin Manuel Olarak Başlatılması ve Konfigürasyonun Test Edilmesi: İlk olarak aşağıdaki komutla daemon başlatılmaya çalışılır.

```
#/etc/ha.d/resource.d/ldirectord -d ldirectord-
209.100.100.3.conf start
```

Komut kullanılınca ekrana aşağıdaki debug mesajı gelecektir.

```
DEBUG2: check_http: http://209.100.100.100/.healthcheck.html is
down
```

Sonraki adımda Apache web sunucu çalıştıran düğüm başlatılır. Düğüm çalıştığında ekrana aşağıdaki debug mesajı gelecektir.

```
DEBUG2: check_http: http://209.100.100.100/.healthcheck.html is
up
```

Daha sonra aşağıdaki komut kullanılarak sanal servisin IPVS tablosuna eklenip eklenmediği kontrol edilir.

```
#ipvsadm -L -n
```

Eğer ldirectord konfigürasyonu doğru ise Ctrl-C tuş kombinasyonu kullanılarak debug işlemi sonra erdirilir.

Son olarak ldirectord aşağıdaki komutla tekrar başlatılır.

```
#/usr/sbin/ldirectord ldirectord-209.100.100.3.conf start
```

ipvsadm komutu watch komutuyla başlatılarak ekrana gelecek görüntünün otomatik olarak sürekli güncellenmesi sağlanır.

```
#watch ipvsadm -L -n
```

ipvsadm komutu aşağıdaki şekildeki gibi bir görüntü ekrana getirecektir.

IP Virtual Server version x.x.x (size=4096)

```
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port     Forward    Weight    ActiveConn
InActConn
TCP  209.100.100.3:80 wrr
  -> 209.100.100.100:80     Route      1         0              0
  -> 127.0.0.1:80           Local      1         0              0
```

Test etmek için düğümde çalışan Apache web sunucusu durdurulabilir veya düğümün ağ kartına bağlı olan ağ kablosu çekilebilir. 20 saniye içinde veya tanımlanmış olan checktimeout süresinden sonra düğümün ağırlığı 0 yapılacak ve sonraki bağlantılar bu düğüme gönderilmeyecektir. Apache web sunucu tekrar başlatıldığında ağırlık tekrar 1 olacaktır.

ldirectord'nin Heartbeat Konfigürasyonuna Eklenmesi: Heartbeat programı kurulduktan sonra `/etc/ha.d/haresources` dosyasına aşağıdaki satırlar eklenir.

```
heartbeat.sun-cluster.com 209.100.100.3 ldirectord::ldirectord-
209.100.100.3.conf
```

heartbeat.sun-cluster.com Birincil Yöneticinin host adıdır. haresources Heartbeat'e heartbeat.sun-cluster.com isimli makinenin 209.100.100.3 VIP adresi için ldirectord kaynağına sahip olduğunu bildirir. ldirectord kaynağı Heartbeat tarafından başlatılırken `/etc/ha.d/conf/ldirectord-209.100.100.3.conf` konfigürasyon dosyasını kullanır.

Sonra ldirectord aşağıdaki komutla durdurulur.

```
#/etc/ha.d/resource.d/ldirectord ldirectord-209.100.100.3.conf
stop
```

ldirectord'nin çalışmadığından emin olmak için aşağıdaki komut kullanılabilir. Eğer ekrana bir mesaj gelmezse ldirectord çalışmamaktadır.

```
#ps -elf | grep ldirectord
```

ldirectord'nin normal boot scripti olarak başlatılmadığından emin olmak için aşağıdaki komut kullanılabilir.

```
#chkconfig --del ldirectord
```

Daha sonra Heartbeat aşağıdaki komutla durdurulur.

```
#/etc/rc.d/init.d/heartbeat stop
```

Heartbeat durdurulduktan sonra ipvsadm tablosu aşağıdaki komutla temizlenir.

```
#ipvsadm -C
```

Son olarak Heartbeat daemonu aşağıdaki komutlarla çalıştırılabilir ve daemonun çıktıları görülebilir.

```
#/etc/rc.d/init.d/heartbeat start
#tail –f /var/log/messages
```

Aynı işlemler Yedek LVS Yöneticide de tekrarlanır.

IPVS Tablosunun Birincil Yöneticiden Yedek Yöneticiye Geçişi: Birincil Yöneticide sorun meydana geldiğinde ldirectord Yedek Yöneticide bulunan IPVS tablosunu yeniden oluşturur. Bunu yaparken ldirectord konfigürasyon dosyasında bulunan ipvsadm konfigürasyon kurallarını kullanır. Buna rağmen Birincil Yöneticiden Yedek Yöneticiye geçiş sırasında Yedek Yöneticideki ldirectord aktif istemci bağlantılarını tekrar oluşturmaz. Bu nedenle küme düğümlerine doğru olan tüm istemci bağlantıları kaybolur.

Aşırı yüklü bir kümede istemciler sürekli olarak gelip gittiği için Birincil Yöneticide bulunan bağlantı izleme kayıtları değiştiklerinde Yedek Yöneticiye gönderilmelidir. LVS programlamada geliştirilen multicast tekniği sayesinde bağlantı izleme tablosu Yedek Yöneticiye çoğaltılır. Bu yöntem "server sync state daemon" tekniği olarak adlandırılır. Bu yöntemi devreye almak için Birincil Yönetici de aşağıdaki komut kullanılmalıdır.

```
/sbin/ipvsadm --start-daemon master
```

İkincil Yöneticide de aşağıdaki komut kullanılmalıdır.

```
/sbin/ipvsadm --start-daemon backup
```

Bu komutlar init scriptlere eklenebilir. Böylece sistemler boot ettiğinde daemonlar otomatik olarak başlayacaktır.

Birincil ve Yedek Yöneticiler birbirleri ile 224.0.0.81 multicast adresi üzerinden multicast paketleri ile konuşurlar. Birincil Yönetici bağlantı izleme kayıtlarında olan değişiklikleri anons eder. Yedek Yönetici bu değişiklikleri sürekli olarak dinler ve bunları bağlantı izleme tablosuna girer. *ifconfig* komutu kullanılarak düğümlerin multicast desteği olup olmadığı kontrol edilmelidir. Eğer multicast desteği yoksa çekirdek tekrar derlenmeli ve multicast desteği sağlanmalıdır.

Sync state daemonu durdurmak için aşağıdaki komut kullanılmalıdır.

```
ipvsadm --stop-daemon
```

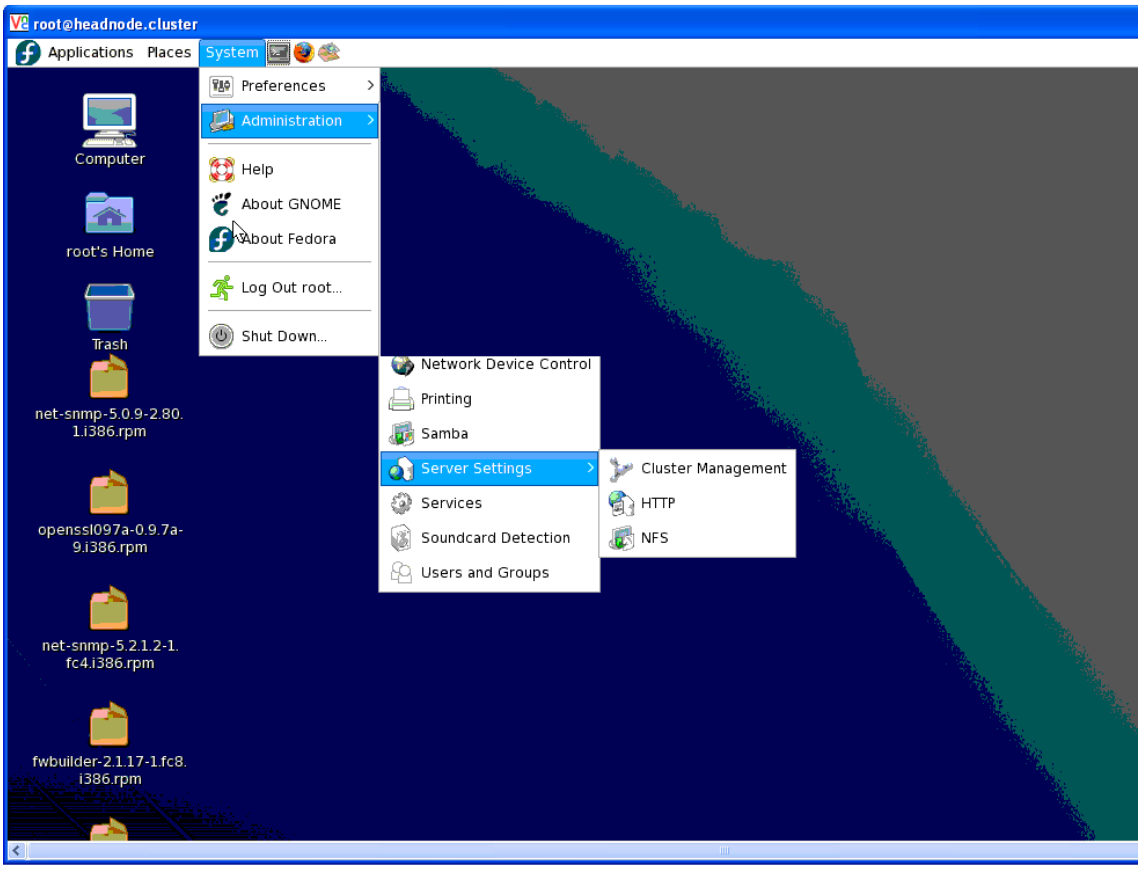
4.11.4. Apache web sunucu konfigürasyonu

Linux sürümlerinin hepsinde bulunan KDE ve GNOME pencere yöneticileri kullanılarak Apache web sunucu konfigürasyonu görsel olarak yapılabilir.

Fedora altında Apache web sunucu konfigürasyonu görsel olarak Panel / System / Administration / Server Settings / HTTP kısayolu kullanılarak yapılabilir.

Konfigürasyon yapılırken bakılması gereken ayarlar şunlardır;

- ServerName
- DocumentRoot
- BindAddress (Apache 2.0 öncesi sürümlerde)
- Port (Apache 2.0 öncesi sürümler)
- Listen (BindAddress ve Port yerine)



Şekil 4.37. Görsel APACHE konfigürasyonu

ServerName: `/etc/httpd/conf/httpd.conf` Apache konfigürasyon dosyasında bulunan ServerName satırı sunucu ismini belirler.

`#ServerName <yourserver>` satırı kaldırılarak aşağıdaki satır eklenir.

```
ServerName www.domainadi.com
```

`www.domainadi.com` satırı yerine kümenin DNS kaydı girilir.

DocumentRoot: DocumentRoot dizini HTML dosyalarının bulunacağı varsayılan dizini belirtir. Aşağıda HTML dosyaları için varsayılan dizin bulunmaktadır.

```
DocumentRoot "/www/htdocs"
```

httpd.conf dosyasında DocumentRoot ayarını belirten satır aşağıdaki gibidir.

```
<Directory "/www/htdocs">
```

Listen: Listen direktifi Apache 2.0 öncesi sürümlerde bulunan BindAddress ve Port direktiflerinin yerini almıştır. Apache web sunucunun dinlemesi gereken IP adresi ve port numarasını belirtir.

```
Listen 193.255.141.39:80
Listen 193.255.141.39:443
```

Eğer sistemde bulunan tüm IP adreslerine gelen istekler dinlenecekse Listen direktifinde IP adresleri belirtilmez.

```
Listen 80
Listen 443
```

Aşağıdaki komut kullanılarak Apache web sunucunun hangi portları dinlediği kontrol edilebilir.

```
#netstat –apn | more
```

Apache Virtual Host Konfigürasyonu: Eğer düğümler sadece tek bir DNS kaydı üzerinden hizmet sunacaksa httpd.conf dosyasında değişiklik yapmaya gerek yoktur. Ancak bir sunucu farklı host adreslerinde farklı web sayfaları yayınlayacaksa virtual host konfigürasyonu yapılmalıdır [4].

Apache IP Tabanlı Virtual Host Konfigürasyonu: Küme düğümlerinde çalışan Apache web sunucular Apache konfigürasyon dosyasındaki IP tabanlı virtual host direktifleri kullanılarak tek bir düğümde birden fazla IP tabanlı web host olarak çalışabilirler [4].

```
<VirtualHost Küme_sanal_IP_adresi:80 Düğüm_IP_adresi:80>
    #
    #
      ServerName www.domainadi.com
      ServerAdmin administrator@domainadi.com
    DocumentRoot /www/htdocs
    ErrorLog /var/log/httpd/error_log
    TransferLog /var/log/httpd/access_log
</VirtualHost>
```

Eğer birden fazla düğüm hizmet verecekse dosya aşağıdaki şekilde düzenlenmelidir.

```
<VirtualHost Küme_sanal_IP_adresi:80 Düğüm1_IP_adresi:80
Düğüm2_IP_adresi:80>
    #
```

```
    #
    ServerName www.domainadi.com
    ServerAdmin administrator@domainadi.com
    DocumentRoot /www/htdocs
    ErrorLog /var/log/httpd/error_log
    TransferLog /var/log/httpd/access_log
</VirtualHost>
```

Virtual Host kabının kullanım formatı aşağıdaki gibidir.

```
<VirtualHost   VIP:Port   Virtual RIP1-1:Port   Virtual RIP2-
1:Port  ... >
    Diğer seçenekler
</VirtualHost>
```

Apache İsim Tabanlı Virtual Host Konfigürasyonu: Tüm web tarayıcılar host adı kullanılarak HTTP isteği gönderebilirler. Böylece IP adresleri daha az israf edilmiş ve sunucuda farklı host adları ile farklı sayfalar yayınlanmış olur.

Apache isim tabanlı virtual hostlar için http isteğindeki IP adresini ile Virtual Host kabı ile eşleştirir ve ServerName direktifine bakarak ilgili Virtual Host kabının doğru kap olup olmadığına karar verir.

Eğer birden fazla domain adında yayın yapılacaksa httpd.conf dosyası aşağıdaki gibi olmalıdır.

```
# Enable Name-based virtual hosts on all IP addresses.
NameVirtualHost *

# First name-based virtual web: www1.domainadi.com
<VirtualHost *>
    ServerAdmin administrator@domainadi.com
    UseCanonicalName off
    DocumentRoot /www/ww1/htdocs
    ServerName www1.domainadi.com
    ErrorLog /var/log/httpd/error_log
    TransferLog /var/log/httpd/access_log
</VirtualHost>

# Second name-based virtual web: www2.domainadi.com
<VirtualHost *>
    ServerAdmin administrator@domainadi.com
    UseCanonicalName off
    DocumentRoot /www/www2/htdocs
    ServerName www2.domainadi.com
    ErrorLog /var/log/httpd/error_log
```

```
    TransferLog /var/log/httpd/access_log
</VirtualHost>
```

SSL protokolü kullanılıyorsa IP tabanlı virtual host kullanma olanağı yoktur. Şifrelenmiş SSL iletişimi http bağlantısından önce kurulmalıdır. Bu problemi çözmenin bir yolu her bir virtual host kabında http portundan gelen istekler için bütün virtual hostlar tarafından paylaşılacak IP adresi tanımlamaktır.

```
# Enable Name-based virtual hosts for one IP address
NameVirtualHost Küme_sanal_IP_adresi:80

# First name-based virtual web: www1.domainadi.com
<VirtualHost Küme_sanal_IP_adresi:80>
    ServerAdmin administrator@domainadi.com
    UseCanonicalName off
    DocumentRoot /www/ww1/htdocs
    ServerName www1.domainadi.com
    ErrorLog /var/log/httpd/error_log
    TransferLog /var/log/httpd/access_log
</VirtualHost>

# Second name-based virtual web: www2.domainadi.com
<VirtualHost Küme_sanal_IP_adresi:80>
    ServerAdmin administratpr@domainadi.com
    UseCanonicalName off
    DocumentRoot /www/www2/htdocs
    ServerName www2.domainadi.com
    ErrorLog /var/log/httpd/error_log
    TransferLog /var/log/httpd/access_log
</VirtualHost>
```

Virtual Host Konfigürasyonunun Doğrulanması: Konfigürasyonun tamamlanmasından sonra yapılmış olan Apache konfigürasyonu aşağıdaki komutla doğrulanabilir.

```
#/usr/sbin/httpd -t -DDUMP_VHOSTS
```

Apache versiyonlarının çoğunda aynı işlem aşağıdaki komutla da yapılabilir.

```
/usr/sbin/httpd -S
```

IP tabanlı virtual hostlarda komutların çıktıları aşağıdaki gibidir.

```
#/usr/sbin/httpd -t -DDUMP_VHOSTS
VirtualHost configuration:
Düğüm_IP_adresi:80            www.domainadi.com
(/etc/httpd/conf/httpd.conf:1215)
Küme_sanal_IP_adresi:80       www.domainadi.com
(/etc/httpd/conf/httpd.conf:1215)
```

httpd.conf konfigürasyon dosyasında 1215 numaralı satırda VirtualHost IP adresleri bulunmaktadır.

İsim tabanlı virtual hostlarda komutun çıktısı aşağıdaki gibidir.

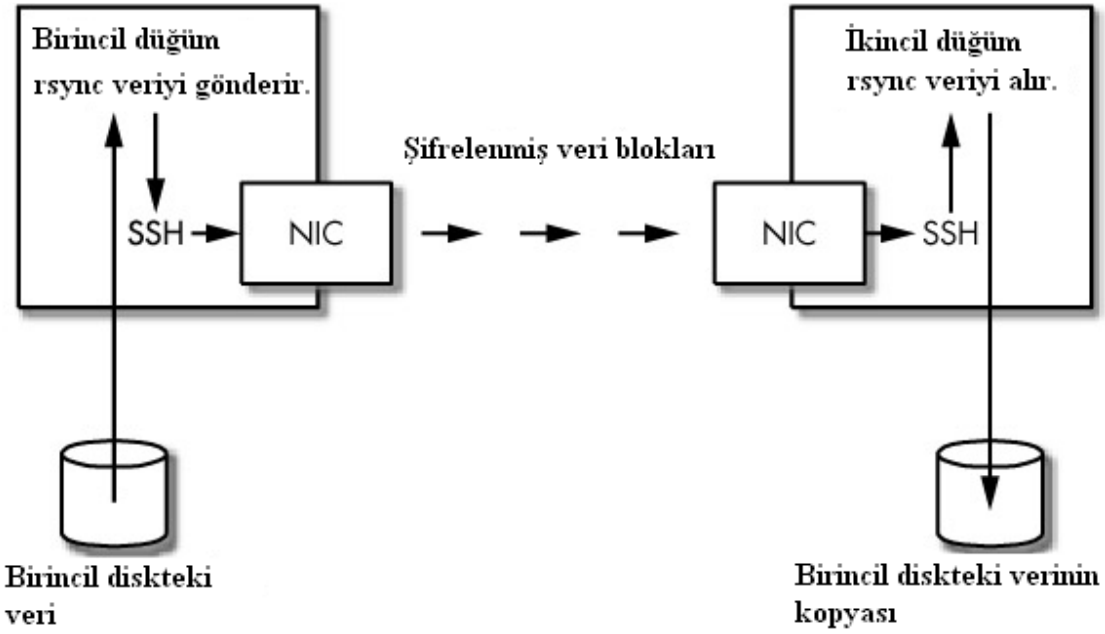
```
#/usr/sbin/httpd -t -DDUMP_VHOSTS
VirtualHost configuration:
wildcard NameVirtualHosts and _default_ servers:
*:*                     www1.domainadi.com
(/etc/httpd/conf/httpd.conf:1215)
*:*                     www2.domainadi.com
(/etc/httpd/conf/httpd.conf:1225)
```

4.11.5. Küme yapısındaki düğümlerin rsync ile eşleştirilmesi

rsync Unix sistemler için geliştirilmiş olan bir yazılımdır. Açık kaynak kodlu bir yazılım olup tüm Linux sürümlerinde kullanılabilir. Amacı farklı makineler arasında klasör ve dosya senkronizasyonu yapmaktır. Normal dosya kopyalamaya göre avantajları vardır. Bu senkronizasyon sırasında verileri sıkıştırarak transfer eder. Sadece değişen dosyalar senkronize edilebilir. rsync dosyaları karşı sisteme itebilir veya karşı sistemden çekebilir [4]. rsync yazılımının avantajları şunlardır;

- Kaynak dosyaları inceleyerek sadece değişmiş olan veri bloklarını kopyalar.
- Secure shell ile birlikte çalışarak veriyi ağa iletmeden önce şifreler.
- Ağa iletilmeden önce veriyi sıkıştırabilir.
- Kaynak sistemden silinmiş olan dosyaları otomatik olarak hedef sistemden de siler.
- WAN bağlantıları için dosya transfer hızını kısıtlayabilir.
- Aygıt dosyalarını kopyalama özelliği sayesinde sistem klonlamasını sağlayabilir.

Veriler şifrelenerek karşı makineye gönderilecekse /etc/passwd dosyasında bulunan düz metin şifreleri kullanılarak veriler şifrelenir. Bu işlem aşağıdaki şekilde görülebilir.



Şekil 4.38. rsync ve SSH

rsync GNU Genel Kamu Lisansında serbest bir yazılımdır. Rsync artalanda çalışırken TCP 873 nolu portu dinler. rsync programı Andrew Tridgell tarafında geliştirilmiş olan bir algoritma kullanır. Bu algoritma sayesinde farklı makineler arasında etkili ve hızlı bir senkronizasyon sağlanır.

rsync çoğunlukla rcp gibi davranan bir uygulamadır, ancak daha fazla seçenek içerir ve özellikle hedefteki dosyanın mevcut olması halinde dosya aktarımını çok daha hızlı gerçekleştiren rsync uzak güncelleme protokolünü kullanır. rsync uzak güncelleme protokolü rsync'in ağ bağlantısı üzerinden sadece iki dosya grubu arasındaki farkları aktarmasını mümkün kılar. Bunu bu pakete eşlik eden teknik raporda açıklanan verimli bir toplama sağlaması arama algoritmasını kullanarak yapar [4].

rsync'e özgü bazı ek özellikler bulunmaktadır. Bunlar;

- Bağların, aygıtların, sahip, grup ve izinlerin kopyalanmasını destekler.
- GNU **`tar`**'a benzer exclude ve exclude-from seçenekleri vardır.
- CVS'nin yok sayacağı dosyaların yok sayılmasını sağlayan CVS dışlama (exclude) kipi vardır.
- **`rsh`**, **`ssh`** gibi uzak şeffaf kabukları kullanabilir.
- root kullanıcısının yetkilerine ihtiyaç duymaz.

- Görünmeyen maliyetleri asgari düzeyde tutmak için dosya aktarımlarında borulama yapar.
- Hem anonim hem de kimlik doğrulamalı **`rsync`** sunucu desteği (yansılama için ideal) bulunmaktadır.

rsync sekiz farklı yöntemle kullanılabilir. Bunlar;

- Yerel dosyaları kopyalamak için. Bu çağrıda ne kaynak ne de hedef dosya yolu bir : ayracı içermez.
- Yerel makineden uzaktaki bir makineye bir uzak kabuk uygulaması (ssh veya rsh gibi) kullanarak dosyaları kopyalamak için. Bu çağrıda sadece hedef dosya yolu tek bir : ayracı içerir.
- Uzaktaki bir makineden yerel makineye bir uzak kabuk uygulaması (ssh veya rsh gibi) kullanarak dosyaları kopyalamak için. Bu çağrıda sadece kaynak dosya yolu bir : ayracı içerir.
- Uzaktaki bir rsync sunucusundan yerel makineye kopyalama yapmak için. Bu çağrıda sadece kaynak dosya yolu bir :: ayracı veya bir rsync:// URL'si içerir.
- Yerel makineden uzaktaki bir rsync sunucusuna kopyalama yapmak için. Bu çağrıda sadece hedef dosya yolu bir :: ayracı veya bir rsync:// URL'si içerir.
- Uzaktaki makinede bulunan rsync sunucusunu ve bir uzak kabuk uygulamasını kullanarak, uzaktaki makineden yerel makineye kopyalama yapmak için. Bu çağrıda kaynak dosya yolunun bir :: ayracı içermesi yanında --rsh=komut (ya da -e komut) seçeneği de kullanılır.
- Uzaktaki makinede bulunan rsync sunucusunu ve bir uzak kabuk uygulamasını kullanarak, uzaktaki makineye yerel makineden kopyalama yapmak için. Bu çağrıda hedef dosya yolunun bir :: ayracı içermesi yanında --rsh=komut (ya da -e komut) seçeneği de kullanılır.
- Uzaktaki makinede bulunan dosyaların listesini almak için. Bu işlem uzaktaki makineden kopyalama işlemindeki gibi ancak yerel hedefi belirtmeden yapılır.

Tüm durumlarda (burada listelenenler dışında kalanlar dahil) kaynak ve hedef belirtimlerinden biri daima yerel olmalıdır (yani `rsync` ile iki uzak makine arasında işlem yapılamaz).

4.11.5.1. rsync ile tek bir dosyanın kopyalanması

rsync ile tek bir dosya kopyalamak için aşağıdaki komut kullanılır. Komut 192.168.0.3 IP adresli düğüme kopyalama yapar. Hedef dosya adı seçimliktir.

```
#rsync –v –a –z –e ssh /etc/sysconfig/iptables
192.168.0.3:/etc/sysconfig/
```

Aşağıdaki komut 192.168.0.3 IP adresli düğümden tek bir dosyayı üzerinde çalışılan düğüme kopyalar.

```
#rsync –v –a –z –e ssh 192.168.0.3:/etc/sysconfig/iptables
/etc/sysconfig/iptables
```

4.11.5.2. rsync komutunun yavaş WAN bağlantılarında kullanılması

Eğer senkronize edilecek düğümler birbirine yavaş bir WAN bağlantısı üzerinden bağlıysa rsync komutunun gönderme hızı sınırlandırılabilir. Örneğin rsync komutunun gönderme hızının saniyede 8 KB ile sınırlandırılması için aşağıdaki komut kullanılabilir.

```
rsync –v –a –z –e ssh --delete --bwlimit=8 /www/ 10.1.1.2:/www
```

4.11.5.3. rsync ile zamanlanmış anlık durum görüntüsünün alınması

Birincil düğümün belirli aralıklarla rsync işlemini yapması sağlanabilir. Cron daemonu kullanılarak zamanlama yapılabilir. Örneğin /www dizinindeki verinin

12:30,13:30 şeklinde her saat diğer düğüme senkronize edilmesi için gereken ayarlar aşağıdaki gibi yapılabilir.

Önce crontab dosyası bir editörde açılır. Sonra aşağıdak satırlar dosyaya eklenir.

```
# This is a job to synchronize our files to the backup server.
30 * * * * rsync –v –a –z –e ssh --delete /www/ 10.1.1.2:/www
```

Eğer rsync işleminde hata olursa bunun kullanıcıya e-mail yoluyla bildirilmesi isteniyorsa rsync işlemi aşağıdaki şekilde ayarlanır.

```
rsync –v –a –z –e ssh --delete /www/ 10.1.1.2:/www || echo
"rsync failed" | mail admins@yourdomain.com
```

Yüksek erişilebilir küme yapısında rsync işlemi aşağıdaki gibi yapılabilir.

Sunucu tarafında yapılması ilk gereken /etc/rsyncd.conf dosyasına aşağıdaki satırları eklemektir.

[ha]
path = /var/www/htdocs
comment = Cluster

Daha sonra sunucuda rsync daemon modunda çalıştırılır ve bu komut gerekli olan sistem başlangıç scriptlerine eklenir.

#echo "rsync --daemon" >> /etc/rc.local

İstemci tarafında yapılması ilk yapılması gereken aşağıdaki shell script in yazılmasıdır.

----- /etc/rsync.sh ----------

```
#!/bin/sh

LOGFILE=/var/log/rsync.log
LOCALDIR=/var/www/htdocs
REMOTEDIR=rsync://www1.cluster.org/ha
rsync -avzH --stats --delete \
```

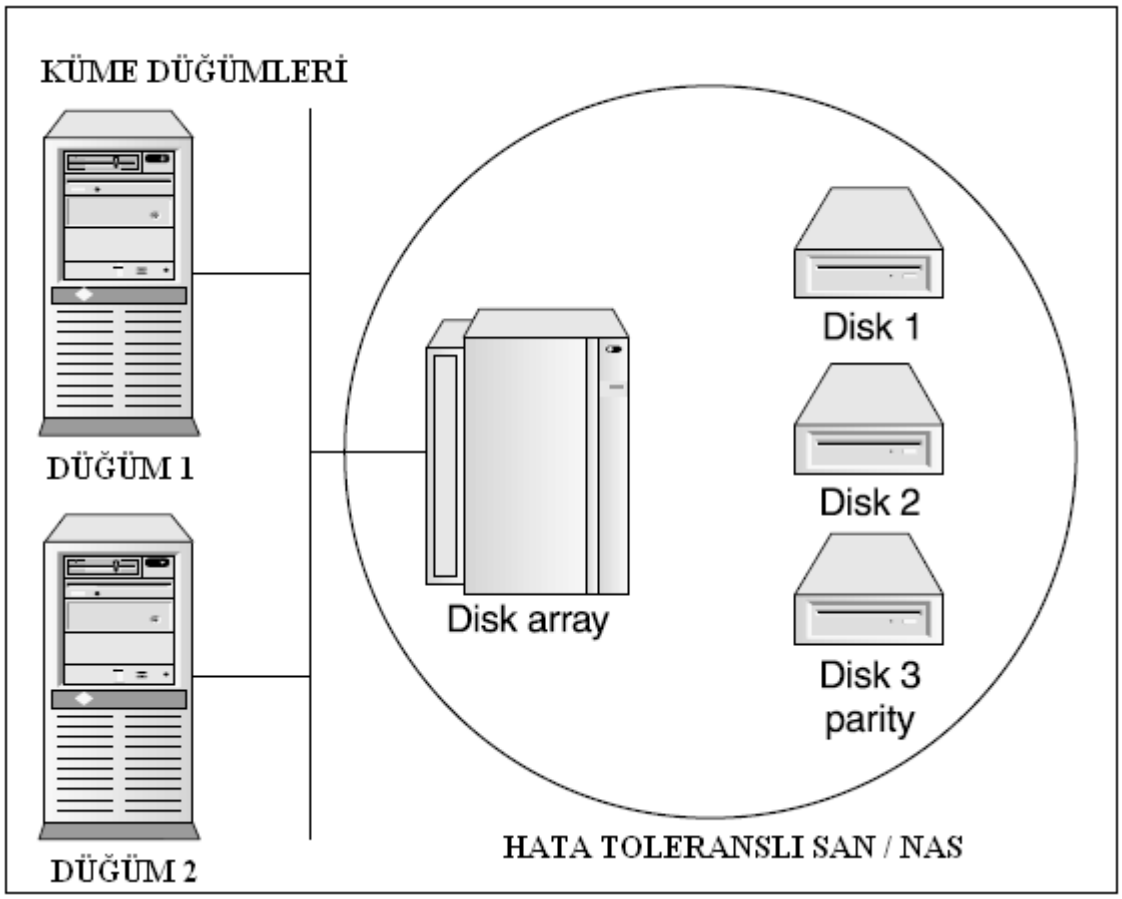
$REMOTEDIR $LOCALDIR >> $LOGFILE
---------/etc/rsync.sh----------

Daha sonra bu script cron a eklenir. Böylece her 5 dakikada bir www1 deki değişiklikler (yeni dosyalar ya da silinen dosyalar) istemciye geçirilir.

rsync programının kümeleme uygulamasında kullanılmasındaki amaç küme yapısını oluşturan düğümler arasında senkronizasyonu sağlamasıdır. Kullandığımız küme yapısında hata toleransına konu olan uygulama Apache üzerinde koşan web tabanlı bir uygulama olduğu için rsync uygulamaya ait olan klasörlerin senkronizasyonu için kullanılacaktır.

4.11.6. MYSQL veritabanı sunucusu kullanılarak düğümlerin ortak bir veritabanı kullanmasının sağlanması

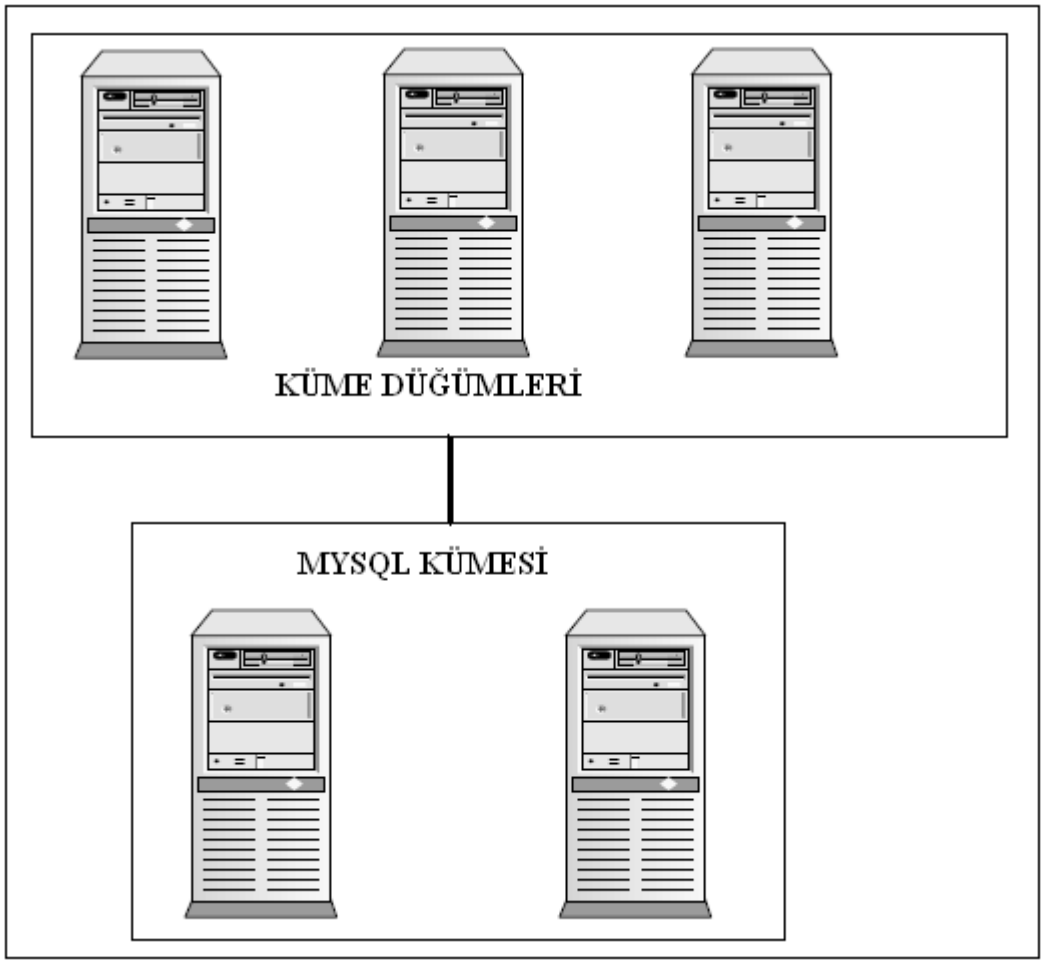
Düğümlerde koşacak uygulama seçimi küme yapısını direkt olarak etkileyecektir. Örneğin düğümlerde web tabanlı bir uygulama çalışacak ve bu uygulama da bir veritabanı kullanacaksa düğümler arasında veri tutarlılığı olması için düğümlerin kullanacağı veritabanları da ortak olmalıdır. Aksi taktirde farklı düğümlerde farklı veritabanları olacak ve veriler arasında tutarsızlık olacaktır. Bu noktada kullanılabilecek iki çözüm vardır. Birinci çözüm SAN veya NAS gibi bileşenler kullanarak ortak verileri bunlar üzerinde saklamaktır. İkinci çözüm ise küme yapısında çalışan MYSQL sunucuları kullanmaktır.



Şekil 4.39. SAN / NAS İle Küme Yapısında Ortak Verilerin Saklanması

Tez kapsamında kurulan hata toleranslı yük dengelemeli küme yapısında Fedora sunucular üzerinde Apache web sunucular çalışmaktadır. Apache üzerinde PHP ile geliştirilmiş ve MYSQL veritabanı kullanan uygulama çalışmaktadır. Düğümlerde Apache sunucular üzerinde yüklü olan PHP uygulamaları rsync ile eşleştirilmektedir. Birincil düğümde eğer bir değişiklik olursa bu değişiklik ikinci düğümede rsync ile yansıtılacaktır.

Düğümlerde çalışan PHP uygulamalarının farklı isteklere aynı verileri sunabilmesi için ortak veritabanı kullanılması gerekmektedir. Bu nedenle ayrı bir MYSQL sunucu kümesi kurulmuş ve veritabanları bu uzak sunucu kümesinde tutulmuştur. Böylece veri tutarlılığı da sağlanmıştır.



Şekil 4.40. MYSQL Sunucu Kümesi İle Ortak Güvenli Veritabanı Erişimi

BÖLÜM 5

5. HATA TOLERANSLI UYGULAMALAR

Günümüzde küreselleşmeyle birlikte acımasız hale gelen rekabet dolayısıyla şirketler sundukları servisleri iyileştirmek için büyük bir gayret içindedir. Devamlılık sunulan servislerde en büyük gereksinimdir. Örneğin bir hosting firmasını ele alırsak firma sunucularında hata toleransını sağlamak zorundadır. Sunucu arızası dolayısıyla sitesine erişim kesilen bir firma hosting firmasıyla olan anlaşmasını sona erdirebilir. Özellikle yüksek hit alan sitelerde bu problem daha da büyük olacaktır. Bir alışveriş sitesini ele alırsak sunucu arızası dolayısıyla hizmetlerde kesinti meydana gelirse müşteriler alışverişini başka alışveriş sitelerinden yapabilir ve firma büyük zararlara uğrayabilir.

Hata toleransı sağlamak için farklı çözümler bulunmaktadır. Çözümler genelde donanımsal ve yazılımsal çözümler olarak iki grupta toplanabilir. Donanımsal çözümler yazılımsal çözümlere göre daha pahalı ama daha kararlı çözümler olarak değerlendirilebilir. Donanımsal ürünlere örnek olarak Cisco firmasının Content Services Switch ve Cisco ACE Application Control Engine Module ürünleri verilebilir. Hata toleransı için kullanılacak çözümler hangi uygulama için hata toleransı uygulanacağına da bağlıdır. Örneğin bir web sunucu havuzunda kullanılacak çözüm veritabanı için kullanılacak çözümden farklı olacaktır.

Bazı uygulamalar dahili hata toleransı desteği ile gelmektedir. Bu uygulamalara örnek olarak Oracle veritabanı sunucusu verilebilir. Oracle RAC (Real Application Cluster) Oracle veritabanı sunucularını kümelemeyi sağlar. Böylece hem yüksek performans hem de hata toleransı sağlanmış olur.

Tez kapsamında incelenen Linux Virtual Server ise belirli bir uygulama türü için geliştirilmiş bir çözüm değildir. Farklı uygulamalardaki farklı senaryolara göre kolaylıkla uygulanabilir. Örneğin bir LVS sunucu kümesi hem web uygulamalarında hem veritabanı uygulamalarında hemde ortak dosya sistemi uygulamalarında kullanılabilir. Yapılması gereken sadece kurulum sonrası uygulama türüne göre yapılması gereken ayarlamalardır. Linux Virtual Server sadece hata toleransı sağlamaz. Bunun yanında yük dengeleme de sağlar. Heartbeat paketi kurulursa tek bir yönetici düğümden kaynaklanan tek hata noktası probleminde önüne geçilmiş olur. Birden fazla yönetici yedekli yapıda çalışarak gelen istekleri düğümlere iletirler. Yüksek Erişilebilir Küme Yapısı çözümlerinin en yaygın kullanılanlarından birisi Linux Virtual Server yazılım ailesidir.

Hata toleransı çözümüne karar verilmeden önce yapılması gereken mevcut durumun analizini yapmak, ihtiyaçları belirlemek, bilgi işlem sistemini göz önüne almak ve bütçelemeyi yapmaktır. İhtiyaçlar belirlenirken sadece bugünün ihtiyaçlarına bakılmamalı gelecekte olabilecek ihtiyaçlarda göz önüne alınmalıdır. Ayrıca alınacak ürün firmanın mevcut bilgi işlem sistemine de uygun bir ürün olmalıdır. Örneğin sadece iki sunucuyu destekleyen bir hata toleransı çözümü ikiden fazla sunucu olduğunda kullanılamaz hale gelecektir. Dolayısıyla ürün seçimi detaylı bir değerlendirme aşamasından sonra olmalıdır.

5.1. Linux Küme Yapılarının Kullanılabileceği Uygulamalar

Linux küme yapıları farklı uygulamalarda farklı ihtiyaçları yerine getirmek için başarıyla kullanılmaktadır. Özellikle Internetin hayatın her alanına girmesiyle birlikte web sunucu kümeleri yaygın olarak kullanılmaktadır. Linux kümelerinin kullanım alanları şunlardır;

- Veritabanı sunucuları
- Web sunucuları

- FTP sunucular
- ERP uygulamaları
- Yük Dengeleyici sunucular
- E-Mail sunucular
- Firewall uygulamaları
- Dosya sunucular
- DNS sunucular
- DHCP sunucular
- Proxy Caching sunucular
- Ağ Geçidi uygulamaları
- Çoklu Ortam Yayın sunucuları (Video, ses, Video on Demand, IPTV)

Veritabanı Sunucuları: Veritabanı sunucuları günümüzde bütün istemci-sunucu tabanlı uygulamalar tarafından yaygın olarak kullanılmaktadır. Yaygın olarak kullanılan veritabanı sunucularına örnek olarak Microsoft SQL Server, Oracle ve MYSQL verilebilir. Birçok veritabanı sunucusu ilave bileşenlerle kümelemeyi desteklemektedir. Örneğin Oracle RAC (Real Application Cluster) Oracle veritabanı sunucularını kümelemek için kullanılır.

Veritabanı sunucusu kümelerinin avantajları şunlardır;

- Servis dışı olma süresi sıfıra yakındır.
- Operatör müdahalesi olmadan hata yapan sunucuyu devreden çıkararak yedek veritabanı sunucusunu devreye alır.
- Sistem güncellemelerinin ve yenilemelerinin sistemin devre dışı bırakılmadan yapılabilmesini sağlar. Böylece veritabanı hizmetlerinde kesinti olmaz.
- Sunucu, ağ ve veritabanı bakımından kaynaklanan servis dışı olma süresini azaltır.
- Kümeleme sunucuların tekrar isimlendirilmesini gerektirmez. Hata meydana geldiğinde yedek veritabanı sunucusuna geçiş işleminden kullanıların haberi olmaz.
- Yedek veritabanı sunucusuna geçiş çok hızlıdır.

- Birincil ve yedek veritabanı sunucuları aktif–aktif modunda çalışarak sistemin performansını arttırmak için kullanılabilir. Birincil veritabanı sunucu çok yoğun olduğunda uygulamaların veritabanına erişim isteklerini yedek veritabanı sunucusu yerine getirecektir.

Veritabanı sunucu kümelerinin dezavantajları şunlardır;

- Hata toleransı olmayan sunuculara göre maliyeti yüksektir.
- Kurulumu zaman almaktadır.
- Bakımı tek sunucudan daha uzun sürmektedir.
- Daha deneyimli veritabanı yöneticileri ve ağ yöneticileri tarafından yönetilmelidir.

Web Sunucular: Web sunucular hem Internet üzerinde hem de Intranet lerde yaygın olarak kullanılmaktadır. Günümüzde istemci-sunucu temelli programların yerini web tabanlı yazılımlar almıştır. Bu nedenle web sunucular şirketlerin bilgi işlem altyapılarında da önemli bir yer tutarlar. Web sunuculardan en bilinenleri Microsoft Internet Information Server, Apache ve Lotus Domino dur. Internet üzerinde yaygın olarak kullanılan Apache sunucular Linux işletim sistemlerinde çalışmaktadır. Linux işletim sistemlerinde Apache kümeleri kurularak kesintisiz, yük dengelemeli ve yüksek performanslı web servisleri sunmak mümkün olacaktır. Bu yapıya alternatif olarak Round Robin DNS çözümü kullanılabilir ancak bu çözüm kısıtlı olarak yük dengeleme sağlar ve yüksek performans sağlamaz.

FTP(File Transfer Protocol) Sunucular: FTP sunucular Internet üzerinden dosya indirirken yaygın olarak kullanılan sunuculardır. Bazı FTP sunuculara aynı anda binlerde kullanıcı bağlanabilmektedir. FTP sunucularından beklenti kesintisiz olarak yüksek hızla dosya indirmeye izin vermedir. Linux kümeleri bu kesintisiz hizmeti sağlamak için kullanılabilir.

ERP (Enterprise Resource Planning: Kurumsal Kaynak Planlaması) Uygulamaları: Kurumsal Kaynak Planlaması uygulamaları işletmelerde mal ve hizmet üretimi için gereken işgücü, makine, malzeme gibi kaynakların verimli bir şekilde kullanılmasını sağlayan bütünleşik yönetim sistemleridir. İşletmenin bütün fonksiyonlarının tümleşik olarak çalışabilmelerine olanak vererek, kritik kaynakların yönetilmesini ve analiz

edilebilmesini sağlayan uygulamalardır. ERP yazılımları organizasyon yapısı içerisinde karmaşık işlemleri kolaylaştırırken şirketlerin küresel iş yaşamanına uyumlarını sağlamaktadır. ERP yazılımları ile işletmelerde güvenilir bilgiye daha hızlı ve etkin erişim sağlanır. Kalite, etkinlik, müşteri memnuniyeti ve hız gibi kavramları ön plana çıkaran ERP yazılımları işletmelerde etkinliği arttırırken, maliyetleri de azaltmaktadır. Tüm bu nedenlerle ERP uygulamalarında kesintisiz çalışma şirketler için kritik önem taşıyan bir husustur. Tek başına çalışan bir ERP sunucusunda yaşanacak sorun şirketin tüm fonksiyonlarında aksama meydana getirecektir. Bu nedenle ERP sunucu kümeleri kullanmak şirketler için çok önemlidir. ERP sunucu kümesi ERP uygulamalarına kesintisiz olarak ulaşabilmeyi sağlayacaktır. Aynı zamanda sunucu kümeleri sayesinde yedekli yapıda yüksek performansla çalışma imkanı da olacaktır.

Yük Dengeleyici Sunucular: Yük Dengeleyici sunucular çok farklı uygulamalarda kullanılabilir. Piyasada yük dengeleme amacıyla kullanılan donanımsal çözümler bulunmaktadır. Cisco firmasının Content Services Switch ve Cisco ACE Application Control Engine Module ürünleri yük dengeleme amacıyla kullanılan çözümlerden bazılarıdır. Linux sunucular yük dengeleme işlevini yerine getirmek için kullanılabilir. Benzer çözümlere göre büyük maliyet avantajı sunan çözümlerdir.

E-mail Sunucular: E-mail uygulamaları günümüzde yaygın olarak kullanılmaktadır. İş hayatında önemli bir yeri olan e-mail uygulamalarında da kesintisiz erişim önemlidir. Yaygın olarak kullanılan Microsoft Exchange Server ve Lotus Domino Server gibi e-mail sunucuların dahili kümeleme destekleri bulunmaktadır. Linux üzerinde çalışan Postfix, Qmail ve Sendmail gibi e-mail sunucuları Linux Virtual Server ile kümelemek mümkündür.

Firewall Uygulamaları: Firewall çözümleri tüm firmalarda kullanılan uygulamalardır. Özellikle kesintisiz olarak Internete bağlı olan tüm firmalarda zorunlu olarak bulunması gereken çözümlerdir. Hosting firmaları, Internet servis sağlayıcılar ve bankalar uzun zamandır firewall çözümleri kullanmaktadır. Güvenlik çözümleri şirketlerin bilgi işlem altyapısında çok önemlidir. Piyasada yaygın olarak kullanılan firewall çözümlerinden bazıları Checkpoint, Cisco, Fortigate ve Microsoft ISA Server dır. Bu ürünlerin yanında Linux işletim sistemi üzerinde ücretsiz olarak gelen iptables yazılımı da yaygın olarak kullanılmaktadır. Firewall bir bilgi işlem altyapısında sürekli olarak çalışması

gereken bir bileşendir. Cisco gibi donanımsal firewall çözümlerinin kendi yedekli çalışma çözümleri bulunmaktadır. Linux üzerinde çalışan firewall çözümlerinde ise Linux sunucu kümeleri kullanılarak kesintisiz ve yüksek performanslı firewall çözümleri elde etmek mümkündür.

Dosya Sunucular: Bilgisayar ağları sayesinde bilgi paylaşımı yapmak mümkün olmaktadır. Örneğin bir şirketin muhasebe bölümünde çalışan tüm kullanıcılar ortak bir sunucu üzerinde çalışmakta ve birbirlerinin dosyalarına erişebilmektedirler. Eğer dosya sunucusunda bir sorun meydana gelirse çalışmalarda kesinti yaşanabilmektedir. Linux işletim sisteminde NFS sunucu kümeleri sayesinde yüksek erişilebilir ortak dosya sistemleri elde etmek mümkündür.

DNS Sunucular: DNS sunucular Internet bağlantısı olan tüm firmalarda kullanılmaktadır. DNS sunucular host adı-IP adresi dönüşümünden sorumlu olan sunuculardır. Linux işletim sistemi üzerinde çalışan ve yaygın olarak kullanılan BIND DNS sunucularla küme yapıları kurulabilir ve istemci bilgisayarlara kesintisiz DNS servisi sunulabilir.

DHCP Sunucular: DHCP sunucunun görevi şirket ağına bağlı olan bilgisayarlara otomatik olarak IP adresi, alt ağ maskesi, DNS sunucu, WINS sunucu ve ağ geçidi bilgilerini aktarmaktır. Özellikle ağa bağlı olan bilgisayar sayısı fazla ise DHCP sunucu büyük önem taşıyacaktır. DHCP sunucuda bir sorun meydana gelirse bilgisayarlar IP adresi alamayacak ve ağ erişimleri olmayacaktır. Bu durumda IP adresi ve diğer bilgilerin tek tek elle girilmesi gerekecektir. Bu nedenle DHCP sunucunun kesintisiz hizmet sunması çok önemlidir. Linux kümeleri ile kesintisiz DHCP servisi sunmak mümkündür.

Proxy Caching Uygulamaları: Proxy sunucular Internet bağlantısını istemci lehine gerçekleştiren ve istenilen sayfayı getirip istemciye teslim eden çözümlerdir. Proxy sunucular sık olarak erişilen sayfaları cache lerine kaydederek aynı sayfaları yeniden Internetten getirmezler. Hem Internet bağlantısını verimli kullanırlar hem de hız sağlarlar. Proxy sunucu aynı zamanda güvenliği de sağlar. Proxy sunuculardan en bilineni Linux işletim sistemi üzerinde çalışan Squid dir. Microsoft Proxy Windows işletim sistemlerinde Microsoft ISA Server piyasaya sunulmadan önce yaygın olarak

bulunmaktaydı. Squid genelde caching amacıyla kullanılır. Squid kümeleri ile kesintisiz olarak proxy ve caching hizmeti sunmak mümkündür.

Ağ Geçidi Uygulamaları: Ağ geçidi bir firmanın dış ağ ile olan bağlantısından sorumludur. İstemci bilgisayarlar iç ağı hedeflemeyen tüm paketleri ağ geçidine gönderirler. Firewall veya router gibi cihazlar da ağ geçidi olarak kullanılabilir. Ağ geçidi olarak ayrı sunucu kümeleri kullanılırsa farklı bağlantılar veya cihazlar arasından seçim yapmak, yük dengeleme yapmak veya hata durumunda alternatif yolu kullanmak mümkün olacaktır.

Çoklu Ortam Yayın Sunucuları: Çoklu ortam yayın sunucuları video, ses, VOD (video on demand) ve IPTV gibi uygulamalarda kullanılabilir. Özellikle ticari olarak yapılacak çoklu ortam yayınlarında kesintisiz ve yüksek performanslı hizmet çok önemlidir. Linux kümeleri kesintisiz ve yüksek performanslı yayın için gereken servisleri sunarken kullanılabilir.

BÖLÜM 6

6. TEZ KAPSAMINDAKİ KÜME UYGULAMASI

6.1. Linux Kümesini Meydana Getiren Bileşenler

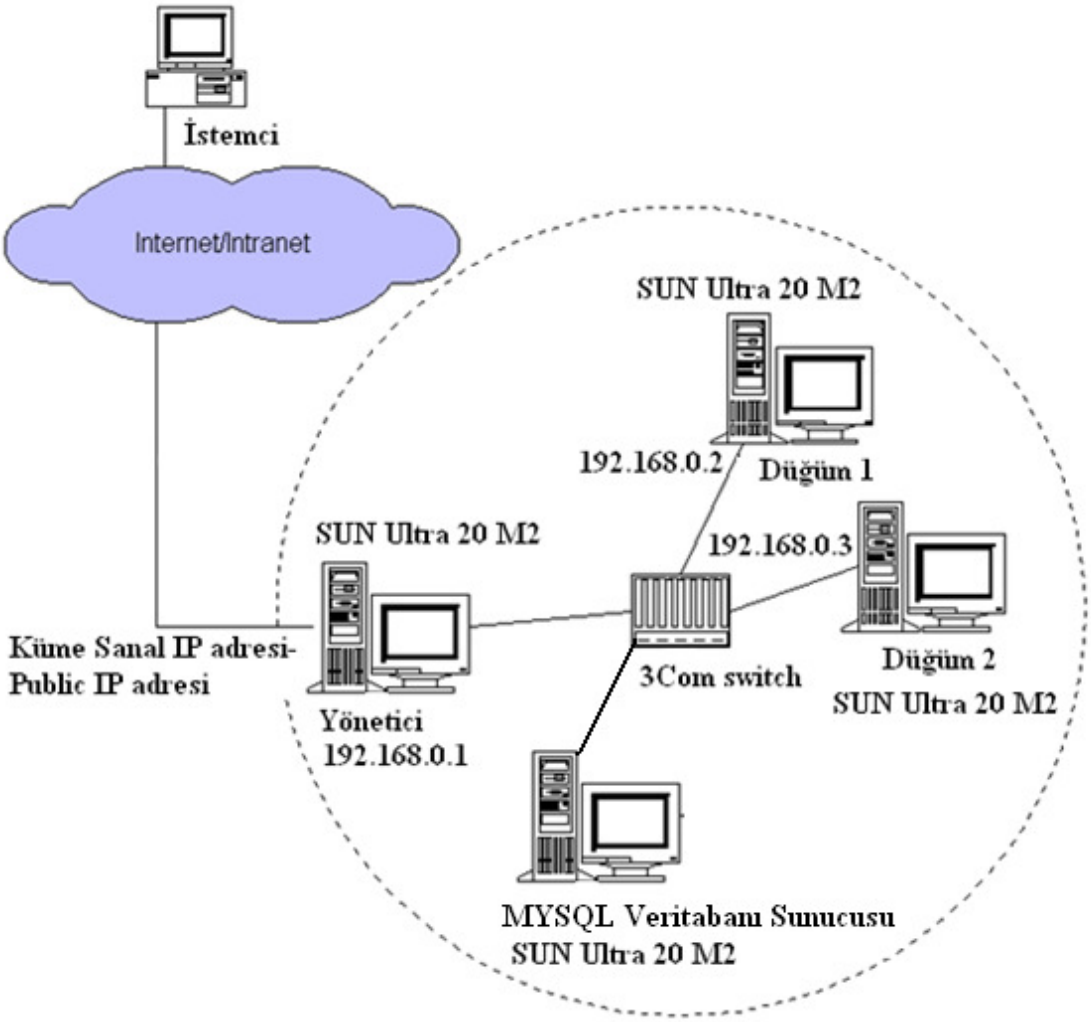
Kümeyi oluşturan düğümlerde *SUN Ultra M2* bilgisayarlar kullanıldı. *SUN Ultra M2* serisine ait olan teknik özellikler aşağıdaki tabloda görülebilir.

CPU	Dual-core AMD Opteron 1200 series CPU with 8-GBps HyperTransport
İkincil Cache Bellek	Çekirdek başına 1 MB
Ana Bellek	1 GB DDR2-667
Ağ Kartı	2 adet Gigabit Ethernet portu
Sabit Disk	160 GB SATA 7200 RPM
Optik Sürücü	DVD-ROM
Ekran Kartı	NVIDIA Quadro FX 560 128-bit 128 MB
Giriş/Çıkış	6 adet USB 2.0, 2 adet IEEE 1394a (Firewire) portu
Ses Kartı	On board ses kartı
Güç Kaynağı	100-120, 220-240 V AC; 47-63 Hz
Monitör	Sun 17-inch Sun flat-panel LCD

Tablo 6.1. Linux Kümesini Meydana Getiren Bileşenler

Küme yapısında bir Yönetici, iki adet Düğüm ve bir adet MYSQL Veritabanı Sunucusu bulunmaktadır. Küme düğümlerinin birbirine bağlantısını sağlamak için *3COM 3C16477A* 8 adet 10/100/1000 portlu switch kullanıldı.

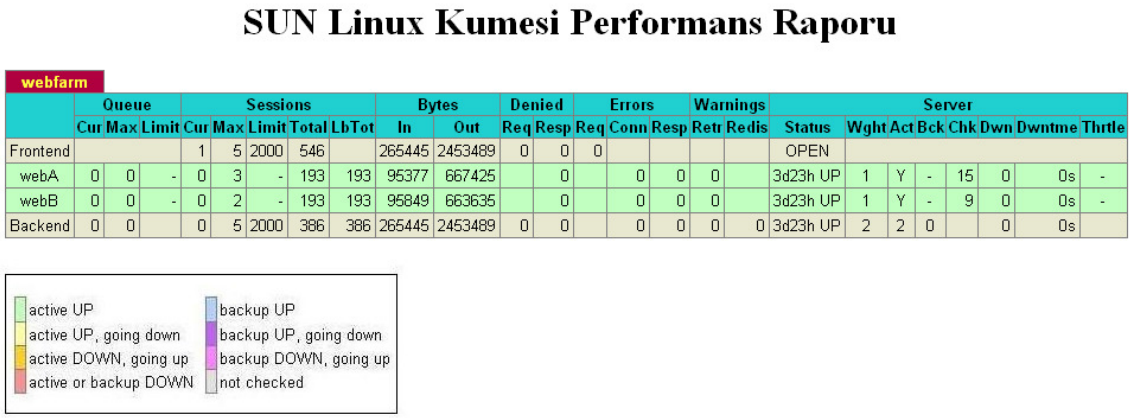
Küme yapısındaki yönetici, iki adet düğüm ve veritabanı sunucusunda Fedora 8 işletim sistemi kullanıldı.



Şekil 6.1. Tez Kapsamında Kurulan Linux Küme Yapısı

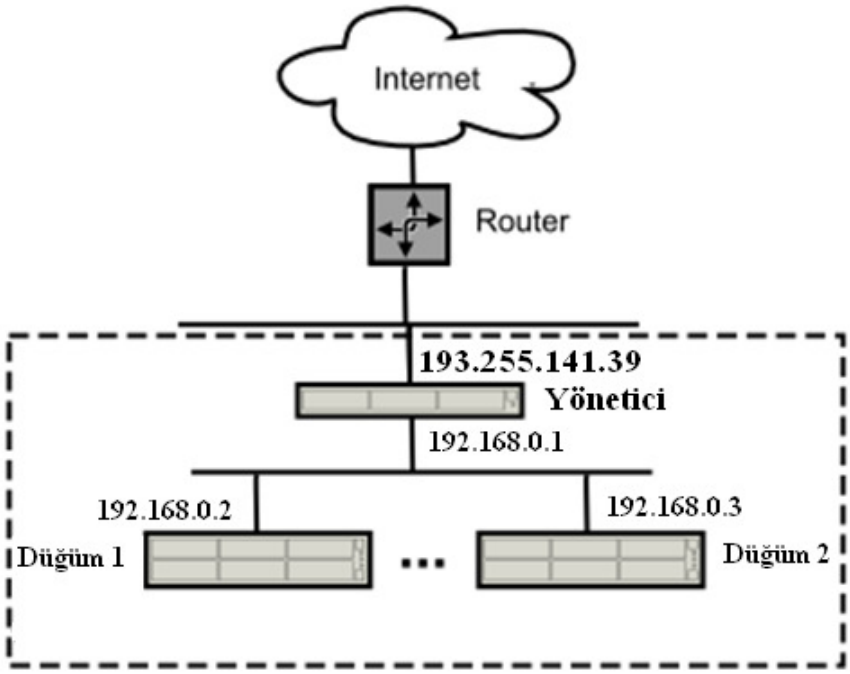
Kümeye kurulan raporlama uygulaması LVS kümesini performans değerleri için sorgulamakta ve bu değerleri aşağıdaki şekilde de görülebileceği gibi raporlamaktadır. Raporlara https://193.255.141.39/ adresinden ulaşılabilmektedir. Rapor Linux

kümesindeki Yöneticinin ve uygulamayı çalıştıran düğümlerin durumlarını göstermektedir.



Şekil 6.2. Küme Performans Raporu

6.2. Linux Hata Toleranslı Yük Dengelemeli Küme Yapısının Kurulması

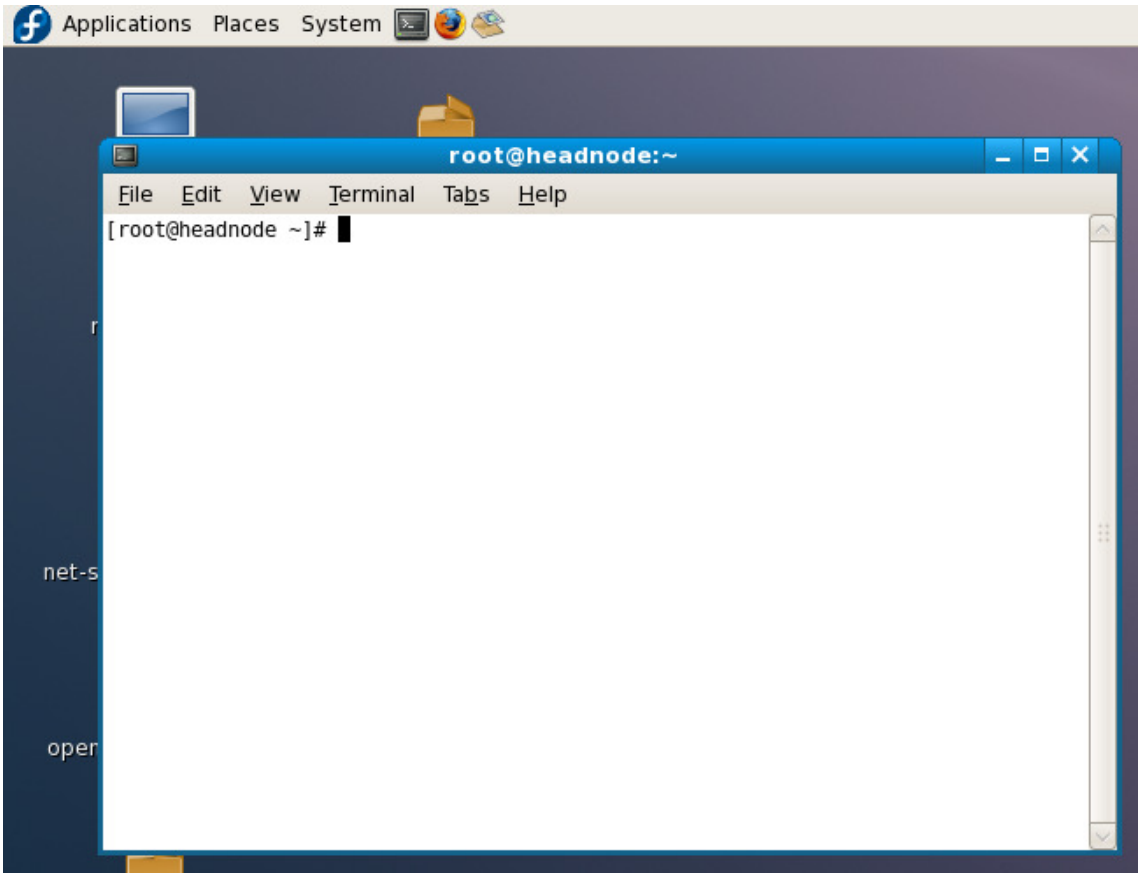


Şekil 6.3. Küme Yapısına Detaylı Bakış (Veritabanı sunucusu şekilde gösterilmemektedir.)

Küme yapısında Fedora 8 sunucular kullanılmaktadır. Bu sunucular üzerine Linux Virtual Server yapısının gerçekleştirilmesi için ipvsadm ve heartbeat paketleri kurulmuştur. Linux Virtual Server kümeleri için gerekli olan tüm yazılımlar http://www.linuxvirtualserver.org sitesinde bulunmaktadır. Farklı Linux sürümleri için programların farklı versiyonları bulunmaktadır.

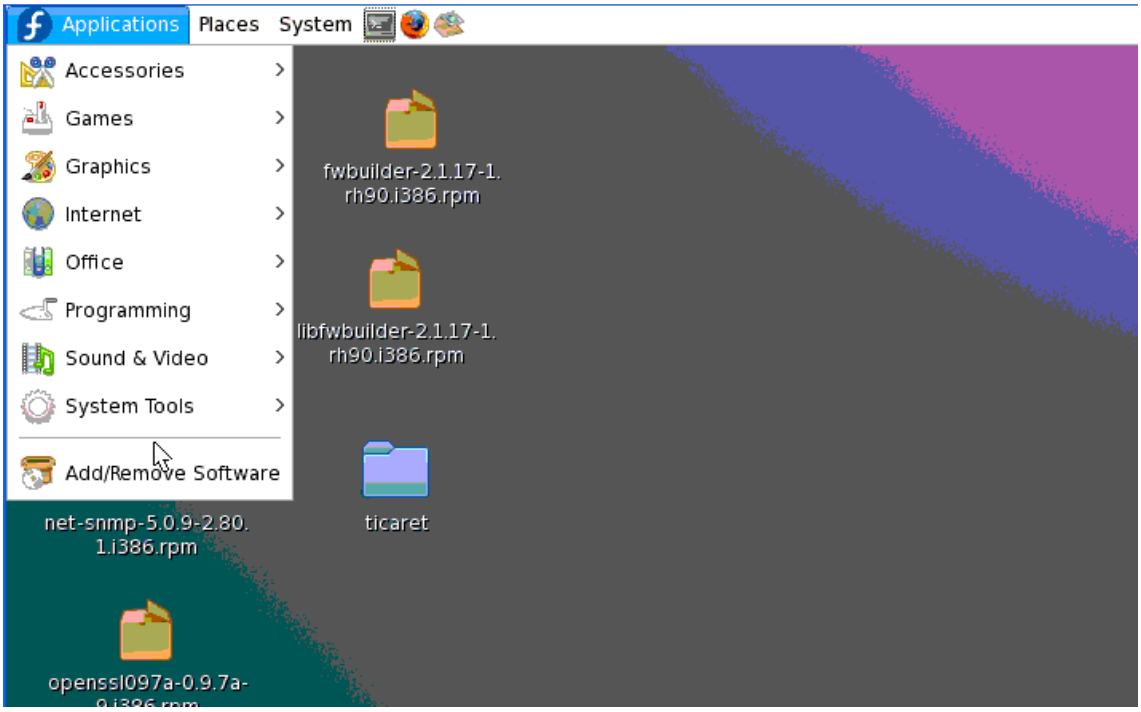
Fedora 8 işletim sisteminde programların kurulumu için aşağıdaki komutlar kullanılabilir. Birçok durumda bağımlılıkları tespit edip gerekli olan programları otomatik olarak kurduğu için pencere yöneticisinde bulunan Add/Remove Software programı kurulum için daha uygun olacaktır. Aşağıdaki komutlar Terminal ekranında kullanılmalıdır.

```
$ sudo yum install ipvsadm
$ sudo yum install ldirectord
$ sudo yum install heartbeat
```

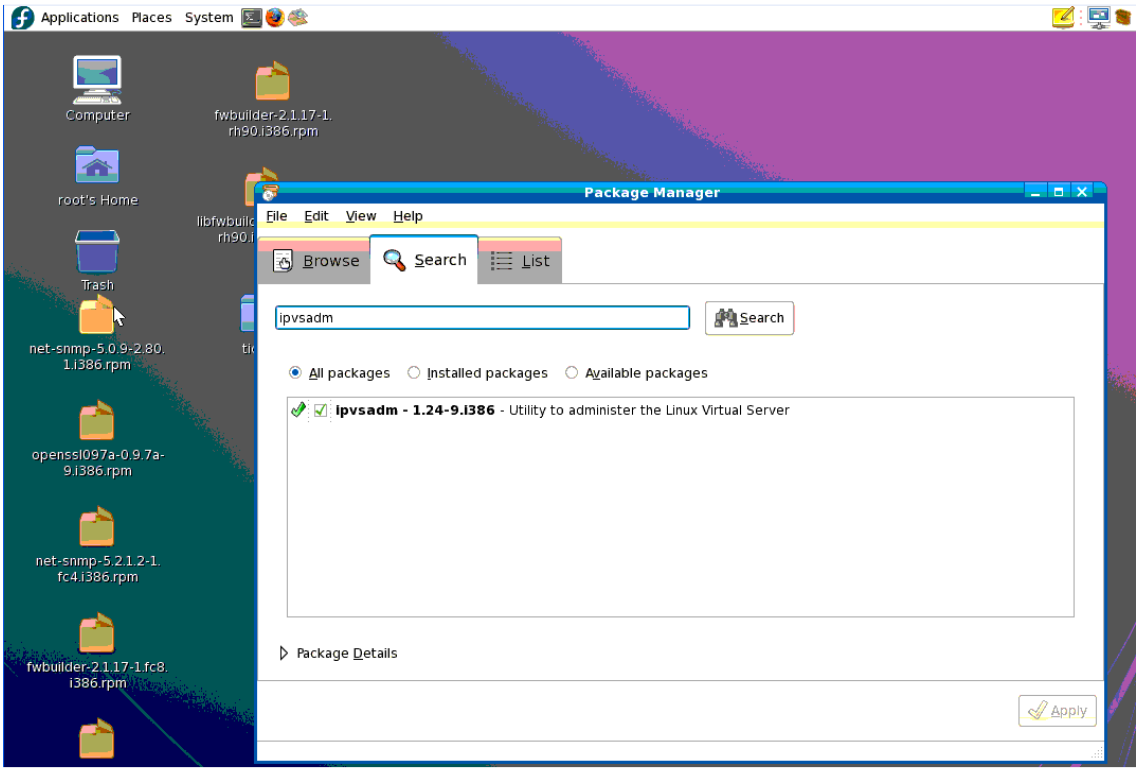


Şekil 6.4. Terminal Ekranı

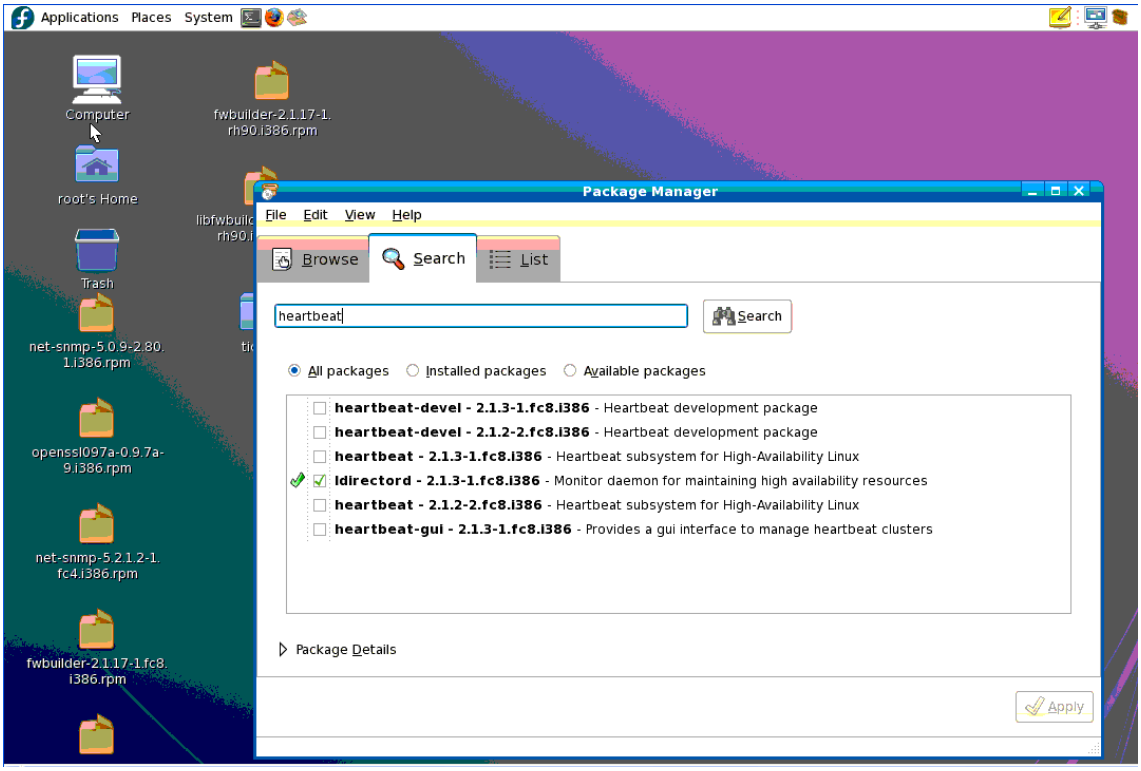
Fedora 8 işletim sisteminde gelen GNOME pencere yöneticisinde Panel altında Applications menüsünde bulunan Add/Remove Software programı kullanılarak paketlerin kurulumu yapılır.



Şekil 6.5. Add/Remove Software



Şekil 6.6. ipvsadm programının kurulumu



Şekil 6.7. heartbeat ve ldirectord programlarının kurulumu

Programların kurulumu tamamlandıktan sonra programların kullandığı konfigürasyon dosyaları oluşturulmalıdır.

6.2.1. Yöneticide Yapılması Gereken İşlemler

6.2.1.1. Çekirdeğin tekrar oluşturulması

Küme yapısında kullanılan Fedora 8 işletim sistemi 2.6 sürümü çekirdeğe sahip olduğu için aşağıdak işlemlerin yapılmasına gerek yoktur. Ancak eski çekirdek sürümlerinde aşağıdaki işlemlerin yapılması zorunludur.

2.4.28 sürümünden önceki çekirdekleri kullanan Linux systemlerde çekirdekte virtual server desteği gelmemektedir. Bu sürümlerde yamalar yüklenerek çekirdek tekrar oluşturulmalıdır. 2.4.28 sürümünden sonraki 2.4 sürümlerinde ve 2.6 sürümlerinde virtual server desteği için bir işlem yapılmasına gerek yoktur.

Gerekli olan yamalar http://www.linuxvirtualserver.org sitesinden indirilmelidir. Farklı çekirdek sürümleri için farklı yamalar bulunmaktadır. Aşağıdaki işlemler 2.4.x serisi çekirdeklerde uygulanması gereken `linux-2.4.21-ipvs-1.0.10.patch.gz` yamasının uygulanışını göstermektedir.

Önce root olarak login olunmalı ve aşağıdaki komutlar uygulanmalıdır. Bu komutlar yamayı çekirdeğe uygulayacaktır. root kullanıcısına geçiş yapmak için *su* komutu kullanılabilir.

```
#cd /usr/src/linux*
#gunzip ../linux-2.4.21-ipvs-1.0.10.patch.gz
#patch -p1 ../linux-2.4.21-ipvs-1.0.10.patch
```

Daha sonra çekirdek derlenmelidir. Aşağıdaki komutlar /usr/src/linux* dizininde kullanılmalıdır.

```
#make mrproper
#make oldconfig
#make menuconfig
```

Aşağıdaki seçenekler ekrana gelecektir. Aşağıdaki seçimler yapılır.

```
virtual server support (EXPERIMENTAL)
[*] IP virtual server debugging
(16) IPVS connection table size (the Nth power of 2)
--- IPVS scheduler
<M> round-robin scheduling
<M> weighted round-robin scheduling
<M> least-connection scheduling scheduling
<M> weighted least-connection scheduling
<M> locality-based least-connection scheduling
<M> locality-based least-connection with replication scheduling
<M> destination hashing scheduling
<M> source hashing scheduling
<M> shortest expected delay scheduling
<M> never queue scheduling
--- IPVS application helper
<M> FTP protocol helper
```

Çekirdek ayarları kaydedilerek menuconfig ekranından çıkış yapılır ve aşağıdaki komut dizisi çalıştırılır.

```
#make dep && make clean && make bzImage && make modules && make
modules_install
```

Yukarıdaki işlem /usr/src/linux*/arch/i386/boot dizininde sıkıştırılmış yeni bir çekirdek imajı (bzImage) yaratacak ve bu çekirdek için modülleri oluşturup kuracaktır. Daha sonra bu çekirdek imajı /boot dizinine kopyalanmalıdır. Son olarak kullanılan boot yükleyicisine göre /etc/grub.conf veya /etc/lilo.conf dosyası düzenlenerek sistemin yeni çekirdekten boot etmesi sağlanmalıdır [6].

6.2.1.2. Yöneticide IPVS desteğinin devreye alınması

Yöneticide IPVS desteği devreye alınmalıdır. IPVS (IP Virtual Server) Linux çekirdeğinde Layer-4 switching yapısını oluşturur. Aşağıdaki komutlar kullanılarak bu işlem gerçekleştirilir.

```
echo ip_vs_dh >> /etc/modules
echo ip_vs_ftp >> /etc/modules
echo ip_vs >> /etc/modules
echo ip_vs_lblc >> /etc/modules
echo ip_vs_lblcr >> /etc/modules
echo ip_vs_lc >> /etc/modules
echo ip_vs_nq >> /etc/modules
echo ip_vs_rr >> /etc/modules
echo ip_vs_sed >> /etc/modules
echo ip_vs_sh >> /etc/modules
echo ip_vs_wlc >> /etc/modules
echo ip_vs_wrr >> /etc/modules
```

Daha sonra aşağıdaki komutlar kullanılır.

```
modprobe ip_vs_dh
modprobe ip_vs_ftp
modprobe ip_vs
modprobe ip_vs_lblc
modprobe ip_vs_lblcr
modprobe ip_vs_lc
modprobe ip_vs_nq
modprobe ip_vs_rr
modprobe ip_vs_sed
modprobe ip_vs_sh
modprobe ip_vs_wlc
modprobe ip_vs_wrr
```

Eğer ekrana hata mesajları geliyorsa çekirdek IPVS desteği ile derlenmemiştir. Çekirdeğin tekrar IPVS desteği ile derlenmesi gerekir.

6.2.1.3. Konfigürasyon dosyalarının oluşturulması

`/etc/ha.d/ldirectord.cf` dosyası aşağıdaki gibi olmalıdır.

ldirectord.cf **konfigürasyon dosyası**

```
#
# Ldirectord will periodically connect to each real server
# and request a known URL. If the data returned by the server
# does not contain the the expected response then the
# test fails and the real server will be taken out of the
available
# pool. The real server will be added back into the pool once
the
# test succeeds. If all real servers are removed from the pool
then
# localhost is added to the pool as a fallback measure.
#

# Global Directives
checktimeout=10
checkinterval=2
#fallback=127.0.0.1:80
autoreload=no
#logfile="/var/log/ldirectord.log"
logfile="local0"
quiescent=yes

# Virtual Server for HTTP
virtual=193.255.141.39:80
        fallback=127.0.0.1:80
        real=192.168.0.2:80 masq
        real=192.168.0.3:80 masq
        service=http
        request="ldirectord.html"
        receive="Test Page"
        scheduler=rr
        #persistent=600
        protocol=tcp
      checktype=negotiate

# Virtual Service for HTTPS
virtual=193.255.141.39:443
        fallback=127.0.0.1:443
        real=192.168.0.2:443 masq
        real=192.168.0.3:443 masq
        service=https
        request="ldirectord.html"
        receive="Test Page"
        scheduler=rr
        #persistent=600
```

```
        protocol=tcp
      checktype=negotiate

# Virtual Service for FTP
# Note that peresistancy needs to be turned on for FTP when
# used with LVS-TUN (ipip) or LVS-DR (gate), but not with LVS-
NAT (masq).
virtual=193.255.141.39:21
        fallback=127.0.0.1:21
        real=192.168.0.2:21 masq
        real=192.168.0.3:21 masq
        service=ftp
        request="welcome.msg"
        receive="Welcome"
      login="anonymous"
        passwd="anon@anon.anon"
        scheduler=rr
        #persistent=600
        protocol=tcp
      checktype=negotiate

## Virtual Service for IMAP
#virtual=193.255.141.39:143
#        fallback=127.0.0.1:143
#        real=192.168.0.2:143 masq
#        real=192.168.0.3:143 masq
#        service=imap
#    #login="test"
#        #passwd="test"
#        scheduler=rr
#        #persistent=600
#        protocol=tcp
#        checktype=negotiate
#
## Virtual Service for POP
#virtual=193.255.141.39:110
#        fallback=127.0.0.1:110
#        real=192.168.0.2:110 masq
#        real=192.168.0.3:110 masq
#        service=pop
#    #login="test"
#        #passwd="test"
#        scheduler=rr
#        #persistent=600
#        protocol=tcp
#
## Virtual Service for SMTP
#virtual=193.255.141.39:25
#        fallback=127.0.0.1:25
#        real=192.168.0.2:25 masq
#        real=192.168.0.3:25 masq
#        service=smtp
#        scheduler=rr
```

```
#        #persistent=600
#        protocol=tcp
#
## Virtual Service for LDAP
#virtual=193.255.141.39:389
#        fallback=127.0.0.1:389
#        real=192.168.0.2:389 masq
#        real=192.168.0.3:389 masq
#        service=ldap
#        scheduler=rr
#        #persistent=600
#        protocol=tcp
#

#Sample configuration for an nntp virtual service.
#Fallback setting overides global
#virtual=193.255.141.39:119
#     real=192.168.0.2:119 masq
#     real=192.168.0.3:119 masq
#     fallback=127.0.0.1:119
#     service=nntp
#     scheduler=wlc
#     #persistent=600
#     #netmask=255.255.255.255
#     protocol=tcp

#Sample configuration for a UDP DNS virtual service.
#Fallback setting overides global
virtual=193.255.141.39:53
      real=192.168.0.2:53 masq
      real=192.168.0.3:53 masq
      fallback=127.0.0.1:53
      request="heartbeat.tu-suncluster.com"
      recieve="127.0.0.1"
      service=dns
      scheduler=wlc
      #persistent=600
      #netmask=255.255.255.255
      protocol=udp

#Sample configuration for a MySQL virtual service.
#virtual = 193.255.141.39:3306
#     real=192.168.0.2:3306 masq
#     real=192.168.0.3:3306 masq
#     fallback=127.0.0.1:3306 masq
#     checktype = negotiate
#     login = "readuser"
#     passwd = "genericpassword"
#     database = "portal"
#     request = "SELECT * FROM link"
#     scheduler = wrr

#Sample configuration for an unsuported protocol
```

```
#The real servers will just be brought up without checking for
availability
#Fallback setting overides global
#virtual=193.255.141.39:23
#     real=192.168.0.2:23 masq
#     real=192.168.0.3:23 masq
#     fallback=127.0.0.1:21
#     service=none
#     scheduler=wlc
#     request="welcome.msg"
#     receive="test"
#     persistent=600
#     #netmask=255.255.255.255
#     protocol=tcp
```

`/etc/ha.d/ha.cf` dosyası aşağıdaki gibi olmalıdır.

***ha.cf* konfigürasyon dosyası**

```
logfacility     local0
keepalive 2
deadtime 10
warntime 10
initdead 10
nice_failback on
udpport 694
bcast   eth1
node    ld1
node    ld2
```

`/etc/ha.d/haresources` dosyası aşağıdaki gibi olmalıdır.

***haresources* konfigürasyon dosyası**

```
ld1 IPaddr::192.168.0.1/24/eth1 IPaddr::193.255.141.39/24/\
eth1 ldirectord::ldirectord.cf
```

6.2.1.4. Paket iletim ayarlarının yapılması

Linux Virtual Server sisteminde yönetici dış ağdan gelen trafiği sunucu düğümlerine yönlendirmekten sorumludur. LVS-NAT yapısında geri dönen paketlerden de sorumludur. Bu nedenle ağ kartlarının ve yolların düzgün olarak çalışabilmesi için

IPV4 forwarding işlemi devreye alınmalıdır. Bu işlem `/etc/sysctl.conf` dosyasına aşağıdaki satırların eklenmesiyle yapılır.

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
```

Değişikliklerin etkili olması için aşağıdaki komutlar kullanılmalıdır.

```
/sbin/sysctl -p
net.ipv4.ip_forward = 1
```

6.2.1.5. ldirectord ayarlarının yapılması

Sunucu düğümlerinin izlenmesi ve bu düğümlerin sunucu havuzuna alınması ve havuzdan çıkarılmasından ldirectord sorumludur. ldirectord `/etc/ha.d/ldirectord.cf` dosyasındaki ayarları kullanır.

Fedora 8 ve diğer tüm RedHat tabanlı işletim sistemlerinde aşağıdaki komutlar kullanılarak ldirectord programının 2,3,5 ve 5 run level seviyelerinde başlangıçta çalışması ve heartbeat programının açılışta çalışmaması sağlanır.

```
/sbin/chkconfig --level 2345 ldirectord on
/sbin/chkconfig --del heartbeat
```

Debian tabanlı işletim sistemlerinde yukarıdaki işlemin gerçekleştirilmesi için aşağıdaki komutlar kullanılmalıdır.

```
/usr/sbin/update-rc.d heartbeat start 75 2 3 4 5 . stop 05 0 1 6
.
/usr/sbin/update-rc.d -f heartbeat remove
```

heartbeat programının çalışmadığından ve ldirectord programının yeni konfigürasyonla çalıştığından emin olmak için aşağıdaki komutlar kullanılmalıdır.

```
/etc/init.d/heartbeat stop
```

```
/etc/init.d/ldirectord start
```

ldirectord debug ve durum bilgilerini syslog kullanarak `/var/log/messages` dosyasına yazar. Bu log dosyaları bir sorun durumunda kullanılır.

O anda çalışmakta olan Linux Virtual Server tablosu ipvsadm komutuyla görülebilir.

```
ipvsadm -L -n
IP Virtual Server version 1.0.11 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn
InActConn
TCP  193.255.141.39:21 rr persistent 600
  -> 192.168.0.2:21               Masq    1      0          0
  -> 192.168.0.3:21               Masq    1      0          0
TCP  193.255.141.39:80 rr
  -> 192.168.0.2:80               Masq    1      0          0
  -> 192.168.0.3:80               Masq    1      0          0
TCP  193.255.141.39:443 rr
  -> 192.168.0.2:443              Masq    1      0          0
  -> 192.168.0.3:443              Masq    1      0          0
```

ldirectord nin durumunun görülmesi için aşağıdaki komut kullanılabilir.

```
ldirectord ldirectord.cf status
```

```
ldirectord for /etc/ha.d/ldirectord.cf is running with pid:
1455
```

6.2.1.6. NAT ayarlarının yapılması

NAT (masq) LVS tarafından kullanılan iletim mekanizmasıdır. LVS dışarıdan gelen bağlantıların NAT işlemlerini kullandığı gibi sunuculardan dışarıya doğru olan NAT işlemlerini de kullanabilir.

NAT işlemi iptables init scripti kullanılarak ayarlanabilir.

```
# Flush existing rules in the nat table
/etc/init.d/iptables stop
```

```
Resetting built-in chains to the default ACCEPT policy:
[  OK  ]

# Masquerade for 192.168.7.0/24 bound for any host
/sbin/iptables -t nat -A POSTROUTING -j MASQUERADE -s
192.168.7.0/24

# Log all packets that attempt to be forwarded
# Useful for Debugging. Questionable for Production
#/sbin/iptables -t nat -A POSTROUTING -j LOG

# Save the rules
/etc/init.d/iptables save
Saving current rules to /etc/sysconfig/iptables:
[  OK  ]

# Make sure rules are activated on reboot (at run levels 2, 3, 4
and 5)
/sbin/chkconfig --level 2345 iptables on

# Activate the rules
/etc/init.d/iptables start
Flushing all current rules and user defined chains:
[  OK  ]
Clearing all current rules and user defined chains:
[  OK  ]
Applying iptables firewall rules:
[  OK  ]
```

Debian tabanlı işletim sistemlerinde aynı işlemin gerçekleştirilmesi için yöneticinin iç ağa bağlı olan ağ kartında masquerading işlemi gerçekleştirilmelidir. `/etc/network/interfaces` dosyasına aşağıdaki satırlar eklenmelidir.

```
auto eth1
iface eth1 inet static
   address 192.168.0.1
   netmask 255.255.255.0
   up iptables -t nat -A POSTROUTING -j MASQUERADE -s
192.168.0.0/24
   down iptables -t nat -D POSTROUTING -j MASQUERADE -s
192.168.0.0/24
```

Debian tabanlı işletim sistemlerinde yapılan değişikliklerin devreye alınması için aşağıdaki komut kullanılmalıdır.

```
/etc/init.d/networking restart
```

Her iki işletim sisteminde de yapılan masquerading ayarlarının görülebilmesi için aşağıdaki komutlar kullanılmalıdır.

```
/sbin/iptables -t nat -L POSTROUTING -n -v
Chain POSTROUTING (policy ACCEPT)
 pkts bytes target     prot opt source               destination
25957 1592K MASQUERADE  all  --  192.168.0.0/24       0.0.0.0/0
```

FTP Virtual Service kullanılıyorsa `ip_vs_ftp` çekirdek modülününde kullanılması gerekir. Bu işlem aşağıdaki komutla yapılır.

```
/sbin/modprobe ip_vs_ftp
```

Bu modülün boot sırasında çekirdeğe eklenmesi sağlanmalıdır.

Fedora 8 gibi RedHat tabanlı işletim sistemlerinde `modprobe ip_vs_ftp` satırı `/etc/rc.local` , Debian tabanlı işletim sistemlerinde ise `ip_vs_ftp` satırı `/etc/modules` e eklenmelidir.

Çekirdekte bulunan modüller `lsmod` komutuyla kontrol edilebilir. Modülün sadece `ip_vs_ftp` modülünü gösteren ekran çıktısı aşağıdaki gibidir.

```
/sbin/lsmod
Module                  Size  Used by
ip_vs_ftp               3232   0
```

6.2.2. Sunucu düğümlerde yapılması gereken işlemler

Sunucu düğümlerde gerekli olan servisleri yerine getiren daemonlar konfigüre edilmeli bunun yanında yöneticide bulunan `/etc/ha.d/ldirectord.cf` dosyasında tanımlanmış olan URL sunucuda bulunmalı ve dosyadaki `receive` satırındaki ifadeyi içermelidir. Örneğin küme web servisleri sunuyorsa sunucu düğümde http daemon doğru konfigürasyonla çalışmalıdır.

Her iki düğümde de ldirectord.html dosyası aşağıdaki gibi olmalıdır. Bu dosya Apache sunucu düğümlerinin durumlarının izlenmesi için kullanılır.

```
vi /var/www/ldirector.html
```

```
Test Page
```

Bunun yanında sunucu düğümlerde varsayılan ağ geçidi olarak yöneticinin iç ağa bağlı olan ağ kartının IP adresi girilmelidir.

Fedora 8 işletim sisteminde aşağıdaki ayarlar `/etc/sysconfig/network` dosyasına girilmelidir.

```
NETWORKING=yes
HOSTNAME=node1.tu-suncluster.com
GATEWAY=192.168.0.1
```

Bu ayarların devreye alınması için aşağıdaki komutlar kullanılmalıdır.

```
/etc/init.d/network restart
Shutting down interface eth0                               [  OK  ]
Setting network parameters                                 [  OK  ]
Bringing up interface lo                                   [  OK  ]
Bringing up interface eth0                                 [  OK  ]
```

Debian tabanlı işletim sistemlerinde gerekli ayarların yapılması için `/etc/sysconfig/network` dosyasına aşağıdaki satırlar girilmelidir.

```
auto eth0
iface eth0 inet static
   address 192.168.0.2
   netmask 255.255.255.0
   gateway 192.168.0.1
```

Ayarların devreye alınması için aşağıdaki komut kullanılmalıdır.

```
/etc/init.d/networking restart
Reconfiguring network interfaces: done.
```

Yapılan ayarların doğrulanması için aşağıdaki komut kullanılmalıdır.

```
/sbin/ip route show 0/0
```

```
default via 192.168.0.1 dev eth0
```

6.3. MySQL Veritabanı Sunucusunun Kurulumu ve Ayarları

MySQL veritabanı kurulacak sunucuda Fedora 8 işletim sistemi çalışmaktadır. Küme yapısındaki düğümlerle aynı ağa bağlıdır. Veritabanı sunucusunun IP adresi 192.168.0.4 olarak ayarlanmıştır.

Öncelikle terminale root olarak login olunmalıdır. Bu işlem için aşağıdaki komutlardan birisi kullanılabilir.

```
su --login

su -l

su -
```

MySQL kurulumu aşağıdaki komutla başlatılır.

```
# yum install mysql mysql-server
Loading "priorities" plugin
Loading "changelog" plugin
Loading "fastestmirror" plugin
Loading "allowdowngrade" plugin
Loading "kernel-module" plugin
Loading "fedorakmod" plugin
Loading "installonlyn" plugin
Loading "protectbase" plugin
Setting up Install Process
Setting up repositories
livna                     100% |=========================| 1.1 kB
00:00
updates                   100% |=========================| 1.2 kB
00:00
core                      100% |=========================| 1.1 kB
00:00
extras                    100% |=========================| 1.1 kB
00:00
Loading mirror speeds from cached hostfile
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 1.8 MB
00:06
extras    : ##################################################
5594/5594
0 packages excluded due to repository priority protections
```

```
0 packages excluded due to repository protections
Parsing package install arguments
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for mysql to pack into transaction set.
mysql-5.0.27-1.fc6.i386.r 100% |=========================|  36 kB
00:00
---> Package mysql.i386 0:5.0.27-1.fc6 set to be updated
---> Downloading header for mysql-server to pack into transaction set.
mysql-server-5.0.27-1.fc6 100% |=========================|  33 kB
00:00
---> Package mysql-server.x86_64 0:5.0.27-1.fc6 set to be updated
---> Downloading header for mysql to pack into transaction set.
mysql-5.0.27-1.fc6.x86_64 100% |=========================|  36 kB
00:00
---> Package mysql.x86_64 0:5.0.27-1.fc6 set to be updated
--> Running transaction check
--> Processing Dependency: perl-DBI for package: mysql-server
--> Processing Dependency: perl(DBI) for package: mysql
--> Processing Dependency: perl(DBI) for package: mysql-server
--> Processing Dependency: perl-DBD-MySQL for package: mysql-server
--> Restarting Dependency Resolution with new changes.
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for perl-DBI to pack into transaction set.
perl-DBI-1.52-1.fc6.x86_6 100% |=========================|  16 kB
00:00
---> Package perl-DBI.x86_64 0:1.52-1.fc6 set to be updated
---> Downloading header for perl-DBD-MySQL to pack into transaction
set.
perl-DBD-MySQL-3.0007-1.f 100% |=========================| 8.5 kB
00:00
---> Package perl-DBD-MySQL.x86_64 0:3.0007-1.fc6 set to be updated
--> Running transaction check

Dependencies Resolved

================================================================================
 Package                 Arch       Version          Repository
Size
================================================================================
Installing:
 mysql                   i386       5.0.27-1.fc6     updates
3.3 M
 mysql                   x86_64     5.0.27-1.fc6     updates
3.3 M
 mysql-server            x86_64     5.0.27-1.fc6     updates
10 M
Installing for dependencies:
 perl-DBD-MySQL          x86_64     3.0007-1.fc6     core
147 k
 perl-DBI                x86_64     1.52-1.fc6       core
605 k

Transaction Summary
================================================================================
Install      5 Package(s)
```

```
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 18 M
Is this ok [y/N]:
```

MySQL server daemon (mysqld) aşağıdaki komutla başlatılır.

```
# service mysqld start
Initializing MySQL database:  Installing all prepared tables
Fill help tables

To start mysqld at boot time you have to copy support-
files/mysql.server
to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:
/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h angstrom password 'new-password'
See the manual for more instructions.

You can start the MySQL daemon with:
cd /usr ; /usr/bin/mysqld_safe &

You can test the MySQL daemon with the benchmarks in the 'sql-bench'
directory:
cd sql-bench ; perl run-all-tests

Please report any problems with the /usr/bin/mysqlbug script!

The latest information about MySQL is available on the web at
http://www.mysql.com
Support MySQL by buying support/licenses at http://shop.mysql.com
                                                           [  OK  ]
Starting MySQL:                                            [  OK  ]
```

MySQL sunucuya *root database admin* olarak bağlanılır.

```
# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.27

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

MySQL sunucunun root database admin şifresi belirlenir.

```
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('mypass');
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Veritabanlarına kimliksiz erişim kaldırılır.

```
mysql> DELETE FROM mysql.user WHERE User = '';
Query OK, 2 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Tüm veritabanları için database admin yetkisinde yeni bir kullanıcı eklenir.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'webadmin'@'localhost'
IDENTIFIED BY 'mypass' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Kurulacak uygulamaya ait veritabanı oluşturulur.

```
mysql> create database kumetest;
Query OK, 1 row affected (0.15 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

Kurulacak uygulamaya ait veritabanı için database admin yetkilerine sahip yeni bir kullanıcı eklenir.

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP,
INDEX, ALTER, CREATE TEMPORARY TABLES, LOCK TABLES ON kumetest.*
TO 'admin'@'localhost' IDENTIFIED BY 'mypass';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

mysql-administrator kurulumu yapılır. Kurulumun tamamlanmasından sonra programa System Tools grubundan erişilebilir.

```
# yum install mysql-administrator
Loading "priorities" plugin
Loading "changelog" plugin
Loading "fastestmirror" plugin
Loading "allowdowngrade" plugin
Loading "kernel-module" plugin
Loading "fedorakmod" plugin
Loading "installonlyn" plugin
Loading "protectbase" plugin
Setting up Install Process
Setting up repositories
Loading mirror speeds from cached hostfile
Reading repository metadata in from local files
0 packages excluded due to repository priority protections
0 packages excluded due to repository protections
Parsing package install arguments
Resolving Dependencies
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for mysql-administrator to pack into
transaction set.
mysql-administrator-1.1.1 100% |=========================|  25 kB
00:00
---> Package mysql-administrator.x86_64 0:1.1.10-3.fc6 set to be
updated
--> Running transaction check
--> Processing Dependency: libsigc-2.0.so.0()(64bit) for package:
mysql-administrator
--> Processing Dependency: mysql-gui-common for package: mysql-
administrator
--> Processing Dependency: libgdkmm-2.4.so.1()(64bit) for package:
mysql-administrator
--> Processing Dependency: libpangomm-1.4.so.1()(64bit) for package:
mysql-administrator
--> Processing Dependency: libglibmm-2.4.so.1()(64bit) for package:
mysql-administrator
--> Processing Dependency: libcairomm-1.0.so.1()(64bit) for package:
mysql-administrator
--> Processing Dependency: libatkmm-1.6.so.1()(64bit) for package:
mysql-administrator
--> Processing Dependency: libgtkmm-2.4.so.1()(64bit) for package:
mysql-administrator
--> Restarting Dependency Resolution with new changes.
--> Populating transaction set with selected packages. Please wait.
---> Downloading header for gtkmm24 to pack into transaction set.
gtkmm24-2.10.5-1.fc6.x86_ 100% |=========================| 7.8 kB
00:00
---> Package gtkmm24.x86_64 0:2.10.5-1.fc6 set to be updated
---> Downloading header for cairomm to pack into transaction set.
cairomm-1.2.4-1.fc6.x86_6 100% |=========================| 5.2 kB
00:00
---> Package cairomm.x86_64 0:1.2.4-1.fc6 set to be updated
---> Downloading header for libsigc++20 to pack into transaction set.
```

```
libsigc++20-2.0.17-2.x86_ 100% |=========================| 6.1 kB
00:00
---> Package libsigc++20.x86_64 0:2.0.17-2 set to be updated
---> Downloading header for glibmm24 to pack into transaction set.
glibmm24-2.12.3-1.x86_64. 100% |=========================| 6.1 kB
00:00
---> Package glibmm24.x86_64 0:2.12.3-1 set to be updated
---> Downloading header for mysql-gui-common to pack into transaction
set.
mysql-gui-common-1.1.10-3 100% |=========================|  17 kB
00:00
---> Package mysql-gui-common.x86_64 0:1.1.10-3.fc6 set to be updated
--> Running transaction check

Dependencies Resolved

================================================================================
=======
 Package                 Arch       Version          Repository
Size
================================================================================
=======
Installing:
 mysql-administrator     x86_64     1.1.10-3.fc6     extras
1.5 M
Installing for dependencies:
 cairomm                 x86_64     1.2.4-1.fc6      extras
40 k
 glibmm24                x86_64     2.12.3-1         extras
145 k
 gtkmm24                 x86_64     2.10.5-1.fc6     extras
1.1 M
 libsigc++20             x86_64     2.0.17-2         extras
49 k
 mysql-gui-common        x86_64     1.1.10-3.fc6     extras
208 k

Transaction Summary
================================================================================
=======
Install      6 Package(s)
Update       0 Package(s)
Remove       0 Package(s)

Total download size: 3.1 M
Is this ok [y/N]:
```

Kurulum işlemleri bittikten sonra doğru sürümlerin kurulduğundan emin olmak için aşağıdaki komut dizisi kullanılmalıdır.

```
# rpm -qa | grep mysql && chkconfig --list | grep mysql
mysql-5.0.27-1.fc6
mysql-5.0.27-1.fc6
mysql-gui-common-1.1.10-3.fc6
mysql-server-5.0.27-1.fc6
```

```
mysql-administrator-1.1.10-3.fc6
mysqld          0:off   1:off   2:off   3:off   4:off   5:off
6:off
```

MySQL daemonunun boot sırasında çalışması için aşağıdaki komut dizisi kullanılmalıdır.

```
# chkconfig mysqld on && service mysqld restart && chkconfig --
list | grep mysqld
Stopping MySQL:                                            [  OK  ]
Starting MySQL:                                            [  OK  ]
mysqld          0:off   1:off   2:on    3:on    4:on    5:on
6:off
```

6.3.1. Veritabanı sunucusundaki MySQL veritabanına düğümlerden erişilebilmesi için yapılması gerekenler

Öncelikle my.cnf dosyası düzenlenir. Dosyanın konumu:

- Debian Linux işletim sisteminde /etc/mysql/my.cnf
- Red Hat Linux/Fedora Linux işletim sistemlerinde /etc/my.cnf
- FreeBSD işletim sisteminde /var/db/mysql/my.cnf

```
# vi /etc/my.cnf
```

Dosya açıldıktan sonra `[mysqld]` satırının bulunduğu yerde aşağıdaki değişiklikler yapılır ve dosya kaydedilir.

```
[mysqld]
user = mysql
pid-file = /var/run/mysqld/mysqld.pid
socket = /var/run/mysqld/mysqld.sock
port = 3306
basedir = /usr
datadir = /var/lib/mysql
tmpdir = /tmp
language = /usr/share/mysql/English
bind-address = 192.168.0.4
# skip-networking
....
```

```
..
....
```

Mysql servisi restart edilir. Bu işlem aşağıdaki komutla gerçekleştirilir.

```
# /etc/init.d/mysql restart
```

Uzaktaki IP adresine erişim yetkisi verilir.

```
# mysql –u root –p mysql
```

Sunucu üzerinde yeni yaratılacak olan veritabanına dışarıdan erişim yetkisi verilir.

```
mysql> CREATE DATABASE kumetest2;
mysql> GRANT ALL ON kumetest2.* TO webadmin@'192.168.0.2'
IDENTIFIED BY 'PASSWORD';

mysql> GRANT ALL ON kumetest2.* TO webadmin@'192.168.0.3'
IDENTIFIED BY 'PASSWORD';
```

Varolan bir veritabanına dışarıdan erişim yetkisi verebilmek için aşağıdaki komutlar kullanılır.

```
mysql> update db set Host='192.168.0.2' where Db='kumetest';
mysql> update user set Host='192.168.0.2' where user='webadmin';

mysql> update db set Host='192.168.0.3' where Db='kumetest';
mysql> update user set Host='192.168.0.3' where user='webadmin';
```

Mysql den çıkış yapılır.

```
mysql> exit
```

Yapılan ayarların kontrol edilmesi için sunucu düğümlerde aşağıdaki komut kullanılır.

```
$ mysql –u webadmin –h 192.168.0.4 –p
```

Aynı işlem telnet ile 3306 portuna bağlanma yöntemiyle de denenebilir.

```
$ telnet 192.168.0.4 3306
```

6.4. rsync İle Web Sitesi İçeriğinin Düğümler Arasında Senkronizasyonu

rsync diğer yöntemlere göre verilerin eşleştirilmesinde daha performanslı bir çözümdür. rsync sadece değişen verileri kopyalar. rsync ana sunucuda silinmiş olan dosyaları yedek sunucudan da siler. Küme yapısında Apache sunucularda bulunan web sayfalarının eşleştirilmesi için kullanılacaktır. 1. Sunucu düğümünde yapılacak değişikliler 2. Sunucu düğümünde de yapılacaktır. Düğümler arasında değişen dosyalar kopyalanacak, yeni dosyalar kopyalanacak, silinen dosyalar ise silinecektir. Dosya ve dizinlerin izinlerinin ve sahipliklerinin korunması için rsync her iki düğümde de root kullanıcısıyla çalıştırılacaktır.

rsync işlemi SSH desteği ile birlikte yapılacaktır. Böylece güvenlik sağlanacaktır. SSH port 22 yi kullanmaktadır. SSH işleminde düğümlerde bulunan public key kullanılmaktadır. Bu key sayesinde rsync işlemi sırasında şifre sorulmayacaktır.

Konfigürasyon sırasında kullanılacak değerler aşağıdaki gibidir.

Sunucu düğümü-1 : node1.tu-suncluster.com IP adresi: 192.168.0.2

Sunucu düğümü-2 : node2.tu-suncluster.com IP adresi: 192.168.0.3

Eşleştirme işlemi yapılacak dizin: `/var/www`

- *rsync kurulumunun yapılması*

İlk yapılması gereken her iki sunucu düğümünde de rsync kurulumunun yapılmasıdır.

Fedora sistemlerde bu işlem aşağıdaki gibi yapılır.

Her iki düğümde de öncelikle root olarak login olunur ve aşağıdaki komut kullanılır.

```
yum install rsync
```

Debian sistemlerde bu işlem aşağıdaki gibi yapılır.

Her iki düğümde de öncelikle root olarak login olunur ve aşağıdaki komut kullanılır.

```
apt-get install rsync
```

- *Sunucu düğümü-1'de yetkisiz bir kullanıcı yaratılması*

Sunucu düğümü-1'de yetkisiz bir kullanıcı yaratılmalıdır. Bu kullanıcı Sunucu düğümü-2'de /var/www dizininin eşleştirilmesi için yapılacak rsync işlemi sırasında kullanılacaktır. Kullanıcının Sunucu düğümü-1'de /var/www dizinini okuma yetkisi olmalıdır. Komut root olarak çalıştırılmalıdır.

```
useradd -d /home/someuser -m -s /bin/bash someuser
```

Daha sonra kullanıcıya şifre atanmalıdır.

```
passwd someuser
```

- *rsync'in test edilmesi*

Sunucu düğümü-2'de rsync'i test etmek için aşağıdaki komut kullanılır.

```
rsync -avz -e ssh someuser@node1.tu-suncluster.com:/var/www/
/var/www/
```

Aşağıdaki gibi bir ekran çıktısı olmalıdır. Ekrana gelecek soruya `yes` yanıtı verilir.

```
The authenticity of host 'node1.tu-suncluster.com
(192.168.0.2)' can't be established.
RSA key fingerprint is
32:e5:79:8e:5f:5a:25:a9:f1:0d:ef:be:5b:a6:a6:23.
Are you sure you want to continue connecting (yes/no)?
```

```
<-- yes
```

someuser kullanıcısının şifresi girildiğinde Sunucu düğümü-1'de bulunan `/var/www` dizini Sunucu düğümü-2'de bulunan `/var/www` diziniyle eşleştirilmelidir.

Aşağıdaki komut her iki düğümde çalıştırılarak aynı dizin ve dosyaların olup olmadığı kontrol edilebilir.

```
ls –la /var/www
```

- *Sunucu düğümü-2'de key lerin yaratılması*

Sunucu düğümü-2'de private/public key çifti yaratılmalıdır. Bu işlem root kullanıcısıyla yapılmalıdır.

```
mkdir /root/rsync
ssh-keygen –t dsa –b 2048 –f /root/rsync/mirror-rsync-key
```

Ekrana aşağıdaki gibi bir mesaj gelmelidir.

```
Generating public/private dsa key pair.
Enter passphrase (empty for no passphrase): [press enter here]
Enter same passphrase again: [press enter here]
Your identification has been saved in /root/cron/mirror-rsync-
key.
Your public key has been saved in /root/cron/mirror-rsync-
key.pub.
The key fingerprint is:
68:95:35:44:91:f1:45:a4:af:3f:69:2a:ea:c5:4e:d7 root@node2
```

Daha sonra root kullanıcısıyla public key Sunucu düğümü-1'e kopyalanır.

```
scp      /root/rsync/mirror-rsync-key.pub      someuser@node1.tu-
suncluster.com:/home/someuser/
```

Public key (mirror-rsync-key.pub) Sunucu düğümü-1'de /home/someuser dizininde olmalıdır.

- *node1.tu-suncluster.com düğümünün ayarlanması*

SSH ile someuser kullanıcısını kullanarak node1.tu-suncluster.com a login olunur ve aşağıdaki komutlar kullanılır.

```
mkdir ~/.ssh
chmod 700 ~/.ssh
mv ~/mirror-rsync-key.pub ~/.ssh/
cd ~/.ssh
touch authorized_keys
chmod 600 authorized_keys
cat mirror-rsync-key.pub >> authorized_keys
```

Bu işlemlerle mirror-rsync-key.pub dosyasının içeriğini /home/someuser/.ssh/authorized_keys dosyasına eklenir. Dosya içeriği aşağıdaki gibi olmalıdır.

```
vi /home/someuser/.ssh/authorized_keys
```

```
ssh-dss AAAAB3NzaC1kc3MAAA[...]lSUom root@
node2
```

Sadece node2.tu-suncluster.com makinesinden gelen bağlantılara izin vermek için aşağıdaki komut /home/someuser/.ssh/authorized_keys dosyasının başına eklenmelidir. Bağlanan kullanıcı sadece rsync kullanabilecektir.

```
command="/home/someuser/rsync/checkrsync",from="node2.tu-
suncluster.com",no-port-forwarding,no-X11-forwarding,no-pty
```

Dosyanın içeriğine tekrar bakılırsa aşağıdaki görüntü ekrana gelecektir.

```
vi /home/someuser/.ssh/authorized_keys
```

```
command="/home/someuser/rsync/checkrsync",from="node2.tu-
suncluster.com",no-port-forwarding,no-X11-forwarding,no-pty ssh-dss
AAAAB3NzaC1kc3MAAA[...]lSUom root@
node2
```

Daha sonra /home/someuser/rsync/checkrsync dosyası oluşturularak script in rsync dışında tüm komutları reddetmesi sağlanır.

```
mkdir ~/rsync
vi ~/rsync/checkrsync
```

```
#!/bin/sh

case "$SSH_ORIGINAL_COMMAND" in
        *\&*)
                echo "Rejected"
                ;;
        *\(*)
                echo "Rejected"
                ;;
        *\{*)
                echo "Rejected"
```

```
                ;;
        *\;*)
                echo "Rejected"
                ;;
        *\<*)
                echo "Rejected"
                ;;
        *\`*)
                echo "Rejected"
                ;;
        rsync\ --server*)
                $SSH_ORIGINAL_COMMAND
                ;;
        *)
                echo "Rejected"
                ;;
esac
```

```
chmod 700 ~/rsync/checkrsync
```

- *Sunucu düğümü-2'de rsync işleminin test edilmesi*

Sunucu düğümü-2'de bir test işlemi gerçekleştilererek someuser kullanıcısının şifresi girilmeden Sunucu düğümü-1'den bir eşleştirme işleminin gerçekleşip gerçekleşmediği görülebilir.

root kullanıcısı olarak aşağıdaki komut çalıştırılır.

```
rsync -avz --delete --exclude=**/stats --exclude=**/error --
exclude=**/files/pictures -e "ssh -i /root/rsync/mirror-rsync-
key" someuser@node1.tu-suncluster.com:/var/www/ /var/www/
```

(--delete seçeneği Sunucu düğümü-1'de silinen tüm dosyaların Sunucu düğümü-2'de de silineceğini belirtir. --exclude seçeneği ile belirtilen dizinlerin eşleştirme işlemine dahil edilmeyeceği belirtilir.

Aşağıdaki ekran eşleştirme işleminin gerçekleştiğini gösterir.

```
receiving file list ... done

sent 71 bytes  received 643 bytes  476.00 bytes/sec
total size is 64657  speedup is 90.56
```

- *Cron job yaratılması*

Senkronizasyon işleminin otomatik hale getirilmesi için Sunucu düğümü-2 de cron job yaratılır. root kullanıcısı olarak login olunarak aşağıdaki komut kullanılır ve şekilde görülen gibi bir cron job oluşturulur.

```
crontab –e
```

```
*/5 * * * * /usr/bin/rsync –azq --delete --exclude=**/stats --
exclude=**/error --exclude=**/files/pictures –e "ssh –i
/root/rsync/mirror-rsync-key"
someuser@server1.example.com:/var/www/ /var/www/
```

Yukarıdaki iş her 5 dakikada bir çalışacaktır.

BÖLÜM 7

7. LINUX KÜMESİNDE PERFORMANS TESTLERİ

7.1. Linux Kümelerinde Performans Ölçümünde Kullanılan Araçlar

Linux kümelerinde performans ölçümü amacıyla çeşitli benchmark testleri vardır. Testler genel olarak düşük seviye testler ve uygulama performansı testleri olarak iki grupta toplanır.

Düşük seviye testler işlemci gücünü, I/O performansını, bellek bant genişliğini ve alt katmanda kullanılan ağ teknolojisinin performansını ölçmeyi amaçlar.

CPU performansını karşılaştırmada en sık kullanılan testler SPEC firmasının testleridir. Bu testler tam sayı ve kayan nokta aritmetik işlemlerindeki performansı ölçmektedir.

I/O performansında kullanılan bazı testler ise Effective I/O Bandwidth Benchmark, ParkBench, BTIO gibi MPI performansını ölçen testler ve Bonnie++ gibi dosya sistemi ve disk performansını ölçen testlerdir.

Bellek bant genişliğini ölçmede en sık kullanılan testlerden biri STREAM dir.

İletişim performansını ölçen testler ise, ağ karakteristiklerini ölçen testler (NetPipe, NetPerf gibi) ve ağ üzerindeki gecikme ve çıktı miktarını, MPI arayüzünü kullanarak test eden Pallas MPI Benchmark ve Mpptest gibi testlerdir.

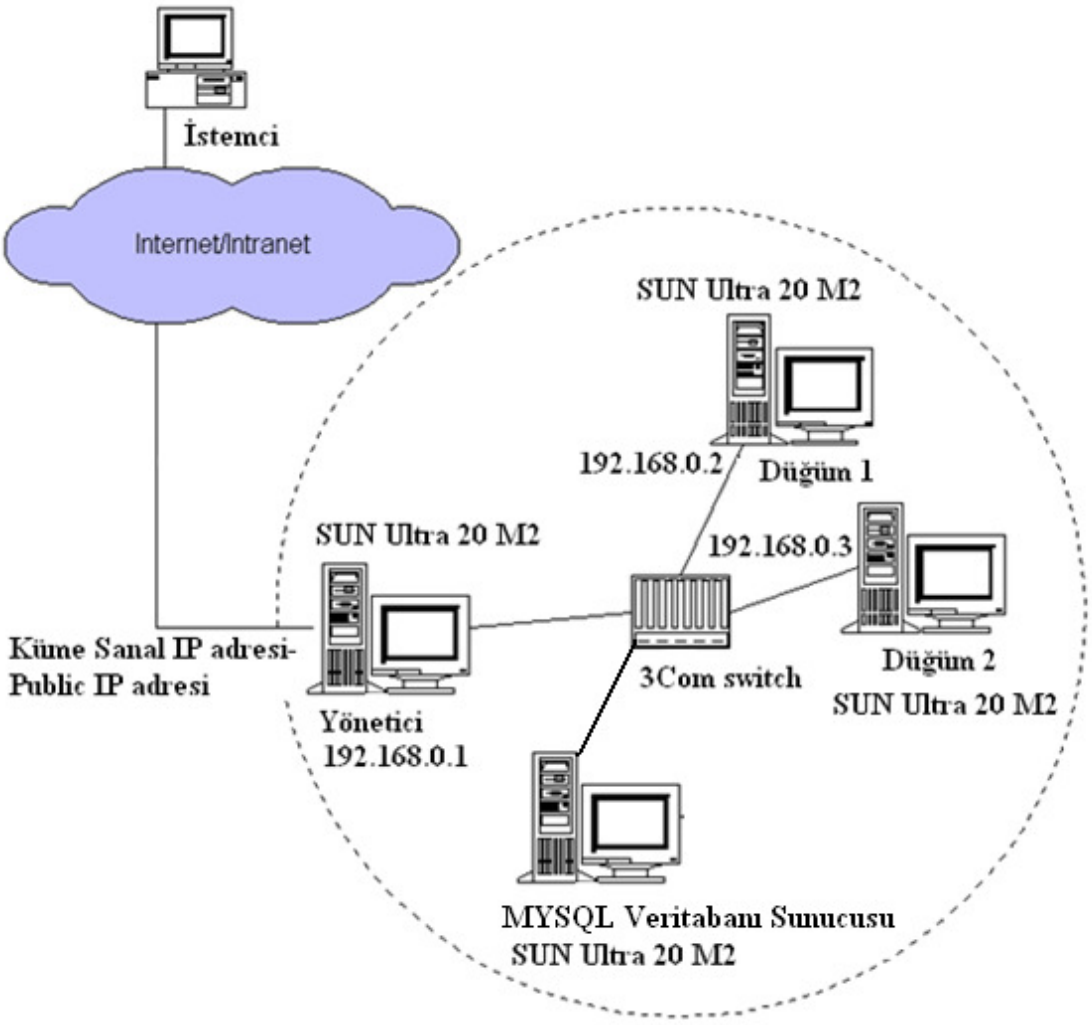
NetPIPE protokol bağımsız performans ölçüm yazılımıdır. NetPIPE programının kullanım amaçları şunlardır;

- Farklı ağ kartları arasında gecikme ve maksimum çıkış değeri karşılaştırmaları
- Farklı ağlar arasında performans karşılaştırmaları
- Mesaj iletim katmanlarındaki yetersizliklerin bulunması
- Mesaj iletim katmanının optimizasyonu ve iletişim alt sisteminin en iyi performansı için işletim sistemi ve sürücü parametrelerinin ayarlanması

Uygulama performansını ölçmek amacıyla yaygın şekilde kullanılan ve standart olarak kabul edilen testler ise NAS Parallel Benchmarks, HPL ve POVRay olarak sıralanabilir.

7.2. Test Ortamı

Test ortamında kullanılan küme yapısı aşağıdaki şekilde görülebilir.



Şekil 7.1. Linux Küme Yapısı

7.2.1. Küme yapısında çalışan yazılımlar

Küme yapısında bulunan Yönetici düğümde ve sunucu düğümlerde Fedora 8 işletim sistemi çalışmaktadır. Yönetici düğümde LVS director yazılımı (ipvs-1.24-9) çalışmaktadır. ipvs-1.24-9 Linux 2.6 çekirdeğine uygun olan LVS yazılımıdır. Bu yazılım küme yapısındaki tüm iletişimden sorumludur.

Küme yapısı üzerinde web uygulamaları çalıştırılacağı için testlerin Apache sunucular üzerinde yapılmasına karar verilmiştir. Fedora 8 işletim sistemi üzerinde Apache Http Server 2.2.6 gelmektedir. Düğümlerde çalışan Apache sunucular üzerine küçük bir HTML sayfası yüklenmiştir.

7.2.1.1. İstemci bilgisayarlarda çalışan yazılımlar

İstemci bilgisayar üzerinde de Debian Linux 4.0r3 çalışmaktadır. İstemci bilgisayarlara httpperf 0.9.0 (ftp://ftp.hpl.hp.com/pub/httperf/) yazılımı yüklenmiştir. Httperf web sunucu performansını ölçmek için kullanılan bir yazılımdır. Httperf web sunucuya belirli bir değerde bağlantı istekleri gönderir ve gelen yanıtlara göre gerçek yanıtlama oranını hesaplar.

httperf --hog --server 10.40.48.38 --port 80 --uri index.html -t 8

10.40.48.38 küme yapısının iç ağ da kullanılan IP adresidir. Dışarıdan erişildiğinde bu değer yerine kümenin Internet üzerinde kullanıldığı gerçek IP adresi yazılmalıdır.

t parametresi ile belirtilen timeout değeri ne kadar sürede yanıt gelmediğinde hata oluşacağını gösteren değerdir.

LVS Yönetici performansının ölçümü için ise testlvs yazılımı kullanılmıştır. Testlvs birçok istemcinin bulunduğu bir ortamı sahta kaynak adresleri kullanarak simüle eder ve ham IP paketleri gönderir.

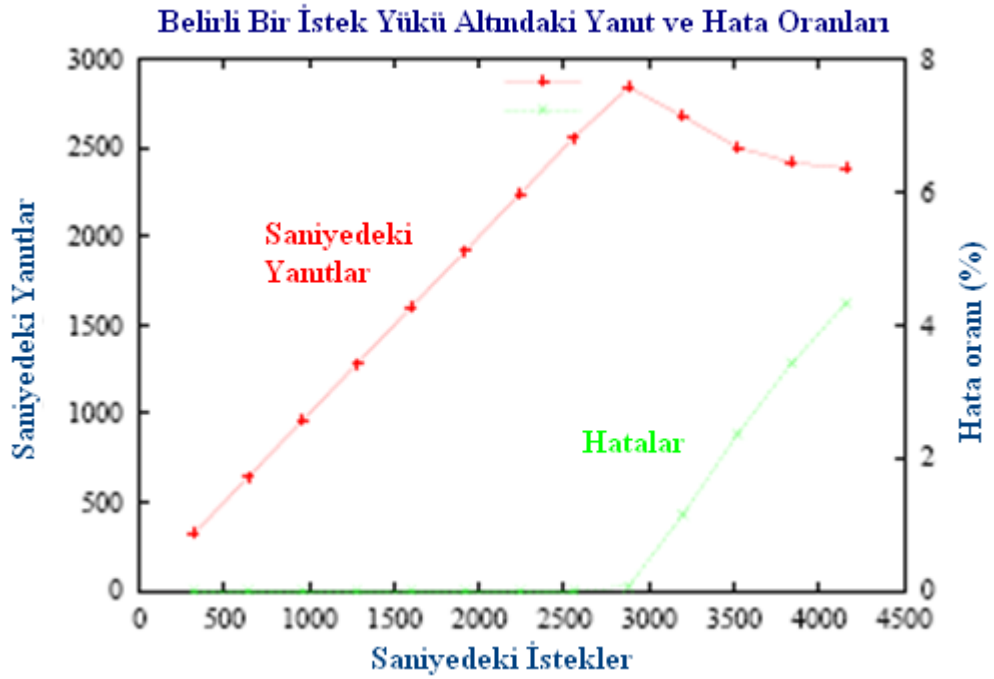
7.3. Test Aşaması 1 – httperf İle Web Uygulamalarında Performans Ölçümü

Her test kümeye saniyedeki istek sayısı ile ölçülen bir istek oranı uygulamaktadır. Belirli bir istek oranında yük uygulanan küme de geriye yanıtlar göndermektedir. Geri dönen yanıt oranı istek oranına eşit veya bu orandan az olabilir. Eğer yanıt oranı istek oranından düşükse kümedeki sunucuların en az birinin maksimum kapasitede çalıştığını söyleyebiliriz.

Performans testi düşük bir istek oranında başlatılmış ve saniyede 320 istek oranında arttırılmıştır.

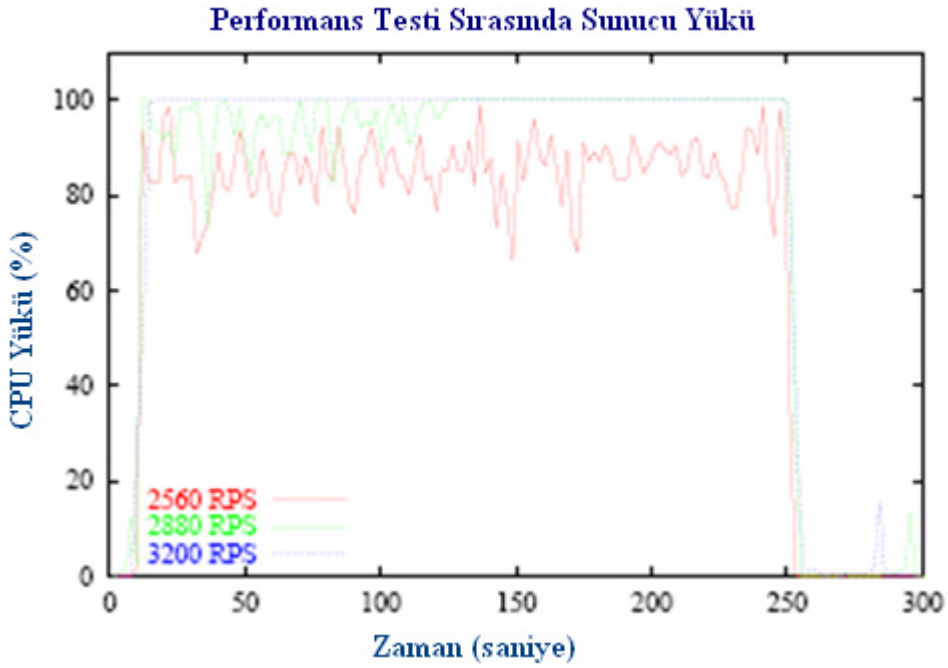
İki adet performans testi yapılmıştır. İlk testte kümedeki iki düğümde de Apache sunucular üzerinde aynı HTML sayfası bulunmaktadır. İkinci testte ise ikinci düğümde bulunan HTML sayfası yerine CGI script koyulmuştur. CGI script kullanılmasının nedeni çalıştırılabilmesi için daha fazla işlemci gücü gerektirmesidir. Böylece sunuculardan birisi daha fazla istek almış ve eşit olmayan yük dağılımı bulunan bir küme yapısı oluşturulmuştur.

Küme Performansı: İlk testte bir yük değeri uygulanmış ve saniyedeki istek oranı 320 arttırılmıştır. Bu arttırma işlemi saniyede 4160 isteğe kadar devam ettirilmiştir.



Şekil 7.2. Eşit Yüklü Kümede Yanıtlama Performansı ve Hata Oranları

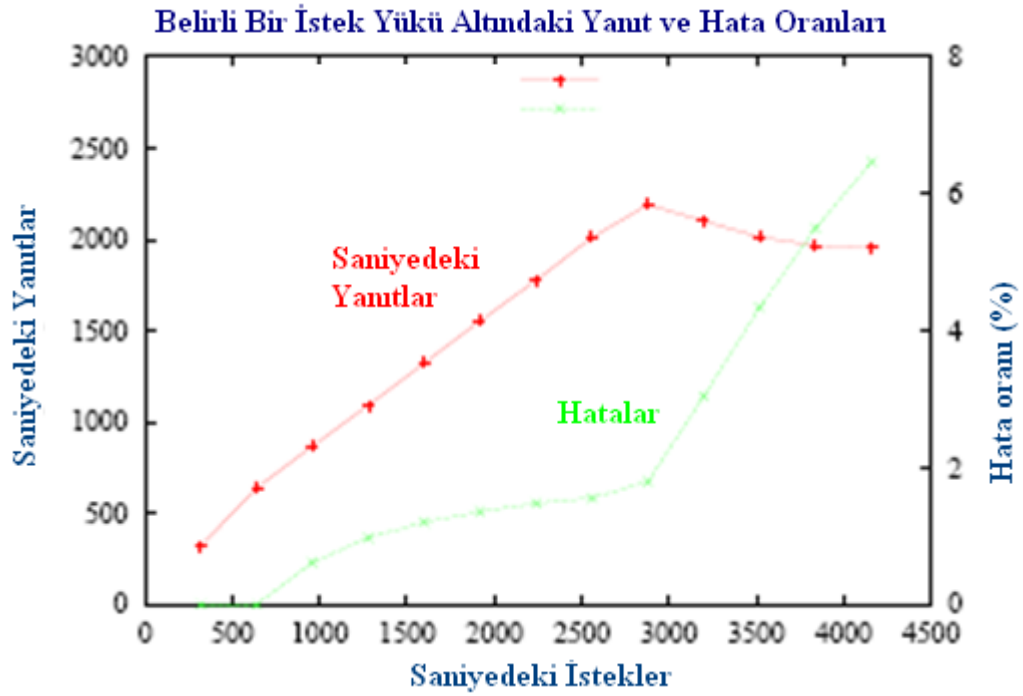
Şekilde de görüldüğü gibi eşit yük dağılımlı küme de saniyedeki 2880 istek maksimum uygulanabilecek değerdir. Bu değerden sonra sunucular aşırı yüklenmiş olmakta ve yanıtlama oranları düşmektedir.



Şekil 7.3. Farklı İstek Oranlarında CPU Yük Değerleri

Bu test sonucunda varılabilecek olan kanılardan birisi sunuculardaki CPU yükünün yanıtlama oranını kısıtladığıdır.

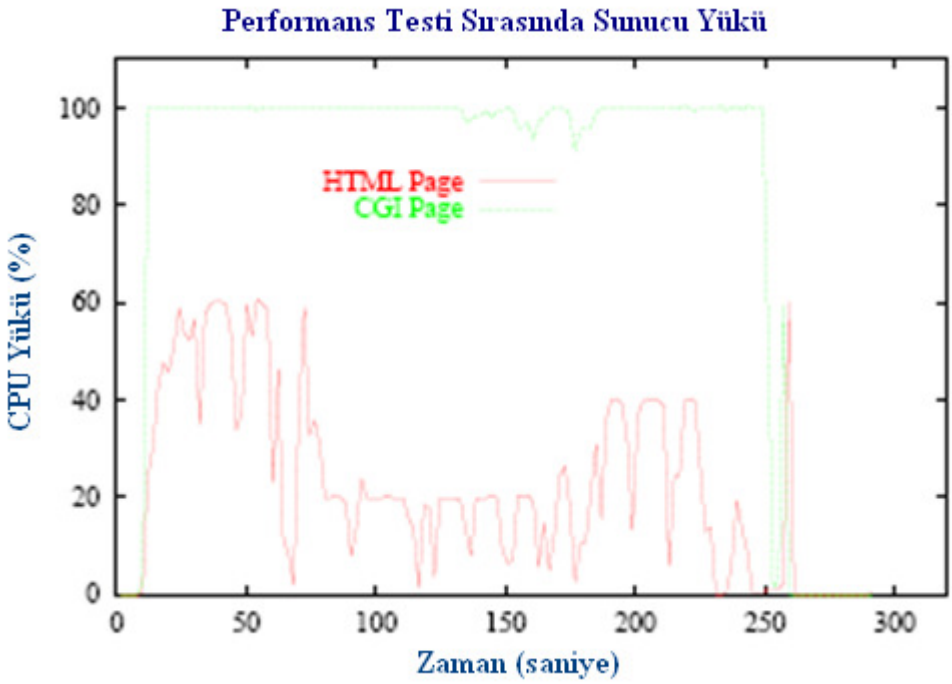
Bu durumda yapılabilecek olan iki şey vardır. Birincisi sunucularda kullanılan işlemcilerin daha yüksek hızlı işlemcilerle değiştirilmesidir. İkincisi ise küme yapısında bulunan sunucu düğümü sayısının arttırılmasıdır. Küme yapılarında tercih edilen ikinci yöntemdir.



Şekil 7.4. Eşit Yüklü Olmayan Kümede Yanıtlama Performansı ve Hata Oranları

CGI scriptinden dolayı eşit yüklü olmayan kümedeki hata oranları eşit yüklü kümeye göre daha yüksektir. Bunun nedeni CGI scriptini çalıştıran sunucunun diğer sunucudan önce %100 yük seviyesine ulaşmasıdır. Saniyede 2880 isteğe kadar olan hata oranları belirli bir oranda iken bu değerden sonra hızla artmıştır. Bunun nedeni HTML sayfa bulunduran sunucununda aşırı yüklü duruma gelmiş olmasıdır.

Aşağıdaki şekilde saniyede 640 istek oranında CPU yük değerleri görülmektedir. CGI scriptinden dolayı daha yüksek yük değerlerine ulaşılmıştır. HTML ve CGI çalıştıran sunucuların CPU kullanım değerleri farklıdır. Bunun nedeni CGI scriptini çalıştırmak için daha fazla CPU kaynaklarının kullanılmasıdır.

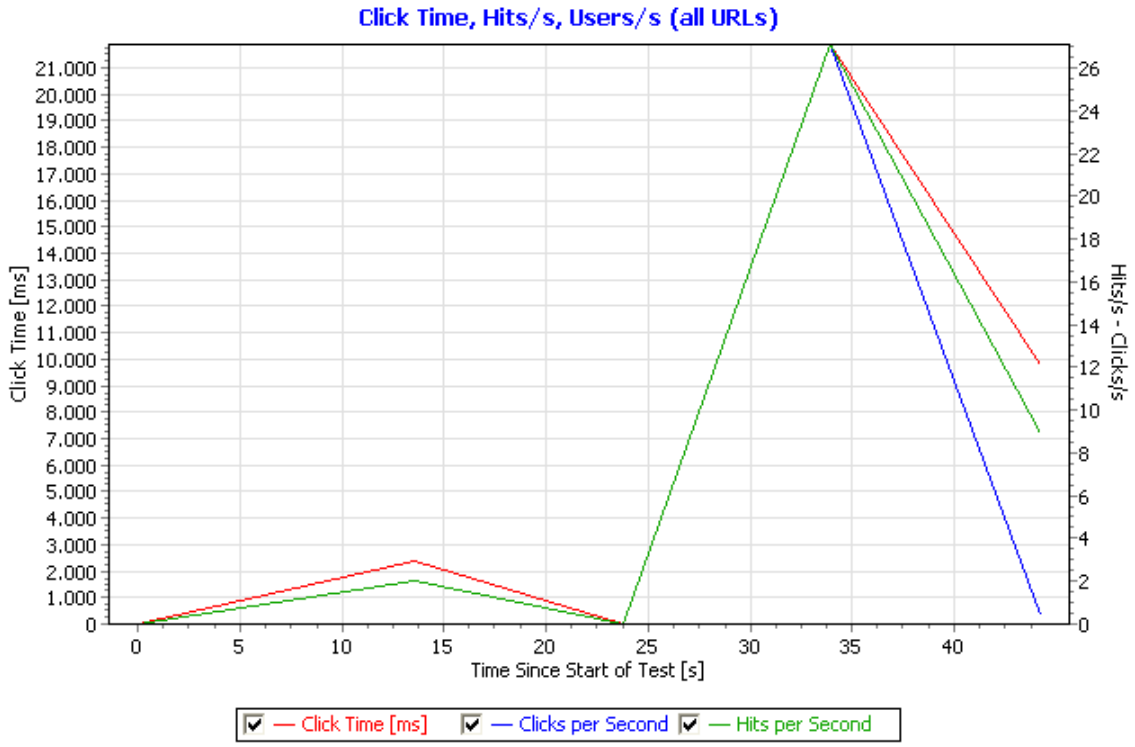


Şekil 7.5. Eşit Yüklü Olmayan Kümede Saniyede 640 İstek Oranında CPU Yük Değerleri

7.4. Test Aşaması 2 – Paessler Web Server Stress Tool 7 İle Performans Ölçümü

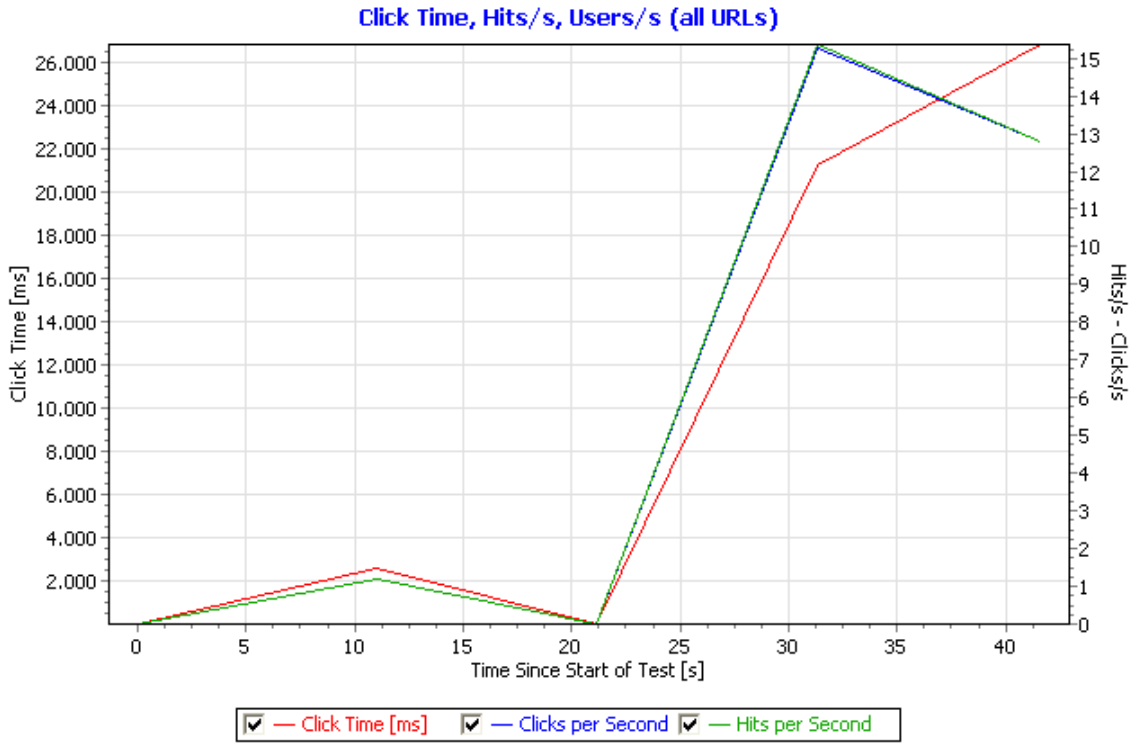
Test sırasında 4 KB'lık basit bir HTML sayfası kullanılmıştır. Testin amacı sanal kullanıcı sayısı arttırıldığında sayfanın getirilme süresinde meydana gelecek değişiklikleri incelemektir.

Aynı anda 1000 kullanıcı, tıklamalar arası 10 sn bekleme



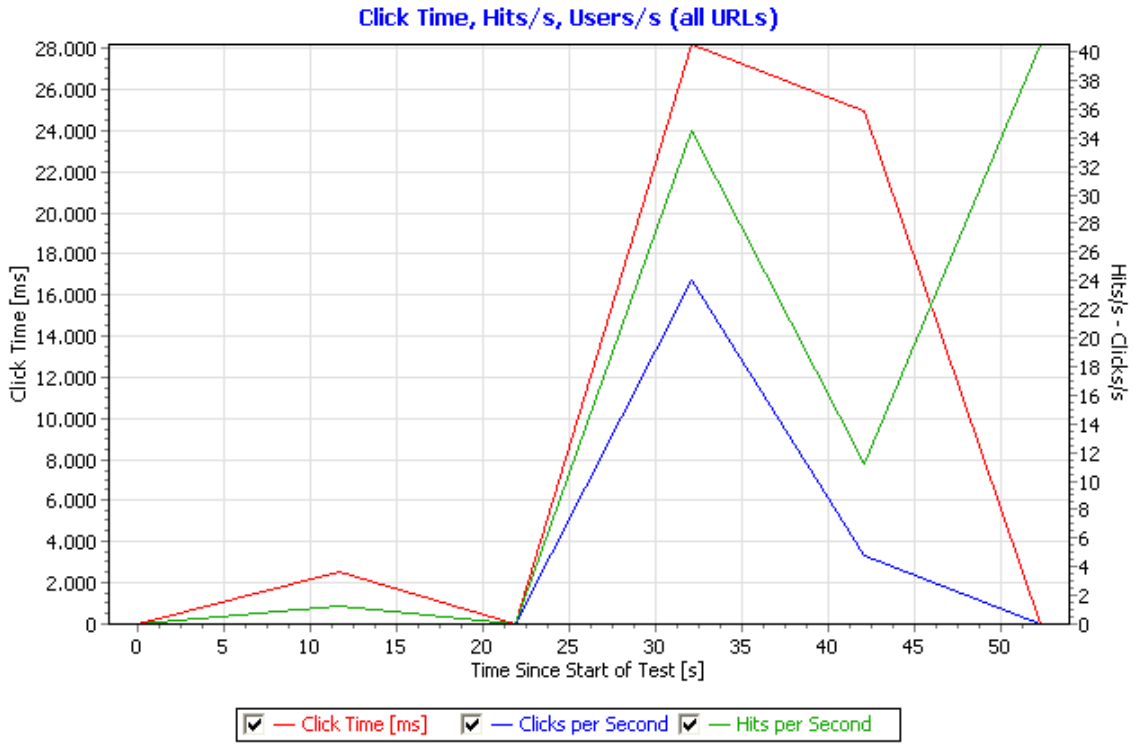
Şekil 7.6. Aynı anda 1000 kullanıcı, tıklamalar arası 10 sn bekleme

Aynı anda 1500 kullanıcı, tıklamalar arası 10 sn bekleme



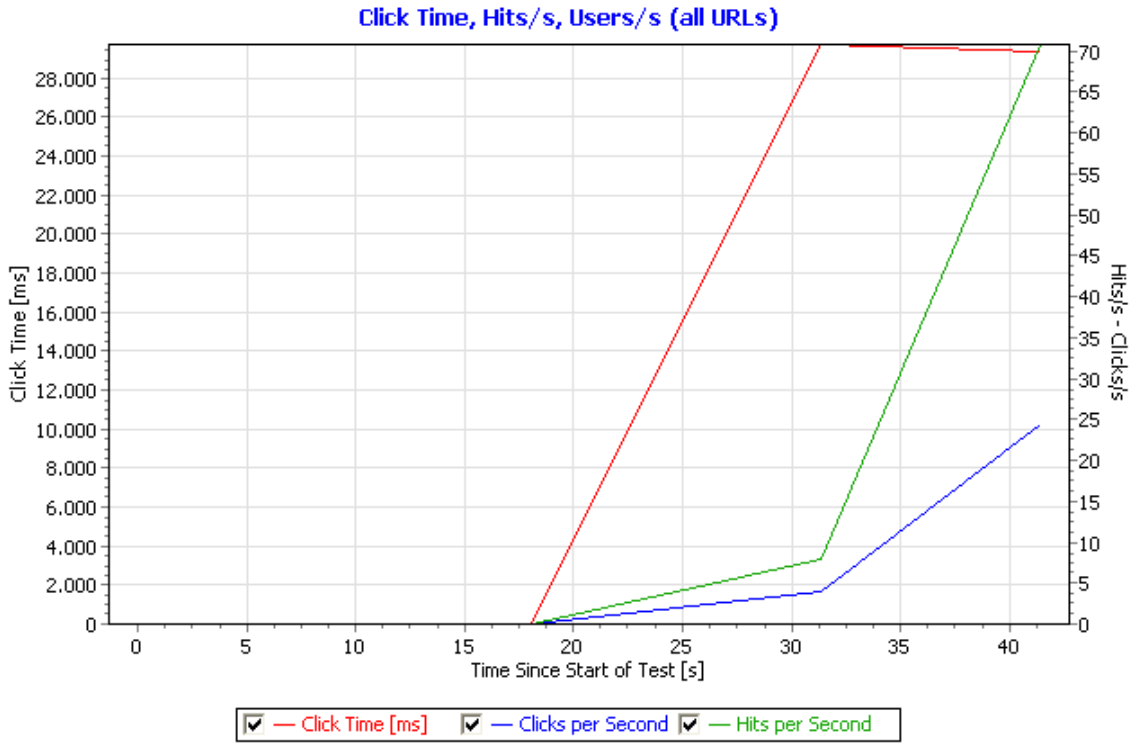
Şekil 7.7. Aynı anda 1500 kullanıcı, tıklamalar arası 10 sn bekleme

Aynı anda 2000 kullanıcı, tıklamalar arası 10 sn bekleme



Şekil 7.8. Aynı anda 2000 kullanıcı, tıklamalar arası 10 sn bekleme

Aynı anda 2500 kullanıcı, tıklamalar arası 10 sn bekleme



Şekil 7.9. Aynı anda 2500 kullanıcı, tıklamalar arası 10 sn bekleme

Test Sonuçlarının Değerlendirilmesi: Test raporlarına ait grafikler incelendiğinde görülebileceği gibi kullanıcı sanal kullanıcı sayısı arttırıldığında sayfanın ekrana getirilme süresi artmaktadır. Getirilme süresi arttığı gibi sayfanın ekrana getirilme süresinin uzamasıda test başlangıcı baz alındığında daha önce gerçekleşmektedir.

SONUÇLAR VE TARTIŞMA

Kümeler yüksek hesaplama gücü, yük dengeleme ve hata toleransı sağlayan ve düğüm adı verilen bilgisayarlardan oluşan paralel işlem yapan süper bilgisayarlardır. Kümeler farklı durumlarda farklı problemlere çözüm getirmek için kullanılabilir. Yüksek Performanslı İşlem, Hata Toleransı, Yük Dengeleme ve Yüksek Erişilebilirlik küme yapılarının kullanım amaçlarından bazılarıdır. Linux Küme yapılarını oluşturan düğümler için özel sunucu veya iş istasyonu gibi yüksek maliyetli bileşenler kullanılması zorunlu değildir. Kümelerin amacı sunulmakta olunan servislerin iyileştirilmesidir.

Tez kapsamında kurulan küme yapısı Hata Toleranslı ve Yük Dengelemeli küme yapılarına örnektir. Küme yapısı kararlı bir uygulama mimarisi olan Linux Virtual Server üzerinde çalışmaktadır. Aynı küme yapısı bir yönetici düğüm ilavesi ve yapılacak konfigürasyonla Yüksek Erişilebilir küme haline dönüştürülebilir. Tez kapsamında kurulan küme yapısının benzer çözümlere göre birçok avantajı bulunmaktadır. Genişleyebilirlik, kararlılık ve maliyet gibi. Örneğin yük dengeleyici olarak satılan donanımsal bir ürün ileride yetersiz kalacak ve kullanılamaz hale gelecektir. Yeni bir maliyet ortaya çıkacaktır. Küme yapılarında ise daha fazla performans için yapılması gereken tek şey kümeye yeni düğümler ilave etmektir.

Küme yapısı sayesinde olası bir sunucu arızası durumunda servislerde yaşanacak duruşların önüne geçilmiş ve yüksek çalışabilirlik elde edilmiştir. Küme yapısında bulunan Apache sunucu düğümlerinden birisinde yaşanacak sorun sadece o düğümü etkileyecek küme servislerinde kesintiye neden olmayacaktır. İki düğümde ayakta iken yük dengelemede yapılacak ve yüksek performans elde edilecektir.

KAYNAKLAR

[1] W. Gropp, E. Lusk, and T. Sterling, *Beowulf Cluster Computing with Linux,* The MIT Press, 2nd edition, 2003.

[2] K. Kopper, *The Linux Enterprise Cluster—Build a Highly Available Cluster with Commodity Hardware and Free Software,* No Starch Press, 1st edition, 2005.

[3] C. Kopparapu, *Load Balancing Servers, Firewalls, and Caches,* Wiley Computer Publishing, 1st edition, 2002.

[4] S. V. Vugt, *The Definitive Guide to SUSE Linux Enterprise Server,* Apress, 1st edition, 2006.

[5] http://www.linuxvirtualserver.org

[6] http://www.ultramonkey.org/papers/lvs_tutorial/html/

[7] http://fedoraproject.org

[8] http://www.kernel.org

[9] http://httpd.apache.org/

ÖZGEÇMİŞ

Gürkan TUNA, 25 Ocak 1975 tarihinde İstanbul'da doğdu. İlk ve Orta öğrenimini Uzunköprü'de, Lise öğrenimini ise Edirne'de tamamladıktan sonra 1993 yılında Yıldız Teknik Üniversitesi Bilgisayar Bilimleri Mühendisliği bölümünü kazandı. 1999 yılında mezun oldu. Üniversite 2. sınıftan başlayarak yaklaşık olarak 8 yıl özel sektörde çeşitli kademelerde çalıştı. Askerlik görevini 2001 yılında Diyarbakır 2. Taktik Hava Kuvvetleri Komutanlığında Teğmen rütbesiyle tamamladı. 2005 yılı Kasım ayında Trakya Üniversitesi Edirne Meslek Yüksekokulunda Öğretim Görevlisi olarak çalışmaya başladı. Halen Trakya Üniversitesi Edirne Teknik Bilimler Meslek Yüksekokulu'nda Bilgisayar Teknolojisi ve Programlama Bölümünde Öğretim Görevlisi olarak çalışmaktadır. Yüksek Lisans Tez aşamasındadır.