Carlos Eduardo Tavares Terra

Implantação de um serviço de Balanceamento de Carga utilizando LVS/Piranha

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2011

Carlos Eduardo Tavares Terra

Implantação de um serviço de Balanceamento de Carga utilizando LVS/Piranha

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *30 de Abril de 2011*

Prof. Tales Heimfarth

Prof. Arlindo Follador Neto

Prof. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil
2011

Dedico esta monografia a meus pais, minha esposa e meu filho que ainda está para nascer.

Agradecimentos

Agradeço a minha esposa, meus pais, familiares e amigos que diretamente ou indiretamente me apoiaram no desenvolvimento deste trabalho. Aos professores do curso ARL pelos ensinamentos compartilhados e em especial ao Prof. Joaquim pelo direcionamento.

Sumário

Lista de Figuras

Resumo

Os serviços de Tecnologia da Informação se tornaram uma parte vital da maioria das empresas no mundo, aumentando sua competitividade no mercado. Devido a isso, exigências quanto a qualidade nos serviços de TI aumentaram, forçando a criação de soluções que garantissem a disponibilidade e escalabilidade que as empresas necessitam para continuar crescendo de forma competitiva. Neste trabalho é apresentada uma solução de balanceamento de carga e alta disponibilidade que foi utilizada em uma empresa para garantir que o serviço de impressão permanecesse em pleno funcionamento, mesmo em momentos de manutenção. A solução adotada foi a utilização do Linux Virtual Server em conjunto com o Piranha.

Palavras-Chave: Load Balancing; LVS; Linux Virtual Server; Balanceamento de Carga; Piranha

Capítulo 1

Introdução

1.1 Considerações Iniciais

Os serviços de Tecnologia da Informação se tornaram uma parte vital da maioria das empresas no mundo. A facilidade para acessar e manipular as informações aumentaram a eficiência das empresas, aumentando sua competitividade no mercado.

Devido ao fato da TI ter se tornado uma parte fundamental de inúmeras empresas, a dependência por seus serviços se tornou ainda mais perceptível, exigindo dos profissionais que soluções de alta disponibilidade e alta escalabilidade fossem adotadas, garantindo uma prestação de serviço de qualidade.

1.2 Motivação

A motivação do desenvolvimento deste trabalho surgiu da necessidade de garantir alta disponibilidade, associada a escalabilidade, do serviço de impressão de uma das empresas onde trabalho. A utilização do serviço de impressão na empresa é extremamente alta e foi solicitada a criação de uma solução que garantisse que o serviço ficasse funcionando o mais próximo possível de 100% do tempo.

Devido a grande quantidade de impressoras existentes na empresa são necessários diversos servidores de impressão para processar os trabalhos enviados pelos usuários e encaminhar para as impressoras de rede. Cada servidor de impressão é responsável por processar um conjunto de filas de impressão diferentes.

Devido a este cenário, foi necessário implementar uma solução de serviço de impressão que tivesse alta disponibilidade e fosse escalável, podendo ter vários servidores processando os trabalhos. Agregado a isso, deveria ser implementada

uma solução que permitisse indisponibilizar um servidor para manutenção sem impactar o serviço prestado.

A solução adotada para este cenário é a utilização de um balanceador de carga, distribuindo os trabalhos de impressão entre todos os servidores pertencentes ao *cluster*.

1.3 Objetivos

O objetivo principal deste trabalho é apresentar a solução de balanceamento de carga adotada para garantir alta disponilidade associada à alta escalabilidade no serviço de impressão da empresa, utilizando *software* livre.

Para atingir o objetivo principal, este trabalho apresentará os conceitos relacionados à *cluster*, focando nos *clusters* de balanceamento de carga e suas vantagens em relação ao cenário apresentado.

Também será apresentada a ferramenta livre "Linux Virtual Server", *software* responsável pelo funcionamento do *cluster*, e as configurações utilizadas para pleno funcionamento da solução, balanceando a carga de todos os serviços delegados a ele. Apresentaremos também a solução Piranha, que garante alta disponibilidade no próprio servidor de balanceamento de carga.

Aproveitando o ambiente criado para o serviço de impressão, também serão apresentados cenários configurando o balanceamento de carga nos serviços de diretórios LDAP e DNS, que podem compartilhar do mesmo servidor de balanceamento, sem impactar os outros serviços hospedados.

1.4 Metodologia

A solução adotada utilizou uma arquitetura composta por dois servidores de balanceamento de carga, com uma solução de *failover*, garantindo com isso alta disponibilidade do próprio servidor de balanceamento. Um deles atua como nó ativo, enquanto que o outro atua como nó de *backup*.

Os serviços, por sua vez, são hospedados em outros dois servidores, conhecidos como servidores reais ou servidores de *backend*. Caso dois servidores reais não sejam suficientes, mais servidores podem ser adicionados.

A solução utilizando LVS permite utilizar diferentes técnicas de balanceamento. Neste trabalho foi adotada uma solução utilizando a técnica de roteamento direto (*direct routing*). Para isto, os servidores de *backend* e os servidores de balanceamento precisam, obrigatoriamente, estar no mesmo segmento de rede e os clientes devem conseguir se comunicar com todos os servidores. Logo, os servidores não podem estar sob nenhum tipo de mascaramento.

Para integrar a solução de balanceamento de carga com a alta disponibilidade dos servidores de balanceamento, foi utilizado o *software* Piranha.

1.5 Estrutura do trabalho

No Capítulo 2 é feito um levantamento dos pontos chave para o entendimento dos principais conceitos abordados no trabalho através de uma revisão da literatura. Nesee capítulo, são abordadas algumas questões sobre balanceamento de carga e seus principais benefícios.

O Capítulo 3 apresenta o desenvolvimento do trabalho propriamente dito. Ele está dividido em Seções que discriminam os materiais utilizados e as fases de desenvolvimento, apontando as estratégias de implementação adotadas e as principais dificuldades encontradas.

No Capítulo 4 são apresentados os resultados obtidos no trabalho bem como uma análise crítica dos mesmos.

Finalmente, no Capítulo 5 é feita uma síntese sobre a importância do trabalho, relacionando-a com os resultados obtidos. Além disso, são feitas propostas para trabalhos futuros na mesma área de concentração.

Capítulo 2

Referencial Teórico

2.1 *Cluster*

Segundo Gropp, Lusk e Sterling (2003), *cluster* é um computador paralelo construído com componentes e que rode *software*. Um *cluster* é composto de nós, cada um contendo um ou mais processadores, memória que é compartilhada por todos os processadores do nó e dispositivos periféricos adicionais, conectados a uma rede que permita que os dados transitem entre os nós.

De acordo com Sloan (2004), "*clusters*" e "computação de alto desempenho" eram sinônimos. Hoje o significado de "*cluster*" expandiu, deixando de ser apenas alto desempenho, incluindo *clusters* de alta disponibilidade (HA) e *clusters* de balanceamento de carga (LB). Entretanto estes tipos de *clusters* não são excludentes entre si. As subseções a seguir apresentam os principais tipos de *clusters* em uso atualmente.

2.1.1 *Cluster* de Alto Desempenho

Este tipo de *cluster* também é conhecido como *Cluster* de Alta Performance, pois tem como objetivo paralelizar o processamento de um determinado trabalho entre diversos processadores, de diferentes computadores, fazendo com que a capacidade de processamento seja escalável. Tem como principal objetivo aumentar a capacidade de processamento.

2.1.2 *Cluster* de Alta Disponibilidade

Clusters de alta disponibilidade, também conhecidos como *High Avaiability Clusters* e *failover clusters*, são normalmente utilizados em aplicações de missão crítica.

Um *cluster* de alta disponibilidade é formado por múltiplos computadores que podem prover um serviço apropriado. Um único computador deste *cluster* estará disponível para os usuários, enquanto que os outros computadores ficam em espera, monitorando o computador que está provendo o serviço. Em caso de falha no computador primário, um computador secundário passará a prover os serviços.

2.1.3 *Cluster* de Balanceamento de Carga

O principal objetivo de um *cluster* de balanceamento de carga, também conhecido como *Load-balancing cluster*, é conseguir melhor performance dividindo o trabalho entre múltiplos computadores.

Todos os computadores do *cluster* permanecem ativos, recebendo as requisições e executando os trabalhos de forma paralela. Devido a isso, a falha de um dos nós não causa interrupção do serviço.

2.2 Linux Virtual Server

A solução Linux Virtual Server[1] (LVS) utiliza um software de *cluster* de balanceamento de carga chamado de IP Virtual Server (IPVS). Segundo Kopper (2005), o software IPVS é uma coleção de *patches* de *kernel* que foram incorporadas ao *kernel* do Linux, começando na versão 2.4.23. Quando combinado com as funcionalidades de roteamento do *kernel* e filtragem de pacotes (*package-filtering*), o *kernel* com IPVS habilitado permite transformar qualquer computador rodando Linux em um balanceador de carga. Juntos, o balanceador e os nós pertencentes ao *cluster* são chamados de Linux Virtual Server (LVS).

2.2.1 LVS Director

O LVS Director, também conhecido como balanceador de carga do *cluster* ou simplesmente Director, é o responsável por receber todas as requisições dos computadores clientes e decide qual nó do *cluster* irá responder à requisição.

O Director é o coração do LVS, portanto seu serviço se torna crucial para funcionamento de todo o *cluster*. Desta forma, costuma-se criar um *cluster* de alta disponibilidade para o serviço do Director, criando o LVS Director ativo e o LVS Director *backup*.

Em caso de falha no Director ativo, o servidor *backup* assume a responsabilidade pelo balanceamento de carga do *cluster*.

[1] LVS: `http://www.linuxvirtualserver.org`

2.2.2 Servidores Reais

De acordo com Dolbier *et al.* (2003), servidor real é o nome dado aos computadores em um *cluster* LVS que realmente executam os serviços virtualizados. Servidores reais são agrupados por serviços que são virtualizados em um servidor virtual.

Os servidores reais também são conhecidos como servidores de *backend*, por estarem atrás dos servidores de balanceamento.

2.2.3 Servidores Virtuais

Segundo Bookman (2003), o servidor virtual é o Director, que repassa as requisições para um ou mais servidores reais.

A configuração de cada servidor virtual contém características específicas, como, por exemplo, endereço IP virtual, algoritmo de agendamento de pacotes para distribuição da carga, interface virtual utilizada no Director e quais os servidores reais compõem o *cluster*.

Cada servidor virtual representa um *cluster* de um serviço, que recebe conexões por uma porta TCP ou UDP.

2.2.4 Nomenclatura dos endereços IP do LVS

Virtual IP (VIP) Address

Este é o endereço IP utilizado pelo servidor virtual que os clientes utilizam para se comunicar com o serviço. Este endereço IP deve ser acessível pelos clientes para que os mesmos possam se comunicar com o *cluster*.

Um Director pode possuir um ou mais endereços VIPs, variando de acordo com a quantidade de servidores virtuais configurados nele.

Real IP (RIP) Address

Este é o endereço IP utilizado nos servidores reais, para comunicação com o Director, ou com os clientes, dependendo da solução de balanceamento adotada.

Director IP (DIP) Address

É o endereço IP configurado na interface de rede do Director. É utilizado para comunicação entre o Director e os servidores reais.

O Director recebe as requisições de clientes pelo endereço VIP e encaminha as requisições utilizando o endereço DIP.

Client IP (CIP) Address

Endereço IP configurado na interface de rede do computador cliente. Este endereço IP deve se comunicar com os endereços VIP.

2.2.5 Tipos de *cluster* LVS

Segundo Kopper (2005), os *clusters* LVS são normalmente descritos pelo tipo de método de encaminhamento utilizado pelo Director para reencaminhar as requisições para os nós do *cluster*.

Network Address Translation (LVS-NAT)

Nesta topologia, conforme Figura 2.1, o Director atua como um roteador, tendo duas interfaces de rede. Em uma das interfaces é configurado um endereço VIP público e na outra interface um endereço VIP privado.

O Director é responsável por concentrar todo o tráfego de entrada e saída, criando um mascaramento entre os clientes e os servidores reais. De acordo com Kopper (2005), este é o método mais simples de ser implementado.

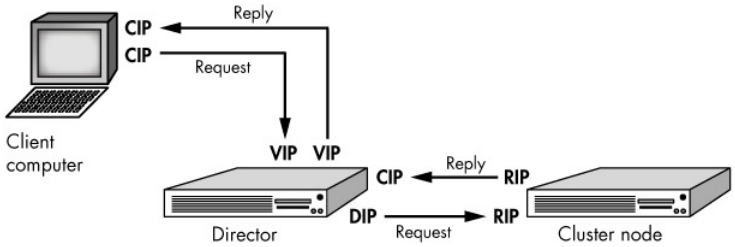

Figura 2.1: LVS utilizando NAT – Fonte: (KOPPER, 2005)

Nesta topologia o Director pode vir a ser um gargalo, caso o tráfego de rede seja muito intenso e seja necessário que o mesmo mantenha uma tabela muito grande de rastreamento de conexões.

Direct Routing (LVS-DR)

Segundo Kopper (2005), este é o melhor método de encaminhamento para *clusters* corporativos. Nesta topologia o Director recebe as requisições e encaminha para o servidor real selecionado, que responde diretamente para o cliente, conforme Figura 2.2

Este método consiste em mudanças no cabeçalho do quadro de dados, conforme Figura 2.3. O Director recebe o pacote com destino ao endereço VIP, altera

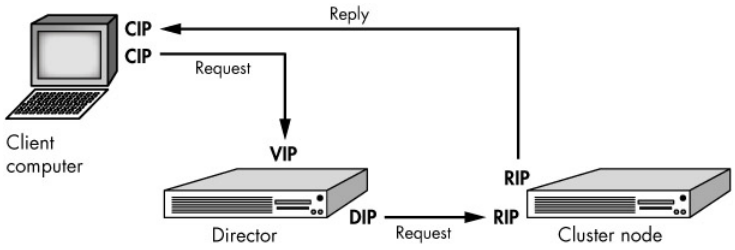

Figura 2.2: LVS utilizando Direct Routing – Fonte: (KOPPER, 2005)

o quadro adicionando o endereço MAC do servidor real e por final envia o novo pacote na rede.

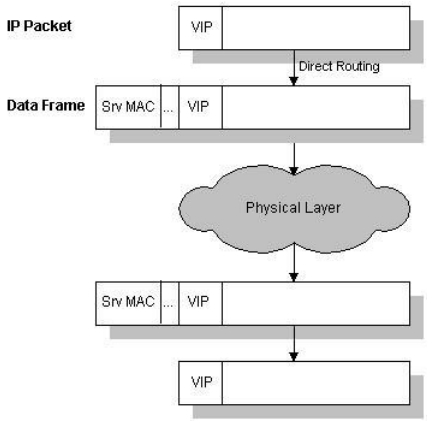

Figura 2.3: Direct Routing Workflow – Fonte: http://www.linuxvirtualserver.org/VS-DRouting.html

Para que este método funcione, os servidores reais e o Director precisam, necessariamente, estar no mesmo segmento de rede. Também se tornam necessárias algumas configurações especiais nos servidores reais.

Segundo Red Hat, Inc. (2009), não é recomendado utilizar o Director como um *gateway* para os servidores reais, pelo fato de adicionar complexidade desnecessária e carga de rede no servidor de balanceamento, que reintroduz o problema de gargalo de rede que existe no LVS-NAT.

IP Tunneling (LVS-TUN)

Esta topologia é similar ao Direct Routing, conforme Figura 2.4. Entretanto o Director e os servidores reais não precisam estar no mesmo segmento de rede, porém

é necessário criar um túnel IP entre os servidores reais e o Director. Com isso, apenas sistemas operacionais que tenham suporte ao protocolo de tunelamento IP podem ser servidores reais.

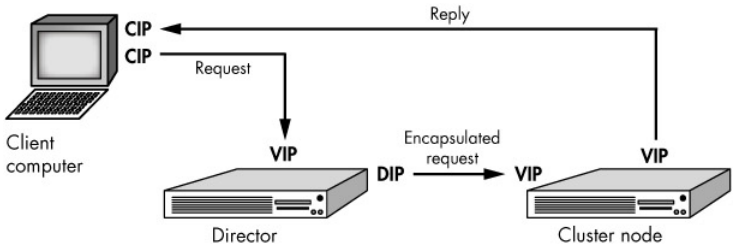

Figura 2.4: LVS utilizando IP Tunnel – Fonte: (KOPPER, 2005)

2.2.6 Algoritmos de balanceamento de carga

De acordo com Red Hat, Inc. (2009), uma das vantagens de se utilizar o LVS é a sua flexibilidade no balanceamento de carga em nível IP. Esta flexibilidade existe pela variedade de algoritmos de balanceamento que podem ser configurados nos servidores virtuais.

O LVS permite que o administrador decida qual o melhor algoritmo a ser utilizado em sua solução, variando com cada cenário que o administrador tenha em mãos. É possível distribuir igualmente a carga entre os servidores reais, mas também é possível utilizar pesos que consideram a capacidade do servidor e a carga de trabalho do mesmo.

Seguem-se os principais tipos de algoritmos de balanceamento de carga suportados pelo LVS. Além dos descritos a seguir, existem também os algoritmos Locality-Based Least-Connection Scheduling[2], Locality-Based Least-Connection Scheduling with Replication Scheduling[3], Destination Hash Scheduling[4] que foram elaborados para utilização em *clusters* de servidores *proxy-cache*, e também o algoritmo Source Hash Scheduling[5] que foi elaborado para ser utilizado em um *cluster* com múltiplos *firewalls*.

[2]Locality-Based Least-Connection Scheduling: `http://kb.linuxvirtualserver.org/wiki/Locality-Based_Least-Connection_Scheduling`

[3]Locality-Based Least-Connection Scheduling with Replication Scheduling: `http://kb.linuxvirtualserver.org/wiki/Locality-Based_Least-Connection_with_Replication_Scheduling`

[4]Destination Hash Scheduling: `http://kb.linuxvirtualserver.org/wiki/Destination_Hashing_Scheduling`

[5]Source Hash Scheduling: `http://kb.linuxvirtualserver.org/wiki/Source_Hashing_Scheduling`

Round-Robin Scheduling

Este algoritmo distribui a carga igualmente entre todos os servidores reais, ignorando diferenças de capacidade e carga.

Weighted Round-Robin Scheduling

Este algoritmo distribui a carga entre os servidores reais, de forma similar ao algoritmo anterior. Entretanto ele considera um peso definido pelo administrador. Um servidor real que tenha um peso maior, receberá um maior número de requisições. Desta forma é possível diferenciar servidores com diferentes capacidades. Segundo Red Hat, Inc. (2009), esta é a melhor escolha caso exista diferença grande de capacidade entre os servidores reais.

Least-Connection

Este algoritmo força que as requisições sejam redirecionadas sempre para o servidor real que possuir o menor número de conexões ativas.

Weighted Least-Connections

Este algoritmo é similar ao anterior, porém adiciona um peso fixado pelo administrador nos servidores reais. Desta forma é possível ter servidores com diferentes capacidades. Este é o algoritmo escolhido pelo autor na elaboração da solução apresentado neste trabalho e também é o algoritmo definido como padrão na distribuição Red Hat Enterprise Linux 5.

2.3 Solução Piranha para balanceamento de carga

Piranha[6] é um dos produtos de *clustering* da Red Hat Inc., que inclui o módulo IPVS do *kernel*, uma ferramenta de monitoração do *cluster* e uma interface *web* para configuração do *cluster*. Piranha é estritamente a implementação de um *software* de alta disponibilidade, com uma interface para administração.

A ferramenta de monitoração do Piranha tem a responsabilidade de verificar o *status* dos servidores de balanceamento ativo e *backup*, e também checar a disponibilidade dos serviços nos servidores reais.

De acordo com Wangsmo (), o Piranha é composto dos seguintes componentes:

- Código de *kernel* do IPVS

[6]Piranha: `http://www.redhat.com/software/rha/cluster/piranha/`

- *Daemon* do lvs que gerencia a tabela de roteamento utilizando a ferramenta "ipvsadm"
- *Daemon* nanny, reponsável por monitorar os servidores e serviços nos servidores reais do *cluster*
- *Daemon* pulse, responsável por controlar os outros *daemons* e chavear o LVS entre os Directors ativo e *backup* em caso de falha
- Ferramenta piranha GUI para gerenciar e administrar o ambiente do *cluster*

Por padrão todos os *daemons* apresentados utilizam o mesmo arquivo de configuração, localizado em */etc/sysconfig/ha/lvs.cf*. As principais funções do Piranha são editar o arquivo de configuração e controlar o serviço "pulse", iniciando e parando quando necessário.

2.3.1 Piranha GUI

Piranha GUI é uma interface *web* que permite manipular o arquivo de configuração */etc/sysconfig/ha/lvs.cf*, gerando toda a parametrização necessária para que os servidores virtuais e o próprio Director funcionem corretamente.

A Figura 2.5 apresenta a tela de controle do LVS. Nela é possível saber o *status* do serviço LVS, quais processos estão em execução e verificar como está a tabela de roteamento IPVS.

A Figura 2.6 apresenta a tela de configurações globais do LVS, onde se definem os endereços IP que serão utilizados pelo Director. Nem todos os campos precisam ser preenchidos, variando de acordo com tipo de topologia escolhido.

Já a Figura 2.7 contém os parâmetros para configuração do Director *backup* informando se a solução adotada possui ou não um Director *backup*. Caso o *backup* não seja configurado, não haverá um servidor assumindo o serviço de balanceamento de carga caso o servidor de balanceamento ativo venha a falhar. É extremamente recomendável que se tenha um *backup* para garantir que o serviço esteja sempre no ar.

As Figuras 2.8 e 2.9 apresentam as telas de configuração dos servidores virtuais. Nesta página é possível configurar todos os parâmetros necessários para que o servidor virtual funcione corretamente. Os principais parâmetros de configuração do servidor virtual são:

- **Name** - Nome do servidor virtual. Utilizado apenas para referência.
- **Application Port** - Porta de comunicação utilizada pela aplicação.

- **Protocol** - Protocolo utilizado para comunicar com a porta configurada anteriormente. Pode ser TCP ou UDP.
- **Virtual IP Address** - Endereço VIP utilizado pelo servidor virtual para receber requisições. Este endereço pode ser utilizado por mais de um servidor virtual.
- **Virtual IP Net Mask** - Máscara de rede do endereço VIP.
- **Device** - Dispositivo de rede onde será configurado o endereço VIP. Normalmente é uma interface de rede virtual, conhecida como *alias*, sobre uma interface de rede física.
- **Scheduling** - Algoritmo de balanceamento de carga utilizado neste servidor virtual.

As Figuras 2.10 e 2.11 apresentam as telas de configuração dos servidores reais, de um determinado servidor virtual. Nesta página é possível configurar os detalhes específicos de cada servidor real, definindo o endereço RIP e o peso associado ao servidor real. Quanto maior o peso, maior a proporção de carga enviada ao servidor.

A Figura 2.12 apresenta um exemplo de arquivo de configuração gerado pelo piranha GUI.

A utilização da interface *web* piranha GUI é de uso opcional, mas é recomendada sua utilização durante a elaboração do primeiro arquivo de configuração. Após a primeira configuração, a edição do arquivo se torna mais simples.

2.3.2 Verificação de disponibilidade dos servidores reais

A verificação da disponibilidade dos serviços nos servidores reais pode ser feita de três formas:

- Verificando se a porta de comunicação TCP está aberta e escutando no servidor real
- Enviando uma requisição para o serviço no servidor real e comparando a resposta obtida
- Executando um *script* e comparando o resultado, como o *script* da Figura 2.13

A Figura 2.14 apresenta a tela de configuração da forma de monitoração dos servidores reais.

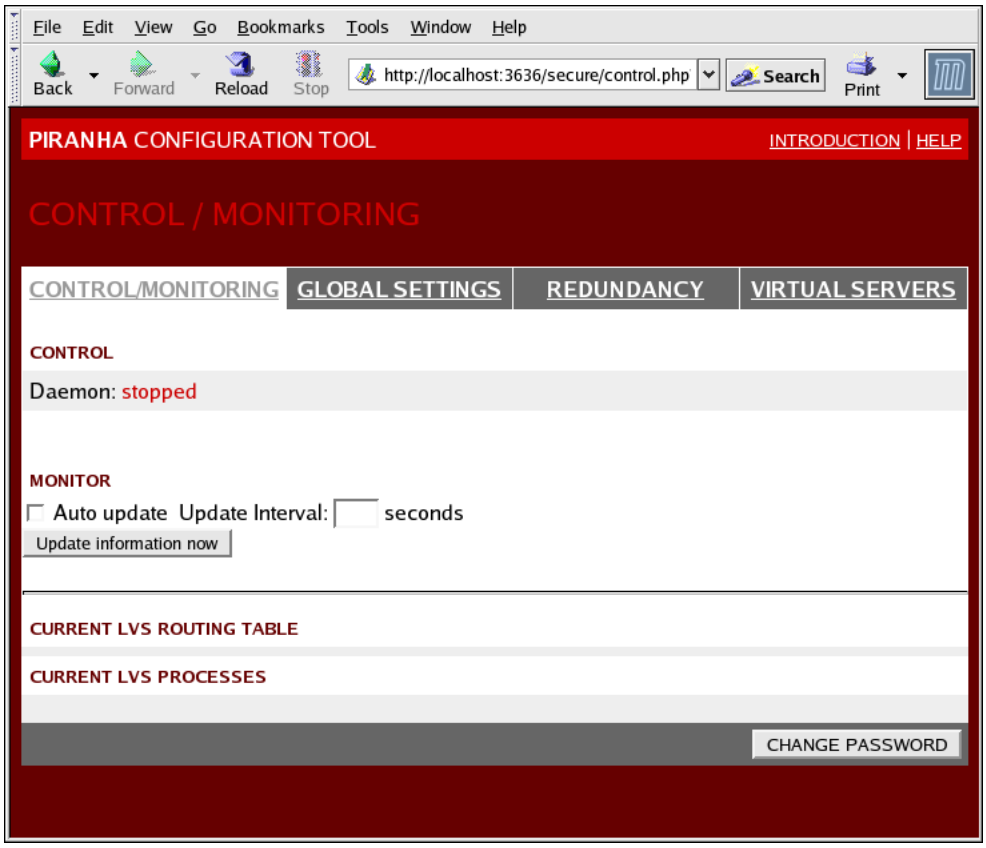

Figura 2.5: Página de controle do LVS no piranha GUI – Fonte: (RED HAT, INC., 2009)

Quando um serviço deixa de responder no servidor real, o *daemon* nanny automaticamente desativa o servidor real na tabela de roteamento do IPVS, fazendo com que o LVS não envie novas requisições a este servidor real até que o mesmo seja reativado.

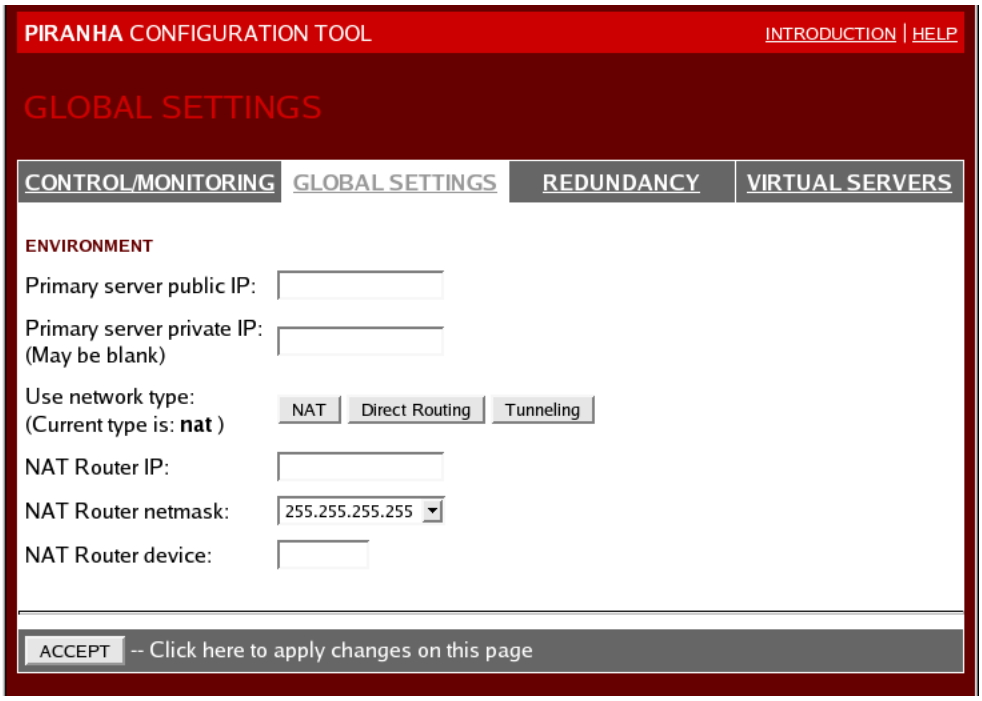

Figura 2.6: Página de configurações globais do piranha GUI – Fonte: (RED HAT, INC., 2009)

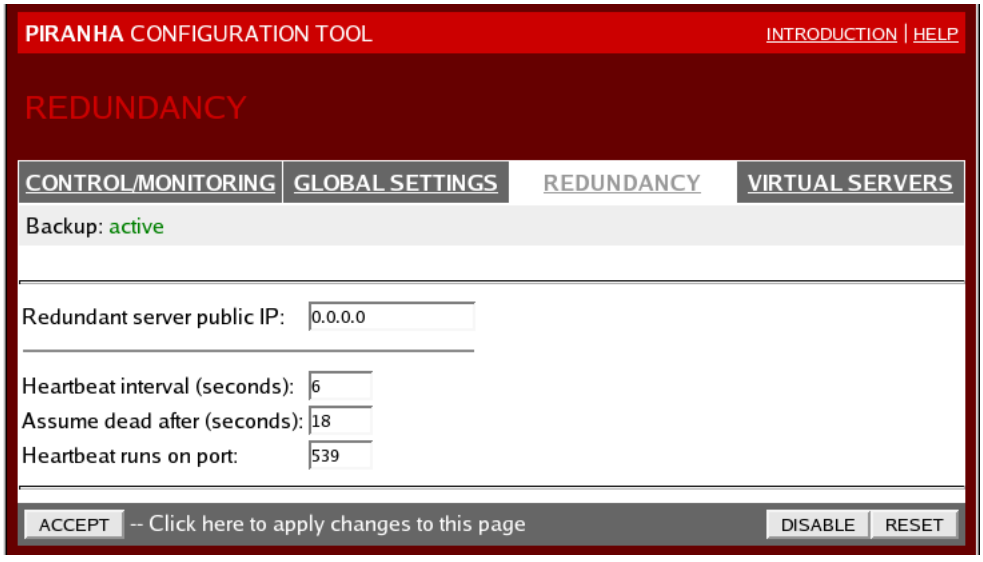

Figura 2.7: Página de configuração do Director backup no piranha GUI – Fonte: (RED HAT, INC., 2009)

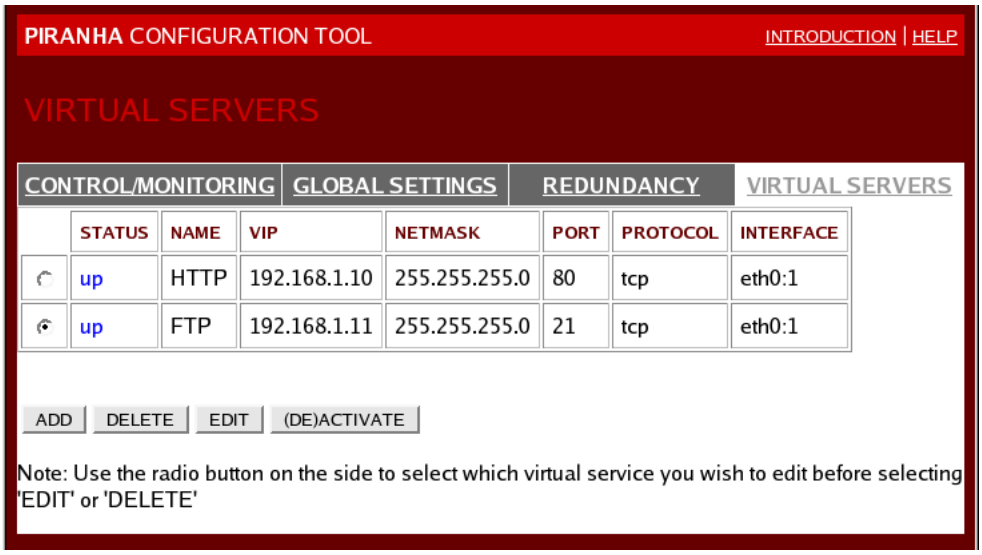

Figura 2.8: Configuração dos servidores virtuais no piranha GUI – Fonte: (RED HAT, INC., 2009)

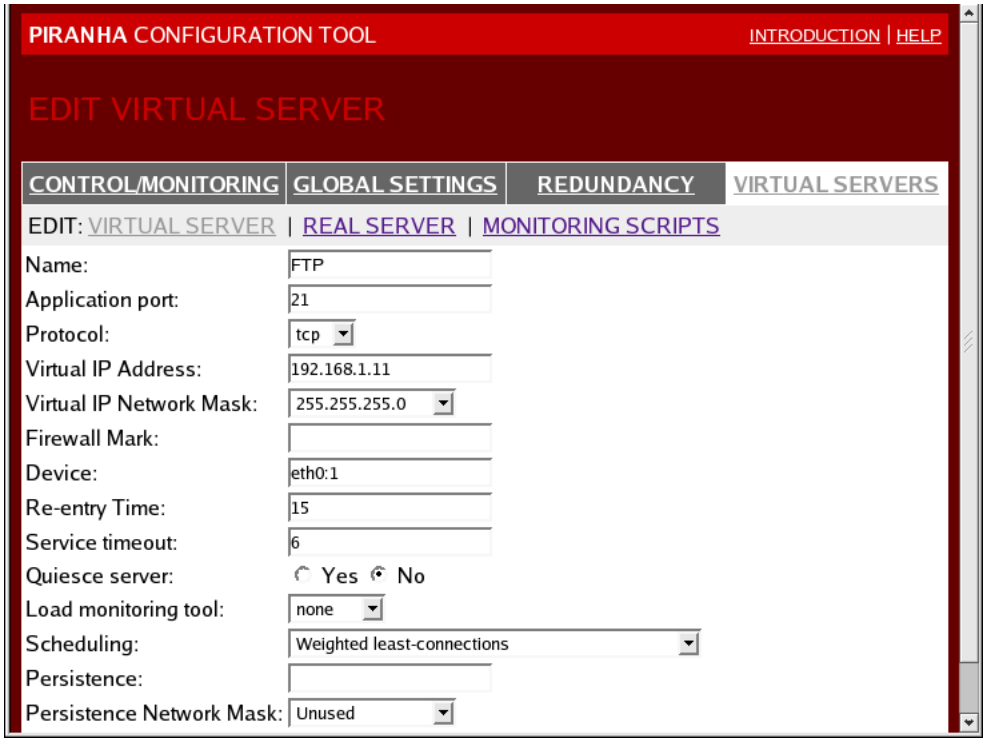

Figura 2.9: Configuração dos servidores virtuais no piranha GUI – Fonte: (RED HAT, INC., 2009)

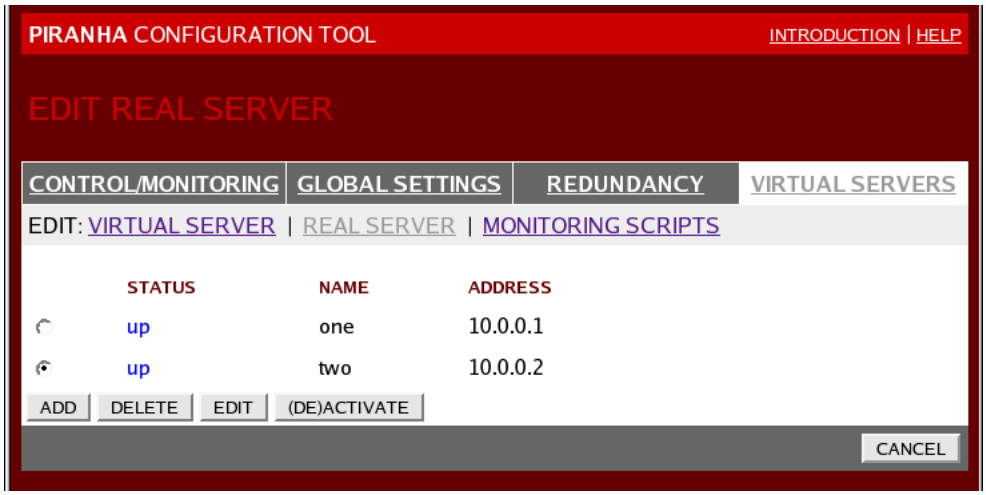

Figura 2.10: Configuração dos servidores reais no piranha GUI – Fonte: (RED HAT, INC., 2009)

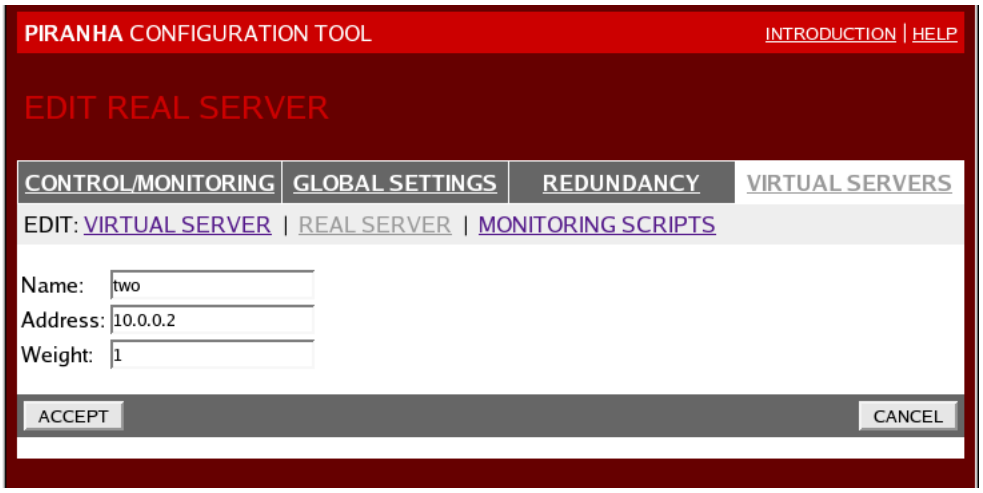

Figura 2.11: Configuração dos servidores reais no piranha GUI – Fonte: (RED HAT, INC., 2009)

```
primary = 10.23.8.1
service = lvs
rsh_command = rsh
backup_active = 1
backup = 10.23.8.2
heartbeat = 1
heartbeat_port = 539
keepalive = 4
deadtime = 12
network = nat
nat_router = 172.18.1.254 eth1:0
nat_nmask = 255.255.255.0
reservation_conflict_action = preempt
debug_level = NONE
virtual web {
     active = 1
     address = 10.23.8.80 eth0:1
     vip_nmask = 255.255.255.255
     port = 80
     persistent = 600
     send = "GET / HTTP/1.0\r\n\r\n"
     expect = "HTTP"
     load_monitor = none
     scheduler = wlc
     protocol = tcp
     timeout = 6
     reentry = 15
     quiesce_server = 0
     server webserver1 {
         address = 172.18.1.11
         active = 1
         weight = 100
     }
     server webserver2 {
         address = 172.18.1.12
         active = 1
         weight = 100
     }
     server webserver3 {
         address = 172.18.1.13
         active = 1
         weight = 100
     }
}
```

Figura 2.12: Exemplo de lvs.cf – Fonte: http://www.linuxvirtualserver.org/docs/ha/piranha.html

```
#!/bin/sh
TEST=`dig -t soa example.com @\$1 | grep -c dns.example.com
if [ \$TEST != "1" ]; then
        echo "OK
else
        echo "FAIL"
fi
```

Figura 2.13: *Script* de verificação de serviço DNS – Fonte: (RED HAT, INC., 2009)

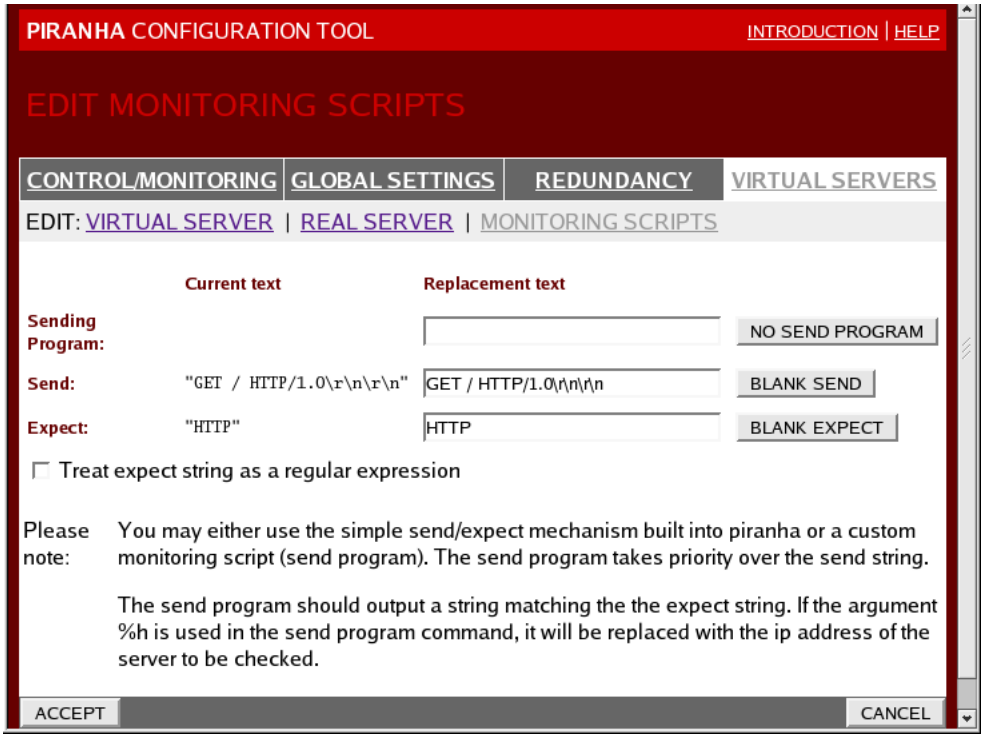

Figura 2.14: Configuração da monitoração dos servidores reais no piranha GUI – Fonte: (RED HAT, INC., 2009)

Capítulo 3

Metodologia

3.1 Comentários iniciais

A empresa foco deste trabalho tinha necessidade de garantir a disponibilidade e a escalabilidade do serviço de impressão, evitando interrupções no serviço por falhas ou por necessidades de manutenção.

Dentre as soluções possíveis, a mais adequada pois tinha uma abrangência que cobria tanto disponibilidade quanto escalabilidade, foi o *cluster* de balanceamento de carga.

Para atender esta demanda poderiam ser utilizadas soluções proprietárias ou livres, utilizando *hardware* ou *software*. Soluções de *hardware* proprietário possuem um custo muito elevado, restando apenas soluções de *software*. Devido a isso, foi adotada a solução de balanceamento LVS em conjunto com a solução Piranha, que possui uma boa relação custo-benefício e é um *software* livre reconhecido e com suporte no mercado.

3.2 Cenário da Solução

Para implantar a solução de balanceamento de carga, foram utilizados quatro servidores com sistema operacional Red Hat Enterprise Linux 5. Dois deles servindo como Director, ativo e *backup*, e os outros dois como servidores reais.

Os servidores que atuam como Director estão com subscrições *Advanced Plataform* e com o canal "*cluster*" habilitado, permitindo instalação dos aplicativos necessários via rede.

Os dois servidores reais proveêm os serviços de impressão, LDAP[1] e DNS[2]. Cada serviço está configurado e funcionando em modo *standalone*. Os aplicativos utilizados para prover estes serviços são, especificamente, CUPS[3], OpenLDAP[4] e Bind[5], sendo que o CUPS aceita os procolos IPP[6] e LPD[7].

Todos os servidores reais possuem a mesma capacidade de processamento e memória.

3.2.1 Definição da infraestrutura de rede

A infraestrutura de rede foi planejada para que todos os servidores reais e o Director estivessem no mesmo segmento de rede e estivessem acessíveis pelos clientes. A definição dos endereços IP ficou da seguinte forma:

- Endereço DIP do Director ativo: 10.0.0.1/24
- Endereço DIP do Director *backup*: 10.0.0.2/24
- Endereço RIP do primeiro servidor real: 10.0.0.3/24
- Endereço RIP do segundo servidor real: 10.0.0.4/24
- Endereço VIP para o serviço de impressão (IPP e LPD): 10.0.0.5/24 (interface eth0:1)
- Endereço VIP para o serviço LDAP: 10.0.0.6/24 (interface eth0:2)
- Endereço VIP para o serviço DNS (TCP e UDP): 10.0.0.7/24 (interface eth0:3)
- *Gateway* de rede: 10.0.0.254

[1] Lightweight Directory Access Protocol - protocolo em nível de aplicação para leitura e edição de diretórios sobre uma rede IP. http://tools.ietf.org/html/rfc4510

[2] Domain Name System - protocolo de rede para resolução de nomes. http://www.ietf.org/rfc/rfc1035.txt

[3] http://www.cups.org

[4] http://www.openldap.org

[5] http://www.isc.org/software/bind

[6] Internet Printing Protocol - protocolo de rede padrão para impressão remota. http://tools.ietf.org/html/rfc3239

[7] Line Printer Daemon - protocolo de rede para impressão remota. http://www.ietf.org/rfc/rfc1179.txt

3.3 Instalação da solução Piranha

Na distribuição Red Hat Enterprise Linux, os pacotes "ipvsadm" e "piranha" são os responsáveis pelo funcionamento do balanceador de carga e a verificação de integridade entre os Directors ativo e *backup*. Eles devem ser instalados apenas nos Directors.

Para instalar estes aplicativos e suas dependências, basta utilizar o comando:

```
# yum -y install ipvsadm piranha
```

Para habilitar a interface *web* piranha GUI no Director ativo é necessário iniciar o serviço "piranha-gui" e configurá-lo para iniciar automaticamente durante o processo de inicialização. Após isso é necessário configurar a senha de acesso à interface. Este serviço não precisa ser configurado no Director *backup*

```
# service piranha-gui start
# chkconfig piranha-gui on
# piranha-passwd
```

Após o serviço estar iniciado e sua senha definida, é possível acessar a interface *web* utilizando um navegador de internet no endereço *http://10.0.0.1:3636*.

Nos servidores reais não é necessária a instalação de nenhum aplicativo adicional.

3.4 Configuração

As configurações montadas para criação do *cluster* de balanceamento de carga foram baseadas na documentação (RED HAT, INC., 2009) e (RED HAT, INC., 2008).

Foi adotada a topologia *Direct Routing* para comunicação entre o Director e os servidores reais, devido a isso, todos os servidores envolvidos foram configurados no mesmo segmento de rede.

3.4.1 Configuração dos Directors

A configuração dos Directors é bastante simples. Todos os parâmetros podem ser ajustados modificando o arquivo */etc/sysconfig/ha/lvs.cf* ou utilizando a interface *web* "piranha-gui".

A Figura 3.1 apresenta um trecho do arquivo de configuração que define os parâmetros globais de funcionamento do Director, configurando também a redundância entre o Director ativo e o *backup*. O arquivo de configuração deve ser o mesmo nos dois Directors para que a redundância funcione corretamente.

```
serial_no = 1
primary = 10.0.0.1
service = lvs
backup_active = 1
backup = 10.0.0.2
heartbeat = 1
heartbeat_port = 539
keepalive = 6
deadtime = 18
network = direct
debug_level = NONE
monitor_links = 1
syncdaemon = 1
```

Figura 3.1: Trecho do arquivo de configuração lvs.cf – Configurações Globais

O parâmetro "serial_no" é utilizado para validar se o arquivo "lvs.cf" está na mesma versão entre os Directors. Sempre que for feita alguma modificação no arquivo é recomendado incrementar o valor deste parâmetro.

Os parâmetros "primary" e "backup" definem os endereços IP de cada um dos Directors. O primeiro é o ativo e o segundo é o *backup*. Entretanto, somente informar o endereço IP do Director *backup* não garante a redundância. É importante ajustar o parâmetro "backup_active", que informa ao Piranha que é para colocar em prática o serviço de redundância dos Directors.

Outro parâmetro importante é o "network". Este define o tipo de topologia utilizada em nossa solução LVS. Foi adotada a topologia Direct Routing para comunicação entre o Director e os servidores reais, de acordo com a recomendação da Red Hat para ambientes corporativos, pois reduz a carga do LVS e não precisa de mascaramento. Devido a isso, todos os servidores envolvidos foram configurados no mesmo segmento de rede.

O manual do "lvs.cf", encontrado no Red Hat Enterprise Linux, possui maiores informações sobre os parâmetros não descritos e sobre outros não mencionados nesta configuração.

3.4.2 Configuração dos Servidores Virtuais

A configuração dos servidores virtuais também é feita no arquivo "lvs.cf". As Figuras 3.2 e 3.3 apresentam um trecho onde são configurados dois servidores virtuais chamados "ipp" e "lpd". Estes nomes servem apenas como referência e representam um serviço que está sendo balanceado.

Para cada serviço é necessário criar uma configuração de servidor virtual independente. Como os serviços IPP e LPD estão associados ao servidor de impressão e sua única diferença é a porta de comunicação, as configurações são bem similares, incluindo o endereço VIP e interface de rede.

Cada servidor virtual apresentado nas Figuras 3.2 e 3.3 possui dois servidores reais, identificados como “real1” e “real2”.

```
virtual ipp {
    active = 1
    address = 10.0.0.5 eth0:1
    vip_nmask = 255.255.255.0
    port = 631
    persistent = 60
    scheduler = wlc
    protocol = tcp
    load_monitor = none
    timeout = 6
    reentry = 15
    quiesce_server = 0
    server real1 {
        address = 10.0.0.3
        active = 1
        weight = 1
    }
    server real2 {
        address = 10.0.0.4
        active = 1
        weight = 1
    }
}
```

Figura 3.2: Trecho do arquivo de configuração lvs.cf – Configuração do Servidor Virtual – Parte 1

Dentre os parâmetros apresentados, os principais são “address”, “vip_nmask”, “scheduler”, “protocol” e “port”. Além de cada seção de configuração dos servidores reais, que possuem os parâmetros “address” e “weight”.

Na seção *virtual*, o parâmetro “address” configura o endereço VIP e a interface de rede onde é configurado o endereço para este servidor virtual. Este parâmetro pode se repetir entre diversos servidores virtuais, mas deve-se prestar bastante atenção para não configurar endereços VIP diferentes para a mesma interface em diferentes servidores virtuais. O parâmetro “vip_nmask” configura a máscara de rede do endereço VIP.

```
virtual lpd {
    active = 1
    address = 10.0.0.5 eth0:1
    vip_nmask = 255.255.255.0
    port = 515
    persistent = 60
    scheduler = wlc
    protocol = tcp
    load_monitor = none
    timeout = 6
    reentry = 15
    quiesce_server = 0
    server real1 {
        address = 10.0.0.3
        active = 1
        weight = 1
    }
    server real2 {
        address = 10.0.0.4
        active = 1
        weight = 1
    }
}
```

Figura 3.3: Trecho do arquivo de configuração lvs.cf – Configuração do Servidor Virtual – Parte 2

O parâmetro "scheduler" configura o algoritmo de balanceamento de carga. Conforme citado anteriormente, neste trabalho foi adotado o algoritmo Weighted Least-Connections, simbolizado na configuração como "wlc".

Os parâmetros "protocol" e "port" configuram a porta de rede que ficará na escuta aguardando as requisições dos clientes. No modo *Direct Routing* a porta utilizada aqui deve ser a mesma porta utilizada pelos serviços nos servidores reais.

O parâmetro "persistent" configura uma memória no Director. O mesmo armazenará um breve histório das requisições recebidas e para onde qual servidor real foram encaminhadas. Quando uma nova requisição, oriunda de um cliente já conhecido, for recebida, o Director consulta em sua memória e encaminha a requisição para o mesmo servidor real utilizado anteriormente. Isso é muito útil quando o serviço balanceado utiliza sessões que não são compartilhadas entre os servidores reais.

Na seção de configuração de cada servidor real, o parâmetro "address" configura o endereço RIP do servidor real e o parâmetro "weight" configura o peso dado ao servidor.

As Figuras 3.4 e 3.5 apresentam a configuração dos servidores virtuais que atendem ao serviço DNS. Foram criados dois servidores virtuais pelo fato do serviço DNS responder tanto pelo protocolo TCP quanto pelo UDP.

No exemplo do servidor virtual que utiliza o protocolo UDP foi necessário utilizar um aplicativo externo de verificação do *status* do serviço, pois não é possível verificar a conectibilidade de uma conexão UDP. Não há impedimentos quanto ao uso de aplicativos em servidores virtuais cujo protocolo seja TCP. O uso de aplicativos pode ter um resultado mais apurado do que apenas um teste de conectibilidade com a porta do serviço.

O parâmetro "send_program" configura o caminho do executável que faz a verificação de integridade do serviço, enquanto que o parâmetro "expect" configura a resposta esperada do executável. Caso o retorno seja diferente do informado no parâmetro "expect", o serviço passa a ser considerado como indisponível no servidor real. A variável interna "%h" é substituída pelo endereço RIP do servidor real sendo testado.

A Figura 3.6 apresenta o *script* utilizado para verificar a integridade do serviço DNS nos servidores reais.

Na Figura 3.7 é apresentada a configuração do último servidor virtual. Este servidor virtual representa o serviço LDAP e não tem grandes diferenças em relação aos outros servidores virtuais.

3.4.3 Configuração dos Servidores Reais

Excluindo a configuração dos serviços balanceados, é necessário configurar alguns detalhes em regras de firewall para que o método *Direct Routing* funcione.

Conforme explicado anteriormente, no método *Direct Routing* o Director recebe as requisições e recoloca o pacote na rede, alterando apenas o campo no cabeçalho do quadro, adicionando o endereço MAC do servidor real que atenderá realmente à requisição. Entretanto, como o endereço VIP não está configurado no servidor real, o mesmo descarta o pacote.

De acordo com Red Hat, Inc. (2009), existem duas solução para resolver este problema. A primeira utilizando "arptables_jf" e a outra utilizando "iptables". Neste trabalho foi adotada a solução utilizando "iptables", pelo fato desta ferramenta estar disponível na instalação padrão do Red Hat Enterprise Linux e pela configuração ser mais simples.

Em ambas as soluções tem-se o objetivo de fazer o servidor real reconhecer um pacote cujo destino seja o endereço VIP. É possível configurar uma interface virtual em cada servidor real, configurando em todos eles o endereço VIP. Entretanto o *gateway* poderia ficar perdido, pois haveria mais de um servidor respondendo às

```
virtual dns {
    active = 1
    address = 10.0.0.7 eth0:3
    vip_nmask = 255.255.255.0
    port = 53
    persistent = 60
    expect = "OK"
    send_program = "/root/scripts/checkdns \%h"
    load_monitor = none
    scheduler = wlc
    protocol = udp
    timeout = 6
    reentry = 15
    quiesce_server = 0
    server real1 {
        address = 10.0.0.3
        active = 1
        weight = 1
    }
    server real2 {
        address = 10.0.0.4
        active = 1
        weight = 1
    }
}
```

Figura 3.4: Trecho do arquivo de configuração lvs.cf – Configuração do Servidor Virtual DNS – Parte 1

requisições ARP para o endereço VIP na rede e com isso começar a enviar todas as requisições para algum dos servidores reais ao invés do Director.

Na solução utilizando "arptables_jf" é utilizado este método, porém é criada uma regra de *firewall* em nível ARP impedindo que o servidor real responda as requisições ARP enviadas para o endereço VIP. Desta forma ele reconhece os pacotes destinados ao VIP, mas não se apresenta na rede como respondendo por este endereço.

Na solução utilizando "iptables", entretanto, não se necessita que o servidor real possua uma interface virtual com o endereço VIP configurado. É criada uma regra de *firewall* redirecionando todo o tráfego com destino ao endereço VIP para o próprio servidor, utilizando o módulo "ipt_REDIRECT".

Em nossa solução utilizamos a regra de *firewall* conforme Figura 3.8, adaptando apenas os endereços VIP, o protocolo e as portas utilizadas, conforme Figura 3.9.

```
virtual dns_tcp {
    active = 1
    address = 10.0.0.7 eth0:3
    vip_nmask = 255.255.255.0
    port = 53
    persistent = 60
    load_monitor = none
    scheduler = wlc
    protocol = tcp
    timeout = 6
    reentry = 15
    quiesce_server = 0
    server real1 {
        address = 10.0.0.3
        active = 1
        weight = 1
    }
    server real2 {
        address = 10.0.0.4
        active = 1
        weight = 1
    }
}
```

Figura 3.5: Trecho do arquivo de configuração lvs.cf – Configuração do Servidor Virtual DNS – Parte 2

```
#!/bin/bash
TEST=`dig -t soa google.com +time=1 @\$1 | grep -c ns1.google.com`

if [ \$TEST -ge 1 ]; then
       echo "OK"
else
       echo "FAIL"
fi
```

Figura 3.6: *Script* checkdns

É importante que as regras de *firewall* estejam configuradas para carregar automaticamente durante o processo de inicialização do sistema operacional.

Estas regras fazem com que todos os pacotes cujo destino sejam os endereços VIP nas portas e protocolos especificados sejam redirecionadas para o próprio servidor real. Desta forma, o servidor real passa a "responder" pelos endereços VIP, entretanto apenas para o conjunto de portas especificadas.

```
virtual ldap {
    active = 1
    address = 10.0.0.6 eth0:2
    vip_nmask = 255.255.255.0
    port = 389
    persistent = 60
    load_monitor = none
    scheduler = wlc
    protocol = tcp
    timeout = 6
    reentry = 15
    quiesce_server = 0
    server real1 {
        address = 10.0.0.3
        active = 1
        weight = 1
    }
    server real2 {
        address = 10.0.0.4
        active = 1
        weight = 1
    }
}
```

Figura 3.7: Trecho do arquivo de configuração lvs.cf – Configuração do Servidor Virtual LDAP

```
# iptables -t nat -A PREROUTING -p <tcp|udp> -d <vip> \
        --dport <port> -j REDIRECT
```

Figura 3.8: Template de regra de *firewall* para *Direct Routing*

3.5 Comentários Finais

Após a conclusão das configurações dos servidores virtuais, bastou subir o serviço "pulse" nos dois Directors e fazer com que os clientes utilizassem o endereço VIP do serviço, ao invés dos endereço IP de cada servidor real.

Com isso o *cluster* de balanceamento de carga está no ar, redirecionando as requisições e monitorando o *status* de cada servidor real. O *cluster* de alta disponibilidade entre os Directors também está sendo monitorado e pronto para chavear em caso de necessidades.

```
# iptables -t nat -A PREROUTING -p tcp -d 10.0.0.5 \
       --dport 631 -j REDIRECT
# iptables -t nat -A PREROUTING -p tcp -d 10.0.0.5 \
       --dport 515 -j REDIRECT
# iptables -t nat -A PREROUTING -p tcp -d 10.0.0.6 \
       --dport 389 -j REDIRECT
# iptables -t nat -A PREROUTING -p tcp -d 10.0.0.7 \
       --dport 53 -j REDIRECT
# iptables -t nat -A PREROUTING -p udp -d 10.0.0.7 \
       --dport 53 -j REDIRECT
```

Figura 3.9: Exemplo de regras de *firewall* para os servidores virtuais

Capítulo 4

Resultados

A implantação da solução adotada levou em torno de 60 dias, considerando a pesquisa pela solução, os testes, implementação do ambiente de produção e a migração gradual dos clientes para o *cluster*.

Uma vez feitas as configurações apontadas no capítulo anterior, os serviços foram contigenciados e consolidados em um único endereço VIP, criando um *cluster* de balanceamento de carga entre os servidores reais e um *cluster* de alta disponibilidade entre os Directors.

A implantação do *cluster* permitiu balancear os serviços de impressão. Isso pode ser verificado de várias formas, dentre elas a tabela de roteamento do IPVS, conforme a Figura 4.1 apresenta. A informação da coluna "ActiveConn" nos informa a quantidade de conexões em estado "ESTABLISHED" existentes para cada servidor real.

Como é possível perceber, as requisições estão sendo balanceadas entre os servidores reais. Em caso de falha em algum dos serviços nos servidores virtuais, o LVS deixa de enviar requisições para o mesmo, entretanto não existe indisponibilidade para o cliente.

Desta forma foi possível criar um ambiente extremamente escalável e com alta disponibilidade. Caso os servidores reais fiquem sobrecarregados basta adicionar mais um servidor real no *cluster*.

Este cenário também permitiu que os técnicos pudessem desativar um servidor ou serviço durante horário de trabalho para manutenções, sem impactar o serviço para os clientes. Isto reduziu custo com horas extras dos profissionais que precisavam ficar após o horário para executar as manutenções necessárias.

Esta configuração também garantiu a alta disponibilidade do Director, que em caso de falha tem seus serviços repassados para o Director *backup*.

```
# ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
 -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.0.5:515 wlc persistent 60
 -> 10.0.0.3:515                 Route   1      0          0
 -> 10.0.0.4:515                 Route   1      0          0
TCP  10.0.0.5:631 wlc persistent 60
 -> 10.0.0.3:631                 Route   1      100        10
 -> 10.0.0.4:631                 Route   1      105        8
TCP  10.0.0.6:389 wlc persistent 60
 -> 10.0.0.3:389                 Route   1      116        208
 -> 10.0.0.4:389                 Route   1      101        147
TCP  10.0.0.7:53 wlc persistent 60
 -> 10.0.0.3:53                  Route   1      0          2
 -> 10.0.0.4:53                  Route   1      0          2
UDP  10.0.0.7:53 wlc persistent 60
 -> 10.0.0.3:53                  Route   1      0          13844
 -> 10.0.0.4:53                  Route   1      0          36283
```

Figura 4.1: Tabela de roteamento do IPVS

O resultado da implantação da solução foi perceptível para os usuários, pois o serviço precisava de paradas periódicas para atualização de *patches* de segurança e manutenções elétricas em *racks*. Como os servidores reais estão fisicamente separados, até manutenções em setores do *datacenter* não indisponibilizam mais os serviços.

No serviço de impressão também houve uma outra percepção positiva por parte dos usuários, já que até então existiam diversos servidores de impressão e os usuários precisavam pesquisar em que servidor de impressão estava sua impressora. Com a consolidação do serviço em um único servidor virtual, os usuários podem fazer o mapeamento da fila de forma mais rápida e sem pesquisas.

Portanto, a implantação da solução de balanceamento de carga trouxe resultados positivos e reais tanto para a equipe de TI quanto para os usuários, facilitando a manutenção dos servidores e mitigando indisponibilidades nos serviços.

Capítulo 5

Conclusão

O objetivo deste trabalho foi apresentar uma solução que pudesse garantir a disponibilidade e a escalabilidade do serviço de impressão em um ambiente em que este serviço é bastante crítico. Aproveitando a solução adotada para o serviço de impressão, outros serviços poderiam passar a usufruir o serviço de balanceamento de carga.

Para atingir tal objetivo, foi utilizada a solução de balanceamento de carga LVS em conjunto com a solução de alta disponibilidade Piranha.

A solução adotada permitiu produzir um ambiente mais confiável, escalável e com maior flexibilidade para manutenção. Desta forma a disponibilidade atendeu às expectativas da empresa, garantindo um serviço de TI de qualidade.

Toda a solução foi elaborada utilizando *Software* Livre, reduzindo os gastos com equipamentos de balanceamento de carga ou outros *softwares* proprietários.

Com isso, foi alcançado o objetivo inicial. A probabilidade de ocorrer uma indisponibilidade no serviço de impressão foi reduzida com a implantação da solução apresentada neste trabalho. E, agregado a esse objetivo, foi possível adicionar outros serviços no mesmo ambiente redudante e escalável.

Após a implementação do balanceador é possível planejar a migração de mais serviços para o novo ambiente, pois não são mais necessários novos investimentos em balanceadores e os ganhos são bastante reconhecidos e palpáveis. Será possível balancear praticamente todos os serviços de rede utilizando apenas um *cluster* de balanceadores de carga.

Referências Bibliográficas

BOOKMAN, C. *Linux clustering: building and maintaining Linux clusters*. [S.l.]: Sams Publishing, 2003.

DOLBIER, G.; BOGDANOVIC, P.; CIMAFRANCA, D.; JOHNG, Y.; JR., R. C. *IBM Eserver BladeCenter, Linux, and Open Source Blueprint for e-business on demand*. IBM, 2003. Disponível em: <http://ibm.com/redbooks>.

GROPP, W.; LUSK, E.; STERLING, T. *Beowulf Cluster Computing with Linux*. 2. ed. [S.l.]: The MIT Press, 2003.

KOPPER, K. *The Linux Enterprise Cluster—Build a Highly Available Cluster with Commodity Hardware and Free Software*. [S.l.]: William Pollock, 2005.

RED HAT, INC. *Red Hat Cluster Suite for Red Hat Enterprise Linux 5*. 5. ed. [S.l.], 2008. Disponível em: <http://docs.redhat.com/docs/en-US-/Red_Hat_Enterprise_Linux/5/html/Cluster_Suite_Overview/index.html>. Acesso em: 03/04/2011.

RED HAT, INC. *Linux Virtual Server (LVS) for Red Hat Enterprise Linux*. 5. ed. [S.l.], 2009. Disponível em: <http://docs.redhat.com/docs/en-US-/Red_Hat_Enterprise_Linux/5/html/Virtual_Server_Administration/index.html>. Acesso em: 03/04/2011.

SLOAN, J. D. *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*. [S.l.]: O'Reilly, 2004.

WANGSMO, M. *White Paper: Piranha - Load-balanced Web and FTP Clusters*. [S.l.]. Disponível em: <http://www.redhat.com/support/wpapers/piranha/index-.html>. Acesso em: 03/04/2011.