Integração de Alta Disponibilidade e Balanceamento de Carga em um Ambiente de Cluster

M. A. R. DANTAS
UFSC – Universidade Federal de Santa Catarina
INE – Departamento de Informática e Estatística
Cx. Postal 476 – CEP 88040-900 – Florianópolis -SC
mardantas@computer.org

ALBA C. M. MELO, R. M. ALVIM E F. L. L. GROSSMANN
UnB - Universidade de Brasília
CIC - Departamento de Ciência da Computação
Cx. Postal 4466 - CEP 70.910-970 Brasília-DF
albamm@cic.unb.br

Resumo

Ambientes de cluster de microcomputadores são configurações interessantes para execução de inúmeras tarefas distribuídas e paralelas. Todavia, estes ambientes computacionais não provêm nativamente uma maior confiabilidade e um balanceamento de carga adequado entre os elementos do cluster. Neste artigo apresentamos uma integração das facilidades de alto-desempenho e balanceamento de carga em um ambiente de cluster, rodando o sistema operacional Linux. Nossos resultados experimentais indicam que a integração pode representar uma melhoria significativa na configuração do cluster para execução de tarefas distribuídas.

1. Introdução

Nos dias atuais, empresas que funcionam ininterruptamente não podem ter seus sistemas computacionais comprometidos, seja por uma hora, ou até mesmo por alguns minutos. Uma fração de tempo do sistema fora do ar pode significar grandes prejuízos para a instituição. Uma estrutura de redundância se faz necessária, a fim de minimizar os riscos da ocorrência de falhas no ambiente computacional. Exemplos de grandes corporações que já perceberam a importância dessa abordagem são a IBM [2] e a Microsoft [6]. Estas duas empresas já iniciaram um investimento comercial significativo no desenvolvimento de produtos para satisfazer os requerimentos de uma maior disponibilidade dos sistemas computacionais.

Em adição as soluções proprietárias, alguns esforços de desenvolvimento de ambientes abertos começam a demonstrar uma maturidade funcional. Exemplos clássicos desses esforços são o Linux Alta-Disponibilidade (*Linux High-Availability -HA*) [3,7] e o Servidor Virtual Linux (*Linux Virtual Server* - LVS) [4,5].

O primeiro ambiente, o Linux Alta-Disponibilidade (Linux HA), consiste de um pacote de software onde podemos fazer que um computador seja a imagem de um outro. Em outras palavras, se ocorrer algum problema com o primeiro computador em operação, imediatamente, o segundo computador espelho do primeiro (*backup*) assume, de forma transparente para os usuários, todos os serviços disponibilizados pelo primeiro computador. Dessa forma, é possível manter aplicações críticas no ar por um maior tempo possível.

No ambiente do Servidor Virtual Linux (LVS), temos como principal característica o balanceamento de carga entre servidores de uma configuração de cluster. À medida que as requisições de serviço chegam, um dos servidores reais do LVS é escolhido para atendê-la. A grande vantagem dessa solução é o paralelismo

alcançado, pois as solicitações de serviço são atendidas ao mesmo tempo por diversos computadores. Desta forma, é possível atender uma maior demanda usando máquinas menos robustas e com menor custo. Contudo, o próprio ambiente LVS pode significar uma vulnerabilidade que pode comprometer a disponibilidade de todos os serviços, uma vez que o computador que estiver executando esse serviço pode apresentar falha.

Neste artigo apresentamos nossa investigação na integração das soluções de alta-disponibilidade e servidor virtual. Nosso objetivo é unificar as vantagens do Linux-HA e do LVS, eliminando a vulnerabilidade do servidor virtual e criando um ambiente altamente confiável a um custo relativamente baixo. Desta forma, na seção 2 abordamos o funcionamento dos ambientes de software Linux HA e LVS. O nossa configuração de cluster é descrita na seção 3. Os resultados experimentais da integração dos ambientes de software são apresentados na seção 4. Finalmente, na seção 5 tecemos nossas conclusões sobre este trabalho de pesquisa.

2. Alta-Disponibilidade e Servidor Virtual

2.1- Linux Alta-Disponibilidade (Linux –HA)

Existem dois conceitos importantes no ambiente Linux-HA. O conceito de *endereço IP virtual* e o conceito de *heartbeat.* O endereço IP virtual é o endereço IP do serviço e não o endereço IP real da máquina que hospeda o serviço. Dessa forma, o endereço IP virtual é *assumido* por uma determinada máquina, momentaneamente, a qual é responsável pelo serviço. Então, em um outro instante este endereço pode ser assumido por outra máquina reserva (*backup*) quando ocorrer uma falha na primeira. Um agrupamento de computadores (*cluster*) pode conter múltiplos IPs virtuais. Cada IP pode estar associado a um ou mais serviços [6]. O *heartbeat* é o programa responsável por monitorar os serviços em ambientes de alta disponibilidade. Além disso, também prove os serviços de comunicação *intracluster*, que fazem a troca de mensagens entre os servidores [7]. Assim, é este software que determina qual servidor assumirá o endereço IP virtual.

O Linux-HA funciona como *daemon* nos dois servidores. A partir da inicialização do Linux, o *heartbeat* é inicializado e há uma verificação do funcionamento dos servidores. O IP virtual é criado no servidor principal e, a partir deste momento, os servidores ficam trocando mensagens entre si. Este procedimento serve para o teste recíproco de suas disponibilidades [3]. Na figura 1 ilustramos um exemplo do Linux-HA em funcionamento.

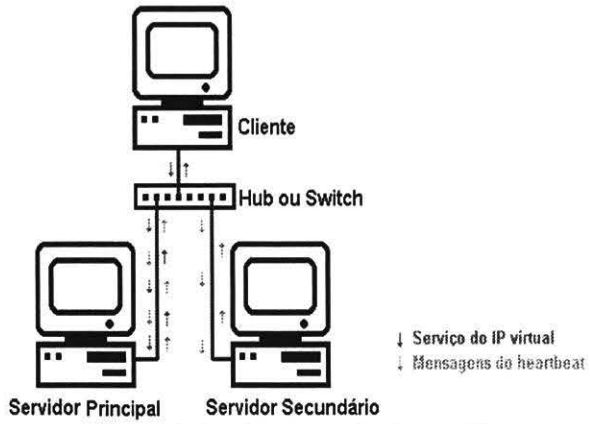

Figura 1: Funcionamento do Linux –HA

No ambiente do cluster é determinado um intervalo máximo que pode ocorrer entre o envio da mensagem do *heartbeat* e sua resposta. Quando este tempo é alcançado, o nó que não respondeu é considerado indisponível e é desabilitado pelo *heartbeat.* Se o IP virtual estava ativo nesta máquina, o *hearbeat* ativa o servidor *backup* e transfere o IP virtual para este. Os serviços que estavam sendo atendidos pelo servidor que falhou são agora atendidos pelo que entrou em operação. As mensagens continuam sendo enviadas pelo servidor operante até que o servidor que apresentou a falha volte a funcionar de maneira satisfatória. Quando isso acontece, o *heartbeat* decide, baseado em uma configuração conhecida como *nice failback* (ligada ou desligada), qual dos dois deve assumir o IP virtual. Se estiver ligada, os servidores trocam de papel a cada falha do primário, fazendo com que o servidor primário assuma o papel de secundário quando voltar a operar normalmente. Se estiver desligada, o servidor primário, ao voltar, retira o secundário de operação e assume novamente as suas atribuições nas solicitações de serviço.

2.2 Servidor Virtual Linux (LVS)

O servidor virtual Linux (LVS) é uma configuração de *cluster* de servidores que aparenta ser um servidor único para um cliente externo. Este *servidor único* é chamado de *servidor virtual.* Do ponto de vista do cliente externo, ele recebe requisições de serviços e as trata da mesma forma que um servidor comum. O servidor virtual é composto por dois tipos de computadores: o balanceador de cargas e os servidores reais. A função do balanceador de cargas é repassar os serviços requisitados para os servidores reais, que irão tratá-los.

Baseado nas informações de IP, contidas no cabeçalho dos pacotes, o balanceador toma decisões e apenas *vê passar* os pacotes entre o cliente e o servidor real. Todo o

relacionamento padrão de cliente-servidor é preservado. Cada cliente *imagina* estar conectado diretamente ao servidor real e cada servidor real *imagina* estar conectado diretamente ao cliente. Assim, nenhum dos dois sabe que um balanceador interveio na conexão [5].

O uso do servidor virtual tem como objetivo prover alto desempenho e alta disponibilidade para aplicações, através de redundância e balanceamento de carga entre máquinas [4]. Como [1] comenta, esta facilidade é essencial para diversas aplicações distribuídas e paralelas em ambientes de *cluster e grid computing* .Na figura 2 mostramos um ambiente LVS em funcionamento.

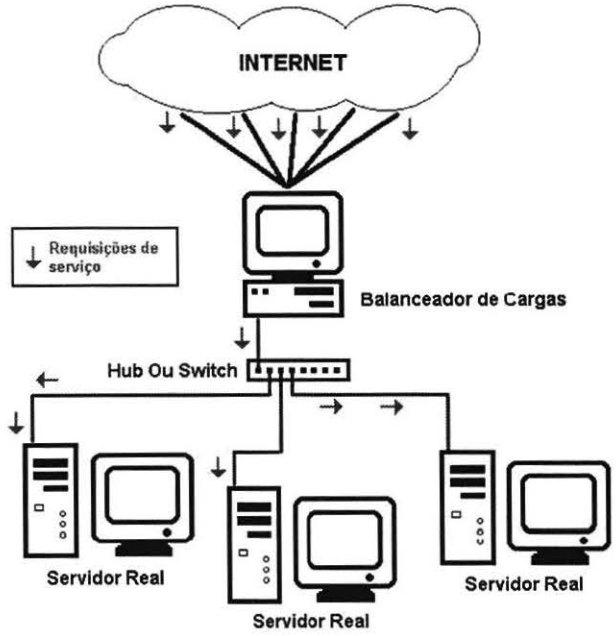

Figura 2: Funcionamento do LVS

No servidor virtual, o programa que cuida do monitoramento e distribuição dos serviços é o *ldirectord*. Executando no balanceador de cargas, este usa o envio de mensagens para identificar se os servidores reais estão em operação ou se apresentam falhas. No caso de falhas, este software se encarrega de redistribuir a carga entre os outros servidores, deixando o equipamento com problemas fora do *cluster* até que ele volte a sua operação normal, quando será reinserido automaticamente.

O *ldirectord* pode utilizar um abordagem de escalonamento diversificada, na qual o software pode decidir de maneira distinta como distribuir a carga entre os servidores reais. Existem diferentes formas de configuração do algoritmo de escalonamento (exemplos *round-robin, round-robin ponderado, conexão mínima, conexão mínima ponderada)*, sendo a *round-robin* a mais utilizada. No *round-robin*, definem-se pesos para cada servidor real (normalmente peso 1 se todos forem iguais). Baseado nesses pesos o balanceador, dentro de uma rodada de requisições, escolhe o servidor real tantas vezes quanto indicadas no seu peso. Desta forma, se (por exemplo) os pesos são 2, 3 e 4, numa rodada de nove requisições o primeiro servidor atenderá duas, o segundo três e o terceiro quatro requisições respectivamente. As vantagens do uso do servidor virtual são:

- apresenta custo relativamente baixo - todo o software utilizado é aberto, sob a licença pública da GNU. O hardware não precisa ser sofisticado, pois a quantidade de máquinas, ainda que mais simples, mantém o desempenho elevado.
- a redundância possibilita manutenção e *upgrade* de servidores reais sem tornar necessária a indisponibilidade do serviço, bastando para isso, retirar uma máquina de cada vez e reinserí-la antes de desconectar outra.
- provê adaptabilidade - se o sistema precisa ser incrementado temporariamente, por exemplo, por causa de um evento ou de um aumento de vendas que acontece em determinada data, o numero de servidores reais pode ser facilmente incrementado e, quando tal período passar, podem ser novamente retirados, novamente sem a interrupção de serviços para os clientes [4].

O grande risco dessa implementação é o ponto crítico de falha, localizado no balanceador de cargas. Se esta máquina falha, todo o servidor virtual fica comprometido e o serviço, indisponível. Sanar esse ponto de falha é um dos focos do nosso trabalho. A eliminação desse problema traz a confiabilidade necessária para o uso em serviços críticos. O que propomos, então, é a redundância do balanceador de cargas no servidor virtual, com o uso do Linux-HA. Dessa forma aproveitamos as vantagens do balanceamento de carga além de prover alta disponibilidade. Ao final o *cluster* fica como o apresentado na figura 3.

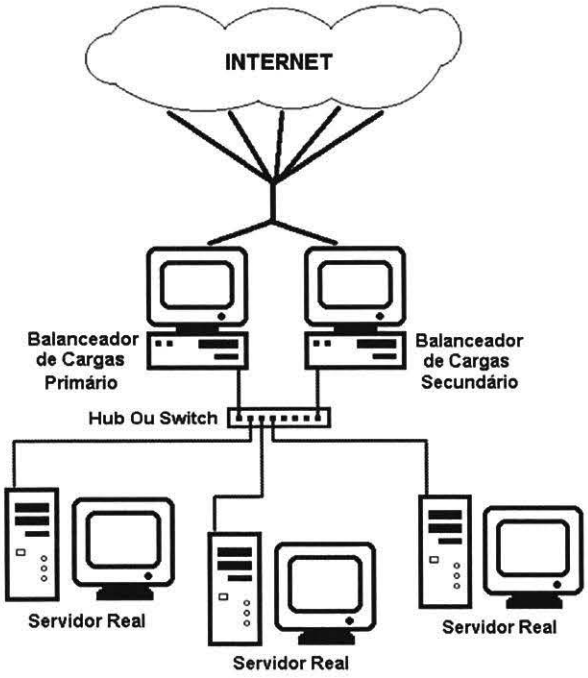

Figura 3: Esquema da integração Linux-HA/LVS

A administração do *cluster* é feita por intermédio de uma ferramenta denominada IPVSADM. Este é um programa presente no balanceador de cargas, que apresenta a composição do *cluster*, mostrando quais são os servidores reais que em determinado instante compõem o servidor real, bem como quantas requisições estão sendo atendidas individualmente por eles.

3. Ambiente do *Cluster*

Na integração do Linux-HA com o LVS, utilizamos um total de cinco microcomputadores do tipo IBM-PC, localizados no LAICO (Laboratório de Sistemas Integrados e Concorrentes), no Departamento de Ciência da Computação da Universidade de Brasília (UnB). Os microcomputadores foram conectados por um *hub-Ethernet*. As características dos microcomputadores utilizados são apresentadas na tabela 1.

Processador	**Pentium 100-100-233-233-233**
Memória	**32-32-64-64-64**
S.O	**Linux 6.0 kernel 2.2.17**

Tabela 1: Ambiente do *Cluster*

4. Resultados Experimentais

Para configurar o ambiente, o primeiro passo foi colocar em operação o Linux-HA. Durante a instalação do sistema operacional Linux, em duas máquinas (denominadas de LAICOY e LAICOZ) selecionamos o pacote alta disponibilidade (as versões atuais do sistema operacional Linux já possuem a ferramenta de alta disponibilidade integrada).

Depois de finalizada a instalação nas duas máquinas, nosso próximo passo foi a configuração da ferramenta. Com este objetivo é necessário o uso do pacote *Linuxconf*. Os passos da configuração consistem em indicar as máquinas que irão compor o *cluster*, qual será o IP virtual e o serviço associado que será atendido, localização de arquivos de *log*, intervalo entre as mensagens do *heartbeat*, e o meio e método de autenticação que o *heartbeat* usa para enviar essas mensagens.

Finalizada a configuração, iniciamos o serviço em cada máquina. Após um breve teste, verificamos que o serviço estava operando corretamente. O segundo passo no estabelecimento do ambiente foi colocar em operação o servidor virtual. Por não ter instalação integrada ao sistema operacional, procuramos os pacotes necessários na Internet. O servidor virtual foi instalado e configurado nas máquinas LAICOY (o *heartbeat* foi temporariamente desabilitado) e demais máquinas do ambiente, denominadas de LAICO2, LAICO3 e LAICO4. A configuração do LVS se baseia na configuração do *ldirectord*, que é composta por duas partes, a configuração nos balanceadores de carga e a configuração nos servidores reais.

Efetuamos alguns testes que nos indicaram a operação correta do servidor virtual. O último passo na montagem do ambiente foi a integração do Linux-HA com o LVS. Para atingirmos este objetivo foi preciso duplicar a configuração do balanceador de cargas feita no LAICOY, copiando-a para o LAICOZ e uma pequena alteração na configuração do Linux-HA, definindo o *ldirectord* como o serviço a ser disponibilizado. O Linux-HA será o responsável por monitorar o funcionamento dos balanceadores de carga somente. Foi escolhido o serviço *http* (páginas da Internet) como o serviço prestado pelo *cluster* e o IP virtual *164.41.14.146* como o IP do serviço. Esse IP virtual é um endereço IP válido na Internet. O *cluster integrado* ficou com ilustrado na figura 4:

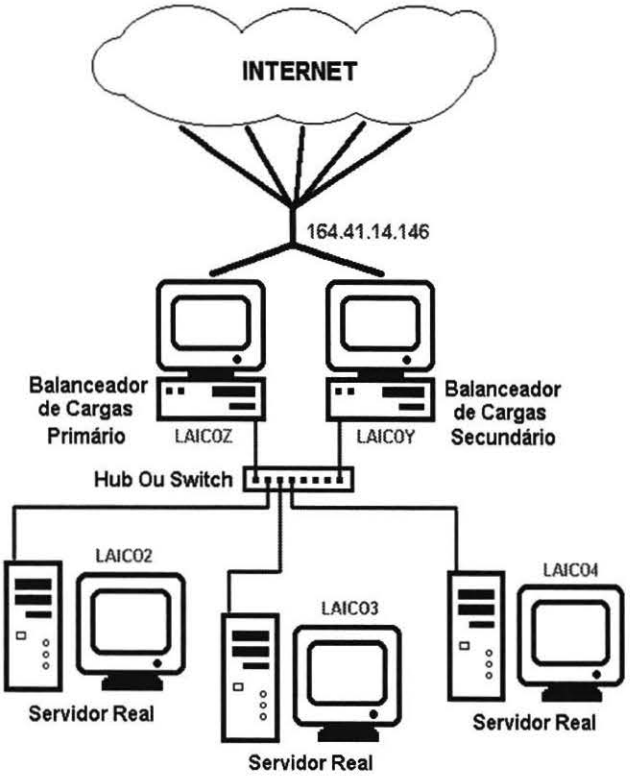

Figura 4: Ambiente Linux-HA / LVS Integrado

É importante destacar que o balanceador primário de nosso cluster é o computador LAICOZ, sendo assim, no correto funcionamento do *cluster*, apenas ele recebe e repassa requisições para os microcomputadores LAICO2, LAICO3 e LAICO 4. O servidor LAICOY é apenas um *backup*, e só entra em operação quando o LAICOZ falhar. Com o objetivo de testar o ambiente integrado dividimos em três etapas os testes :

- fazer várias requisições de serviço ao endereço IP, anotando o estado do ambiente;
- retirar de operação o servidor LAICO2, anotar o estado do ambiente e recolocá-lo em operação;
- retirar de operação o servidor LAICOZ, anotar o estado do ambiente e recolocá-lo em operação.

Para verificarmos o estado do ambiente em cada etapa, utilizaremos a ferramenta IPVSADM. Em condições normais do *cluster*, a saída que tal ferramenta nos apresenta é ilustrada na figura 5.

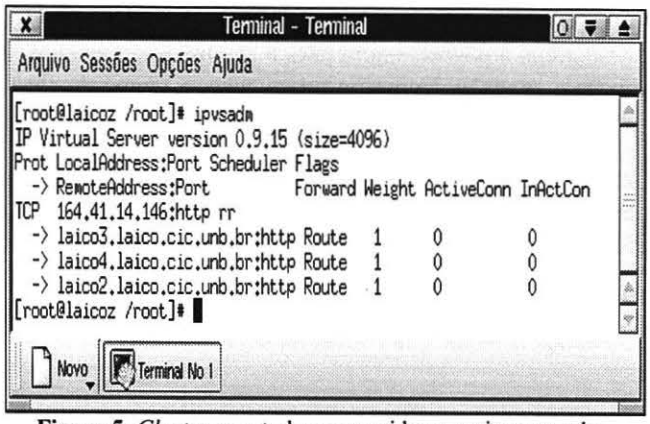

Figura 5: *Cluster* com todos os servidores reais operando.

A primeira linha mostra que estamos na máquina LAICOZ (o balanceador primário). Na segunda linha ilustra a versão da ferramenta, no caso *0.9.15*. Por outro lado, a quinta linha mostra o protocolo utilizado (*TCP*), o endereço IP virtual do *cluster* (*164.41.14.146*), o serviço disponibilizado (*http*) e o algoritmo de escalonamento utilizado (*rr = round robin*). A sexta, sétima e oitava linhas mostram os servidores reais ativos no momento, o peso de cada um no algoritmo de escalonamento (no nosso caso = **1**), o número de conexões ativas (**0**) e o número de conexões inativas (**0**). Estes dois últimos iguais a zero mostram que nenhuma conexão está sendo feita ou foi terminada no momento.

Na primeira etapa de testes, que significa fazer várias requisições ao endereço IP virtual, visando saber como o *cluster* iria atender aos serviços, fizemos requisições de uma máquina da Internet (exemplo : digitamos o endereço IP 164.41.14.164 no seu navegador e solicitamos várias vezes no botão atualizar, totalizando quinze requisições). Ao usar a ferramenta IPVSADM, logo após essas requisições, o resultado é ilustrado na figura 6.

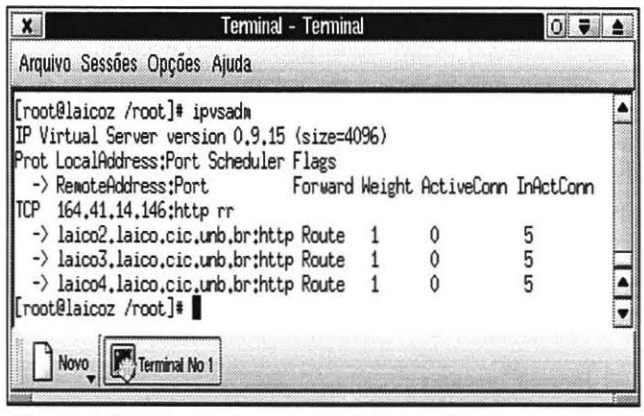

Figura 6: Estado do *cluster* após várias requisições de serviços

As requisições são contadas como inativas, pois ao requisitar páginas *web*, o cliente abre a conexão, descarrega a página inteira para o seu microcomputador e fecha a conexão. Como no nosso caso, o tempo de abrir, descarregar a página e fechar a conexão é insignificante, o IPVSADM sempre mostrará a conexão como inativa. Como exemplo, no caso do TELNET, a conexão apareceria como ativa até que a sessão fosse finalizada. O numero de requisições foi igualmente distribuído, pois definimos o peso sendo **1** para todos os servidores reais. Como o resultado apresentado foi dentro do esperado, podemos concluir que o teste foi bem sucedido.

Como segunda etapa de teste, retiramos de operação (desligando) o servidor real LAICO2 com ilustrado na figura 7.

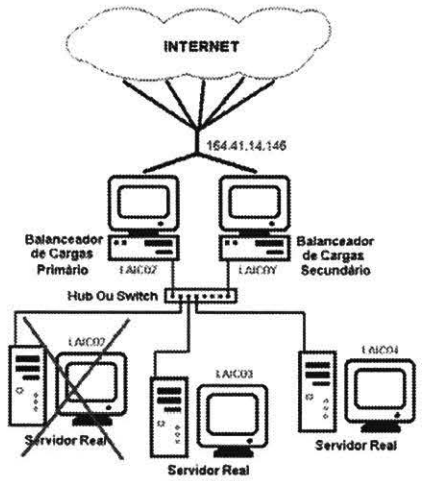

Figura 7: Retirada do servidor real LAICO2

Como conseqüência é esperado que o balanceador de cargas retire o microcomputador LAICO2 do *cluster* e redistribua a carga apenas aos microcomputadores LAICO3 e LAICO4. Para sabermos o estado atual do *cluster*, novamente fizemos uso da ferramenta IPVSADM, cuja saída é apresentada na figura 8.

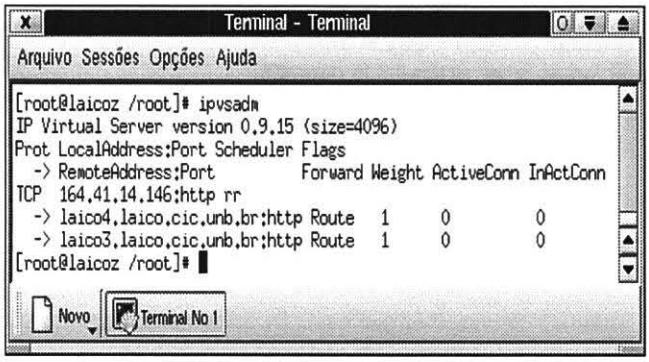

Figura 8: *Cluster* sem o servidor real LAICO2 operando.

Notamos na figura 8 que o microcomputador LAICO2 foi realmente retirado do *cluster*. Religamos o microcomputador LAICO2. Espera-se que o balanceador coloque-o novamente no *cluster*. Usando novamente a ferramenta IPVSADM e obtivemos a seguinte saída:

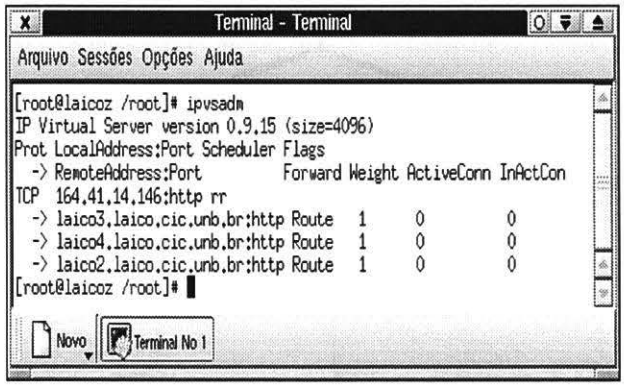

Figura 9: *Cluster* após volta do servidor real LAICO2

Notamos que o servidor real LAICO2 voltou a fazer parte do *cluster*, podemos concluir mais uma vez que o teste foi bem sucedido.

Como terceira etapa dos nossos experimentos retiramos de operação o balanceador primário LAICOZ. Antes de retiramos o balanceador principal de operação, usamos a ferramenta IPVSADM no balanceador secundário para verificar seu estado. A saída apresentada é ilustrada na figura 10:

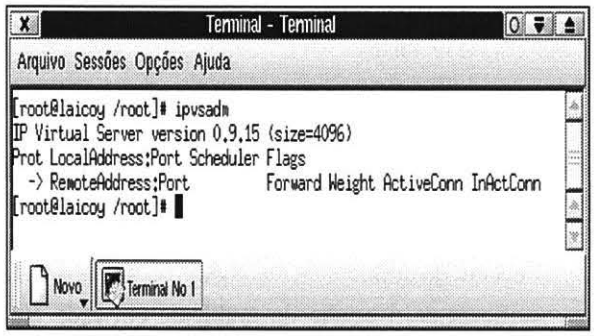

Figura 10: IPVSADM no balanceador secundário, com o primário ainda ativo

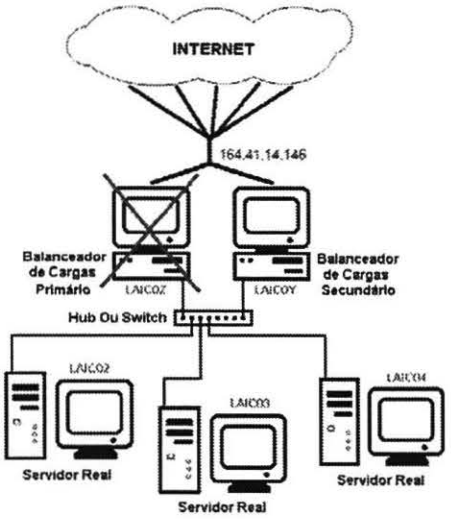

Figura 11: Retirada do balanceador primário LAICOZ

Desligamos o LAICOZ, o balanceador de cargas principal, retirando-o de operação. Dessa forma, é esperado que o *heartbeat* acione o balanceador secundário LAICOY, fazendo a redistribuição de carga entre os servidores reais, mantendo a operação normal do *cluster*. A saída que o IPVSADM apresenta nesse momento (a partir de LAICOY, balanceador secundário) é mostrada na figura 12 a seguir.

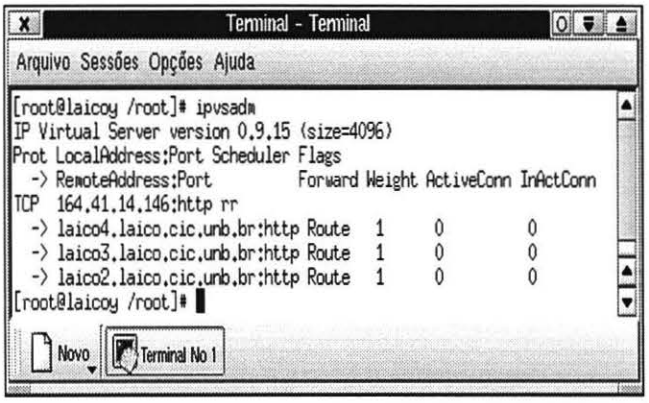

Figura 12: IPVSADM no balanceador secundário, com o primário já inativo

Observamos que o *cluster* já está novamente em operação. Fizemos algumas requisições ao endereço IP 164.41.14.146, a partir de um microcomputador na Internet. O resultado apresentado na saída do comando IPVSADM foi semelhante ao apresentado na figura 6. Logo, o *cluster* está operando normalmente. Nesse momento, religamos o servidor primário. Como o *nice failback* não foi selecionado, é esperado que assim que o balanceador primário LAICOZ volte a operar, ele recupere a função de distribuir a carga, desativando o balanceador secundário LAICOY. Isso realmente acontece, podendo ser demonstrado pela saída do IPVSADM nas duas máquinas após LAICOZ voltar a operar:

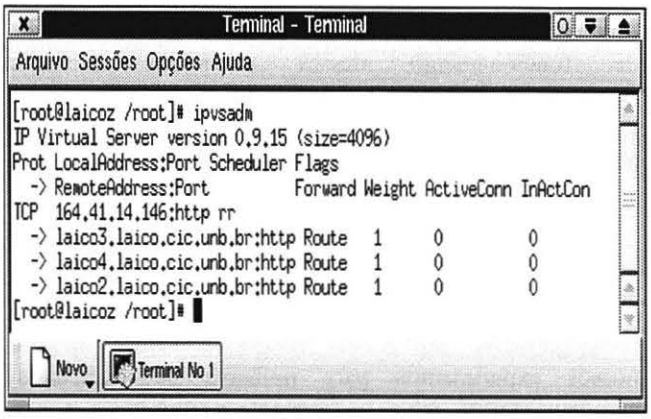

Figura 13: IPVSADM em LAICOZ depois de voltar a operar

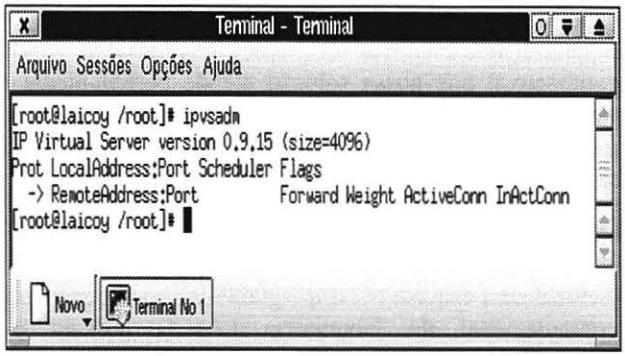

Figura 14: IPVSADM em LAICOY após LAICOZ voltar a operar.

Notamos que o estado do sistema é exatamente igual ao estado inicial, mostrado nas figuras 5 (LAICOZ) e 10 (LAICOY). Novamente o teste realizado nesta etapa foi bem sucedido.

5. Conclusões

Os riscos de prejuízo em uma corporação quando seus sistemas estão baseados em um ambiente computacional só são reduzidos com um planejamento adequado e sério, assim é importante a adoção de planos de contingência em casos de desastres. O mais empregado e mais eficiente plano de contingência é a convencional duplicação dos serviços fundamentais para o ambiente da

organização. A duplicação da base de dados já é comum na maioria das empresas. Porém, as atividades de algumas empresas não dependem apenas das informações guardadas em seu banco de dados, mas também em manter contínuo o serviço que prestam. Desta forma, a duplicação da prestação do serviço pode significar a viabilização de uma eventual substituição de emergência. A importância do nosso trabalho está exatamente em apresentar uma solução nessa área. Não só pelo fato de duplicar servidores, mas também de monitorar o estado de funcionamento desses servidores e agir automaticamente em caso de falhas. Outra grande vantagem é utilizar uma solução de software aberta, acessível por qualquer um, sem custo adicional.

Podemos concluir com base nos nossos testes que a nossa solução experimental de integração dos ambiente Linux-HA e LVS comportou-se de maneira satisfatória . Como prosseguimento de nossas pesquisas vamos ampliar nossos experimentos para milhares de requisições simultaneamente, situação freqüente e comum no mundo real da Internet. Todavia, apesar do número reduzido de requisições podemos afirmar que o presente trabalho aponta esta solução com uma abordagem correta para organizações de pequeno porte que precisem de disponibilidade contínua sem muito investimento. Nosso sentimento é que nossa solução atende perfeitamente a grandes demandas, bastando melhorar a infra-estrutura do *cluster*, com máquinas mais robustas e equipamentos de rede mais rápidos.

Agradecimentos

Este projeto foi parcialmente financiado pelo Conselho Nacional de Pesquisa (CNPq - 300874/006). O ambiente {computacional do laboratório LAICO provem do PROJECTO INEG/CNPq-UnB.

Referências

[1] Dantas, Mario, *Tecnologias de Redes de Comunicação e Computadores*, Axcel Books, 2002.

[2]http://www.ibm.com/servers/eserver/pseries/solutions/ha/hans.html, 2002.

[3] http://www.Linux-HA.org, 2001.

[4] http://www.Linuxvirtualserver.org, 2001.

[5] Mack, J. , "LVS-HOWTO" http://www.linuxvirtualserver.org/Documents.html#manuals, 2001.

[6] Martin, Peter (p.martin@ies.uk.com) e Cronin, John (jsc3@havoc.gtf.org), 05/07/2001, http://www.microsoft.com/SERVICEPROVIDERS/solutions/haoview.asp, 2001.

[7] Milz, H. , "HA-HOWTO" http://www.ibiblio.org/pub/Linux/ALPHA/linux-ha/High-Availability-HOWTO.html