UNIVERSIDADE TECNOLOGICA FEDERAL DO PARANÁ – UTFPR
CÂMPUS PONTA GROSSA
DEPARTAMENTO ACADÊMICO DE INFORMATICA – DAINF
COORDENAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EMANÁLISE E DESENVOLVIMENTO DE SISTEMAS – COADS

JUSSAN DOS SANTOS ZELA
ROBSON HILÁRIO ANTUNES DA SILVA

CRIAÇÃO DE UMA DISTRIBUIÇÃO GNU/LINUX PARA CLUSTER DE BALANCEAMENTO DE CARGA E ALTA DISPONIBILIDADE

TRABALHO DE DIPLOMAÇÃO

PONTA GROSSA
2013

JUSSAN DOS SANTOS ZELA
ROBSON HILÁRIO ANTUNES DA SILVA

CRIAÇÃO DE UMA DISTRIBUIÇÃO GNU/LINUX PARA CLUSTER DE BALANCEAMENTO DE CARGA E ALTA DISPONIBILIDADE

Trabalho de conclusão de curso de graduação, apresentado à disciplina Trabalho de Diplomação, do curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Prof. Me. WillianMassami Watanabe.
Co-orientador: Prof. Me. Wellton Costa de Oliveira.

PONTA GROSSA
2013

TERMO DE APROVAÇÃO

Criação De Uma Distribuição Gnu/Linux Para Cluster De Balanceamento De Carga E Alta Disponibilidade

por

JUSSAN DOS SANTOS ZELA
ROBSON HILÁRIO ANTUNES DA SILVA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado (a) em 26 de março de 2013 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixoassinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____

Willian Massami Watanabe
Prof.Orientador

_____

Wellton Costa de Oliveira
Prof.Co-Orientador

_____

Talita Scharr Rodrigues
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À meus pais, que me proporcionaram as oportunidade para chegar até aqui .

A minha esposa Andressa, que soube compreender e respeitar meu humor durante o desenvolvimento deste trabalho, dando-me todo o apoio necessário.

Aos amigos que contribuíram de alguma forma para a conclusão deste trabalho e não perderam a esperança que um dia eu concluiria este trabalho.

Ao professorWellton, que aceitou e apoiou a ideia do desenvolvimento deste trabalho, sem nenhum beneficio próprio com linha de pesquisa ou outros itens acadêmicos.

Ao professor Willian, que caiu de paraquedas no final do semestre, mas aceitou assumir a responsabilidade de ser nosso orientador enos auxiliando de maneira construtiva e colaborando para a finalização deste trabalho.

A professora Thalita pela colaboração e pelo esforço dedicado ao nosso auxilio.

Robson Hilário Antunes da Silva.

AGRADECIMENTOS

Primordialmente agradeço a Deus. Por ter me iluminado e permitido que eu terminasse mais essa etapa da minha vida, não deixou que desistisse mesmo com tantas dificuldades durante o dia-a-dia.

Ao meu amigo e colega de trabalho, Robson, que foi fundamental para o desenvolvimento deste trabalho.

A minha amiga e esposa do Robson, Andressa, que aguentou toda a bagunça que fizemos na sua casa por todo esse tempo.

Aos professores Thalita, Wellton e Willian, que acreditaram em nosso potencial e fizeram com que o trabalho fluísse, sendo essenciais para que conseguíssemos concluir o mesmo. Não se esquecendo de agradecer, também, a paciência e o companheirismo para conosco, o meu muito obrigado.

A minha namorada, Priscila, que foi um dos pilares da minha caminhada acadêmica, e principalmente da minha vida pessoal nos últimos 4 anos. Quem me deu apoio, compreensão, “puxão de orelha” e inspiração até o presente momento. Muito obrigado pela dedicação e por ser minha namorada.

A minha família e amigos, que contribuíram de alguma maneira para que eu chegasse até aqui.

Por fim agradecer aos meus pais, José e Mariana que me concederam o dom da vida.

Jussan dos Santos Zela.

RESUMO

ZELA, Jussan dos Santos; SILVA, Robson H. Antunes**. CRIAÇÃO DE UMA DISTRIBUIÇÃO GNU/LINUX PARA CLUSTER DE BALANCEAMENTO DE CARGA E ALTA DISPONIBILIDADE** 2013.50f. Trabalho de Conclusão de Curso de Tecnologia em Análise e Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2013.

Este trabalho tem como finalidade demonstrar o uso de ferramentas para agregar a uma aplicação Web a escalabilidade, alta disponibilidade e balanceamento de carga. Tais recursos são obtidos com a utilização de softwares para gerenciamento de *cluster* e também na arquitetura de servidores que irão disponibilizar a aplicação, podendo utilizar conceitos de escalabilidade vertical, e escalabilidade horizontal, clusterização e balanceamento de carga. Além disso, é abordada a criação de uma distribuição *GNU/LINUX,* pre-configurada para facilitar o processo de instalação ou recuperação dos componentes do cluster.

Palavras-chave: Clusterização. Balanceamento de Carga. Desempenho, Alta Disponibilidade.

ABSTRACT

Zela, Jussan dos Santos; SILVA, H. Robson Antunes**. CREATION OF A DISTRIBUTION GNU / LINUX CLUSTER FOR BALANCING LOAD AND HIGH AVAILABILITY** 2013.50f. Conclusion Work Course Technology Analysis and Systems Development - Federal Technological University of Paraná. Ponta Grossa, 2013.

This paper aims to demonstrate the use of tools to add to a web application scalability, high availability and load balancing. Such features are achieved with the use of software for cluster management and also in server architecture that will provide the application and can use the concepts of vertical scalability and horizontal scalability, clustering and load balancing. In addition, we discuss creating a GNU/LINUX, pre-configured for easy installation process or recovery of cluster components.

Keywords: Clustering. Load Balancing. Performance, High Availability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Diante das exigências impostas pelo avanço da tecnologia e a aceleração no modo como as informações são disponibilizadas nos deparamos com um mundo atual cada vez mais carente de tecnologia. Esta tecnologia deve suportar tal demanda de maneira satisfatória, rápida e a um baixo custo, visto que hoje as soluções precisam atender a todas as classes sociais.

Com o ascendente avanço tecnológico, constantemente são desenvolvidos novos recursos e soluções,utilizando software livre para garantir o acesso e a segurança da informação. No entanto, esses novos recursos criam demanda de processamento computacional que somente é possível de ser executada satisfatoriamente por supercomputadores (PITANGA, 2004; TANNENBAUM, 2001).

O custo para aquisição e manutenção de supercomputadores é financeiramente alto e a prestação do serviço é realizada somente pela empresa que projetou o supercomputador. Esse tipo de solução também apresenta como ponto fraco a baixa escalabilidade, quase nenhuma adição de novos componentes no sistema é possível quando é desejado expandir o seu desempenho (PITANGA, 2004).

Para resolver esse tipo de problema alguns pesquisadores iniciaram uma linha de pesquisa com tema em desenvolvimento de *clusters* computacionais.

Segundo Patterson et. al. (2000), os computadores de uso pessoal ficam a cada dia mais baratos e com maior desempenho em termos de processamento de dados, devido aos avanços tecnológicos.

Em muitas empresas de pequeno e médio porte, os departamentos de Tecnologia de Informação (TI) não possuem capital para aquisição deste tipo de tecnologia. Sendo assim, um ambiente computacional que permita escalabilidade e alta disponibilidade aliado a baixo custo de aquisição torna-se o cenário ideal para este público.

Segundo Pitanga (2003), criar um *cluster* de computadores é construir um sistema com dois ou mais computadores com capacidade de dividir tarefas de tal maneira que para o usuário só exista um computador.

Como o sistema consiste na união de computadores, de acordo com Bookman (2003), o *cluster* é caracterizado pela alta disponibilidade e tolerância a falhas que garante um ambiente de alta disponibilidade.

Conforme Gomes (2001), a alta disponibilidade não é algo a ser instalado ou comprado, mas uma característica oferecida pela estrutura computacional.

Como citado porLacerda (2008), a utilização de computadores comuns com informações replicadas entre si permite que o *Google* suporte a alta demanda exigida pelos seus usuários com um custo de implementação e manutenção bem menor, com qualidade e confiabilidade, exceto pelo alto consumo de energia para a operação de todos estes computadores (LACERDA *et. al*. 2008).

1.1. OBJETIVO

Este trabalho tem como objetivo criar uma distribuição *GNU/LINUX* para instalação e configuração de um *cluster* de balanceamento de carga e alta disponibilidade, o qual será utilizado para garantir uma aplicação *web*.

Como objetivos específicos podem ser relacionados os seguintes:

- Configurar uma estrutura de cluster para balanceamento de carga e alta disponibilidade com *failover* (quando uma máquina assume os serviços de outra);
- Configurar uma estrutura em cluster para balanceamento de carga e alta disponibilidade com *failover* utilizando *HAProxy* e *Heartbeat* no *DebianSqueeze6.0.6.*
- Executar testes para validar o desempenho e a escalabilidade do sistema utilizando um sistema *web* já desenvolvido;
- Criar uma imagem ISO com a customização do *Debian Squeeze* com a configuração do cluster.

1.2. JUSTIFICATIVA

Na literatura existem relatos que cluster é uma ferramenta poderosa, de baixo custo se comparado aos mainframes e servidores dedicados. Uma Distribuição pré-configurada serve como ponto de partida para uma instalação nova ou substituições de máquinas pertencentes ao cluster, sem a necessidade

de downloads e configurações de pacotes, desse modo transformam-se alguns desses procedimentos, normalmente demorados e complicados, em algo mais confiável, ágil e prático tanto para os novos clusters quanto aos já configurados.

Para o desenvolvimento de tal sistema optou-se pela distribuição Debian Squeeze6.0.6, pois se observou a quantidade de recursos que a mesma proporciona, além de ser uma solução adquirida abertamente.

1.3. ESTRUTURA DO TRABALHO

O trabalho está dividido na seguinte estrutura: No Capítulo 1 é apresentada uma introdução, abordado os objetivos e a justificativa que levou ao desenvolvimento deste trabalho. No Capítulo 2, a fundamentação teórica é realizada para que se tenha um embasamento sobre as teorias vinculadas com o trabalho. No Capítulo 3, o desenvolvimento, são descritas as tarefas de montagem e configurações da estrutura do trabalho. Por fim, no Capítulo 4 serão apresentados os testes, resultados e conclusão do mesmo.

2. FUNDAMENTAÇÃO TEÓRICA

Esse Capítulo apresenta conceitos relacionados a configuração de *cluster*, balanceamento de carga e alta disponibilidade. Além disso, são apresentadas informações sobre as ferramentas que foram utilizadas para o desenvolvimento do trabalho.

2.1. SERVIDORES WEB

Um servidor web aguarda permanentemente por solicitações oriundas dos clientes. As requisições são recebidas, interpretadas e retornam ao cliente o objeto solicitado. O conteúdo do objeto pode ser dinamicamente criado pelo servidor ou pode ser armazenado estaticamente e disponibilizado pelo servidor. Para um servidor web, a plataforma do navegador cliente é indiferente. O servidor envia como resposta a informação solicitada e o navegador se encarrega de apresentá-la ao usuário. Esta característica é chamada de independência de plataforma e faz parte das aplicações para Internet de uma maneira geral (TEIXEIRA, 2004).

2.2. ALTA DISPONIBILIDADE

Um sistema de alta disponibilidade é aquele capaz de prover estruturas de detecção, recuperação e mascaramento de falhas, mantendo os serviços disponíveis, utilizando redundâncias de software e hardware. Nesse tipo de ambiente, mesmo na ocorrência de falha em um ou mais componentes, o funcionamento do sistema não deve ser afetado significativamente, assegurando a integridade de alterações realizadas durante a utilização do sistema. Se por qualquer motivo, um usuário não obtenha acesso total ou parcial ao sistema, o serviço é considerado como indisponível. O tempo de indisponibilidade é chamado *downtime* (ALVIM *et. al*. 2002, p.2).

A Figura 1 representa a estrutura montada para a Alta Disponibilidade.

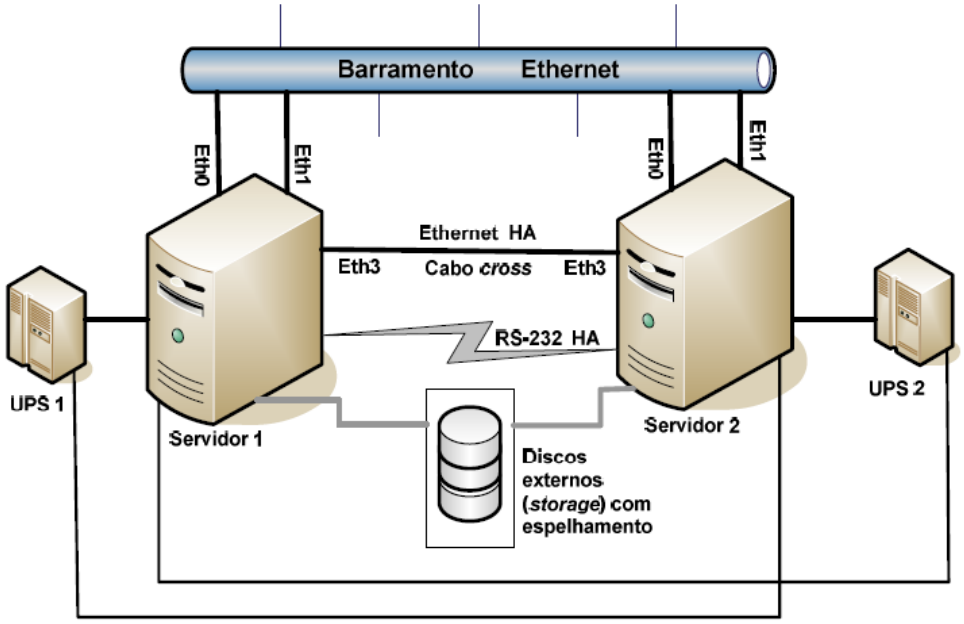


Figura 1 -Arquitetura de hardware em alta disponibilidade.

Fonte: Lopes, 2008.

A redundância de hardware permite a organização dos equipamentos de tal maneira que a ocorrência de falha de um equipamento não afetará no funcionamento do sistema como um todo.

A maneira como uma falha de equipamento será informada aos demais equipamentos pertencentes à estrutura é responsabilidade do software que os gerencia.

Para o trabalho foi utilizado como solução de software de alta disponibilidade a ferramenta chamada *Heartbeat,* que será explicada posteriormente.

2.3. ESCALABILIDADE

Define-se escalabilidade como a capacidade que um sistema ou componente suporta modificação sem tornar o ambiente indisponível. Ela não é medida com a velocidade do sistema. Escalabilidade e desempenho são dois conceitos diferentes. Um sistema pode disponibilizar alto desempenho e não ser escalável (HENDERSON, 2006).

A escalabilidade de um sistema indica a capacidade de processamento em uma crescente quantidade de requisições apropriadamente, de modo que o

usuário seja afetando minimamente e com baixo tempo de resposta (TANENBAUM e VAN STEEN, 2002).

Para projetar uma aplicação escalável, devem-se analisar os limites físicos que podem influenciar o sistema.

Em geral, a escalabilidade é obtida de duas formas:

- Verticalmente, através do incremento de recursos disponíveis em uma máquina, adicionando módulos de memória e adicionando processadores;
- Horizontalmente, que se baseia na a adição de novos nós ao grupo de servidores.

2.4. BALANCEAMENTO DE CARGA

Define-se como balanceamento de carga a capacidade de divisão de trabalho entre dois ou mais nós servidores. Pode-se também definir como o processo de distribuição de requisições de serviços em um agrupamento de servidores, (NATÁRIO, 2011).

Na visão de Costa (2009, p. 10), uma alternativa para solucionar o problema de aplicação sobrecarregada é a utilização do balanceamento de carga. Desde que seja aplicado em pontos considerados como “gargalos” no sistema, caso contrário não contribuirá na melhora de desempenho do sistema, podendo até ser prejudicial ao mesmo, devido às possíveis sobrecargas (*overheads*) do próprio mecanismo de balanceamento.

> “Quando não fazemos o balanceamento de carga entre servidores que possuem a mesma capacidade de resposta a um cliente, começamos a ter problemas, pois um ou mais servidores podem responder a requisição feita e a comunicação fica prejudicada. Por isso devemos posicionar o elemento que fará o balanceamento entre os servidores e os usuários e configurá-lo para isso, entretanto podemos colocar múltiplos servidores de um lado que, para os clientes, eles parecerão ser somente um endereço. Balanceamento de carga é mais que um simples redirecionamento do tráfego dos clientes para outros servidores.” (PITANGA, 2003, p. 2).

Os três algoritmos para encaminhamento de requisição aos servidores são descritos a seguir:

- ***Round Robin* (Escalonamento Circular)**: direciona a requisição ao próximo servidor disponível em forma circular, se o ambiente dispõe de 3 servidores, a primeira requisição vai ao servidor um, a segunda ao servidor dois, a terceira ao servidor três, e a quarta retorna ao servidor um, e assim sucessivamente.
- ***Least Connections* (Menos Conexões):** direciona a requisição para o servidor com menor número de conexões ativas. O mesmo necessita controlar o número de conexões e encaminhar ao balanceador de carga.
- ***Weighted Fair* (Peso Justo):**Essa técnica controla as conexões ativas, de modo que o administrador define um peso a cada servidor, para informar qual a capacidade de processamento. Então, um servidor A com poder de processamento cinco vezes maior que o servidor B, receberá a requisição mesmo que o número de conexões ativas seja o mesmo.

2.5. *MAINFRAMES*

Segundo Ruschel (2009), os *mainframes* foram criados na década de 40 e foram introduzidas no mercado durante o desenvolvimento de empresas como a IBM, HP e UNISYS.

Ao decorrer dos anos, as empresas de médio e grande porte foram desenvolvendo necessidade excessiva com relação à capacidade de armazenamento, disponibilidade e processamento de informações e também conectividade a equipamentos periféricos como impressoras e “*terminais burros"*. Devido a essas necessidades foram desenvolvidos os *mainframes*, também conhecidos como supercomputadores, capazes de atender e suportar todas essas necessidades.

Conseqüentemente por se tratarem de supermáquinas, somadas a avançada tecnologia agregada, os *mainframes* são equipamentos de custo elevado, de maneira que nem todas as empresas, podem adquirir tal recurso.

2.6. *CLUSTER*

Na visão de Pitanga (2003), é considerado *Cluster* um sistema formado por dois ou mais computadores ou sistemas, onde trabalham juntos para executar determinadas tarefas, de tal forma que o usuário tenha a impressão de ser um único sistema. Confiabilidade, desempenho e distribuição de carga são as principais características de um *Cluster*. Sendo assim, podemos chamar de *cluster* um agrupamento de máquinas sincronizadas, responsáveis por um determinado serviço.

Basicamente as estruturas de *clusters* são divididas em duas categorias:

- *Cluster* de Alta Disponibilidade (HA - *High Availability*) desenvolvido para manter um serviço o maior tempo possível de maneira segura;
- *Cluster* de Alto Desempenho Computacional (HPC – *High PerformanceComputing*) desenvolvido para possibilitar grande poder computacional, para grande capacidade de processamento de dados onde um único computador não disponibilizaria o processamento necessário para as tarefas (PITANGA, 2004).

2.7. GNU/LINUX / *SOFTWARE* LIVRE

Segundo Reis (2003, p.17) em 1991, um estudante da universidade de Helsinki chamado Linus Torvalds, começou a desenvolver um núcleo de Sistema Operacional batizado de Linux. Atualmente o Linux é considerado o mais importante exemplo de um software livre. O maior diferencial do Linux é a “abertura” do seu conteúdo do processo de desenvolvimento.

> “Software Livre é qualquer software cuja licença garanta ao seu usuário liberdades relacionadas ao uso, alteração e redistribuição. Seu aspecto fundamental é o fato do código-fonte estar livremente disponível para ser lido, estudado e/ou modificados por qualquer pessoa interessada.” (REIS, 2003 p. 14).

2.8. FERRAMENTAS UTILIZADAS

2.8.1. **Heartbeat**

Na visão de Agarwalet al(2010, p. 79-80), a aplicação *heartbeat*vem da mesma ideia dos batimentos cardíacos, em momentos importantes do sistema a aplicação registra sinais disparados num certo intervalo de tempo. Além disso, a

interface possui duas métricas principais que possibilita medir o desempenho mediante determinada frequência e/ou latência pré-determinada. O primeiro refere-se ao tempo necessário para registrar um segundo, enquanto que o segundo é o decorrente de quando o sinal é registrado em um pedido para que possa ser lido por um processo externo.

2.8.2. **HAProxy**

Segundo Kaushal e Bala (p. 55, 2011) *HAProxy* é uma aplicação utilizada por diversas empresas para balanceamento de carga e *failover* de servidores. *HAProxy* é um *Proxy* que atua na intercepção de requisições entre a aplicação cliente e a servidor, possui um balanceador de carga para distribuir as requisições em um conjunto de servidores *web*. O *HAProxy* funciona também como um sistema tolerante a falhas, sempre que uma máquina apresentar mau funcionamento ou seja removida do *cluster*, ela será excluída do balanceador até que volte a funcionar ou até mesmo substituída.

2.8.3. **Apache**

O apache é um servidor web livre e de código aberto, sendo o mais utilizado atualmente no mundo, acompanhado de perto do IIS da Microsoft.

> "A palavra Apache significa A PAtCHy, e foi utilizada na nomeação de um servidor *web* considerando um código disponibilizado juntamente com uma série de arquivos *patch* (um arquivo que tem apenas as diferenças entre duas versões). Para muitos desenvolvedores, porém, a palavra faz referência aos nativos americanos, ou seja, os índios Apache. Entre as principais características do *Apache*, podemos dizer que é altamente configurável, pode ser executado em diferentes plataformas, é flexível, está sempre em desenvolvimento para a inclusão dos protocolos mais atualizados (por exemplo, HTTP), fornece o código-fonte completo e não possui licenças restritivas, pode ser configurado para diferentes funções, é composto de módulos, cada um implementando uma característica diferente e aumentando a funcionalidade do servidor, além de várias outras características." (GOMES et. al. 2001).

2.8.4. **Remastersys**

Segundo Majo (2012, p. 28),*Remastersys*é uma ferramenta que foidesenvolvida pelo engenheiro canadense Tony Brijeski. O objetivo inicial da

aplicação era de criar um *backup* completo do sistema operacional. Porém, ao decorrer de suas atualizações acabou permitindo a criação de novas distribuições de sistemas operacionais livres, e também que o sistema operacional seja totalmente customizado, com todos os programas, acessórios e configurações desejadas.

O Remastersyspermite gerar um arquivo *iso.* Como citado por Campos (2006), este arquivo possui uma cópia completa do conteúdo de um CD/DVD-ROM, possibilitando que seja efetuada a gravação posteriormente.

2.8.5. **JMeter**

O *Apache Jmeter*é uma ferramenta que pode ser utilizada para testar aplicações que fazem uso de servidores HTTP ou FTP. É fundamentado em *Java* e também altamente expansível mediante uma API fornecida. Um típico teste do *Jmeter*envolve a criação de um laço e um grupo de *threads*. O laço simula requisições sequenciais para o servidor com um atraso programado, enquanto que o grupo de *thread*está projetado para simular uma carga simultânea. O *Jmeter* fornece uma interface de usuário e também dispõe de uma API que permite fazer testes baseados em um aplicativo *Java*(NEVEDROV, 2006, p. 1).

3. DESENVOLVIMENTO DO CLUSTER E GERAÇÃO DA ISO PARA DISTRIBUIÇÃO

Neste capítulo são apresentados os equipamentos que compõem o *Cluster* de testes, os métodos utilizados para instalação e a sequência de configuração dos programas. Também serão apresentados os conceitos de partições de disco utilizados para armazenamento de arquivos no *Debian GNU/LINUXSqueeze.*

3.1. VISÃO GERAL

O *Cluster* proposto é dividido em duas partes principais: Os Balanceadores de Carga, responsáveis por receber as requisições web da internet e os Servidores web, responsáveis por processar as informações e responder ao pedido de serviço. A Figura 2 ilustra a localização de cada parte do *Cluster*.

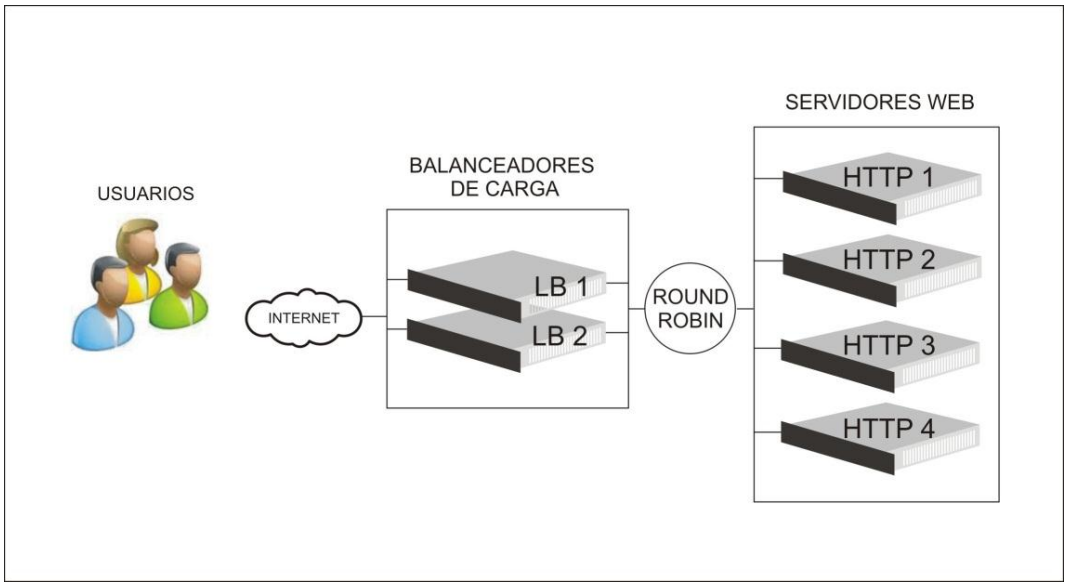


Figura 2- *Cluster* proposto e suas partes.
Fonte: Autoria Própria.

Quando um usuário acessa sites por meio de um navegador, ele tem a impressão de que o site está armazenado apenas em um servidor web. Isto é possível pela transparência na navegação que o cluster disponibiliza, quando de

fato há dois ou mais servidores web. A transação básica ocorre da seguinte maneira:

1. O usuário conecta-se ao balanceador de carga via interface virtual (servidor virtual) por meio de um navegador web de seu computador;
2. A conexão é estabelecida com o balanceador de carga e é tomada a decisão de acordo com o algoritmo configurado qual é o melhor servidor web para encaminhar o pedido. Então muda-se o IP de destino e faz-se o encaminhamento;
3. O servidor web recebe a conexão, processa e responde ao cliente.
4. O usuário recebe o pacote de retorno, como se ele viesse do servidor virtual, a execução do processo continua.

O algoritmo comumente utilizado é o *Round Robin,* onde o balanceador de carga percorre uma lista com todos os servidores *web* disponíveis, de cima para baixo, direcionando cada nova conexão para o próximo servidor web, até o fim da lista, e então retorna ao início. O algoritmo simples e previsível prevê que todas as conexões terão a mesma prioridade e disponibilidade de recursos, o que nem sempre é verdade, devido a possível heterogeneidade de hardware utilizado (SALCHOW, 2010).

3.2. MONTAGEM FÍSICA DO CLUSTER

As especificações dos componentes de hardware utilizados na construção do *cluster* seguem as Tabelas 1 e 2.

NOME	FUNÇÃO	PROCESSADOR	DISCO RIGIDO	PLACA MÃE	MEMÓRIA
LB1	BALANCEADOR DE CARGA	Intel® Core™2 Quad Q8200 (4M Cache, 2.33 GHz, 1333 MHz FSB)	SEAGATE ST3500413AS 500Gb SATA 7200rpm	Asus® P5QPL-AM	2x Corsair CM2X2048-8500C5D (4Gb)
LB2	BALANCEADOR DE CARGA	Intel® Pentium® E2180 (1M Cache, 2.00 GHz, 800 MHz FSB)	SEAGATE ST3500413AS 500Gb SATA 7200rpm	Asus® P5QPL-AM	2x Kingston KVR667D2N5K2/2G (4Gb)
HTTP 1	SERVIDOR WEB	Intel® Pentium® D 915 (4M Cache, 2.80 GHz, 800 MHz FSB)	SAMSUNG HD502HJ/SRA 500Gb SATA 7200rpm	Asus® P5G	2x Kingston KVR667D2N5K2/2G (4Gb)
HTTP 2	SERVIDOR WEB	Intel® Pentium® 4 519J(1M Cache, 3.06 GHz, 533 MHz FSB)	MAXTOR STM3160813AS 160Gb SATA 7200rpm	Intel® Desktop Board D101GGC	2x Kingston KVR400X64C3A/512 (1Gb)
HTTP 3	SERVIDOR WEB	Intel® Celeron® D 310 (256K Cache, 2.13 GHz, 533 MHz FSB)	MAXTOR STM3160813AS 160Gb SATA 7200rpm	Gigabyte® GA-8VM533	2x Kingston KVR667D2N5K2/2G (1Gb)
HTTP 4	SERVIDOR WEB	Intel® Celeron® D 310 (256K Cache, 2.13 GHz, 533 MHz FSB)	MAXTOR STM3160813AS 160Gb SATA 7200rpm	Gigabyte® GA-8VM533	2x Kingston KVR667D2N5K2/2G (1Gb)

Tabela 1 - Componentes de hardware utilizados nos servidores.

Fonte: Autoria Própria.

A Tabela 1 representa os equipamentos utilizados para o *Cluster*.Os dois melhores computadores foram alocados para a tarefa de balanceamento de carga, visto que esta atividade é a que coordena o funcionamento do *Cluster*.Os computadores com menor poder de processamento foram alocados para a tarefa de servidores web, visto que haverá a divisão de tarefas.

TABELA DE EQUIPAMENTOS DE REDE			
EQUIPAMENTO	MARCA	VELOCIDADE/ DE PORTAS	NUMERO DE PORTAS
SWITCH	INTELBRAS	100Mbps	8
ROUTER	TPLINK	100Mbps	5

Tabela 2 - Componentes de hardware de rede utilizados.

Fonte: Autoria Própria.

Onde:

O switch tem a função de conectar os computadores formando uma rede.

O router tem a função de conectar a rede interna com a internet.

A organização dos servidores que constituem o *Cluster* pode servisualizada na Figura 3.

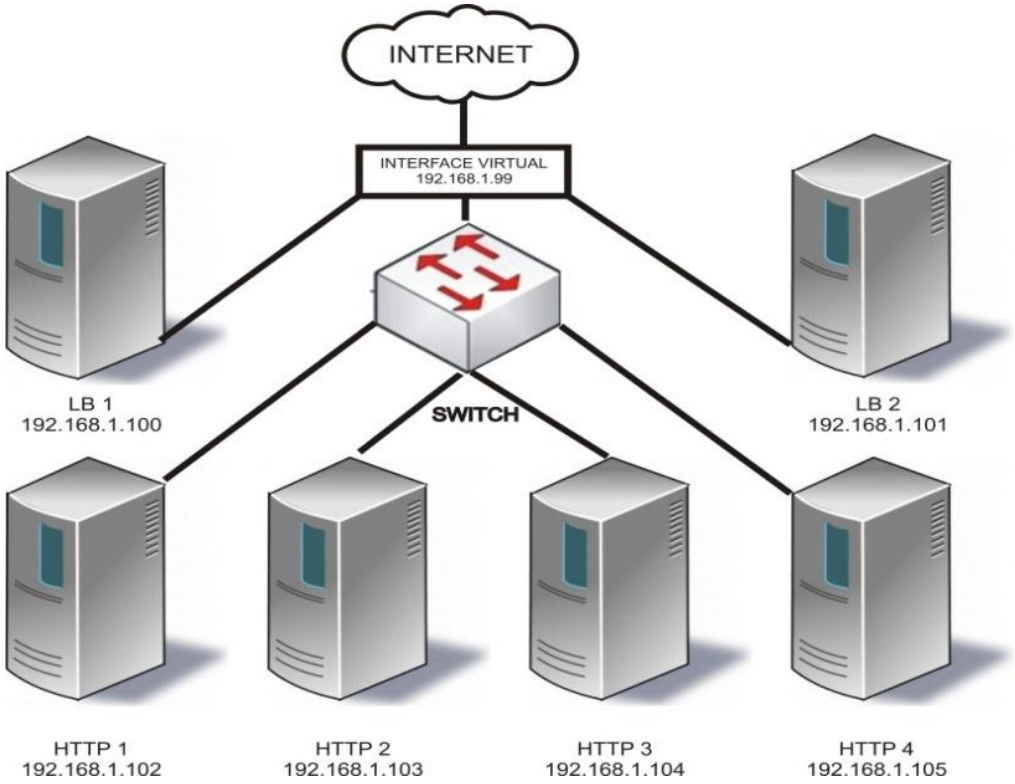


Figura 3 - Visão geral do *Cluster*

Fonte: Autoria própria.

3.2.1. **Construção Da Estrutura Física**

Inicialmente uma estrutura é construída para acomodar os servidores, de modo a diminuir o espaço e o calor gerado. Uma base em acrílico é utilizada para a fixação de cada kit composto por placa mãe, memória, processador, fonte e disco rígido. Ao total são agrupados seis computadores com esta disposição.

3.2.2. **Sistema Operacional Utilizado**

O sistema operacional escolhido é o Debian GNU/Linux Squeeze, por este ser a versão 6.0.6, é gratuito e possui código fonte aberto. Como o objetivo do trabalho é criar uma distribuição de sistema operacional, o sistema é então instalado somente em duas máquinas distintas e depois replicado nas demais utilizando o CD criado.

3.3. CONFIGURAÇÃO DO SOFTWARE PARA O CLUSTER

Esta seção trata da instalação do Debian GNU/Linux Squeeze e das ferramentas necessárias para a configuração dos Balanceadores de carga e dos

servidores web. Também será explicado sobre os arquivos de configuração necessários para o funcionamento das ferramentas**.**

3.3.1. **Instalação do Debian GNU/Linux Squeeze**

A instalação do Debian GNU/Linux Squeeze é utilizada a distribuição 6.0.6, obtida pela internet no site do Debian.Utiliza-se o modo gráfico de instalação, pois este disponibiliza uma interface intuitiva, alem de oferecer dicas sobre os componentes a serem utilizados. O procedimento é basicamente concordar com as predefinições contidas na mídia de instalação.

O Debian GNU/Linux Squeeze dispõe de *drivers* de dispositivos para a grande maioria dos equipamentos disponíveis no mercado.

3.3.2. **Identificação e Arquivos de Configuração de Rede**

Para que os computadores que compõem o cluster funcionem em um ambiente de rede e todos sejam acessíveis, é necessário configurar uma série de arquivos. Cada um dos arquivos possuem parâmetros para customizar os serviços necessários ao funcionamento do cluster.

3.3.2.1. Configurando o arquivo *hostname*

O arquivo *hostname* é o local onde as configurações do nome da máquina são armazenadas. Ele é utilizado para associar um nome ao endereço IP.A utilização de um *hostname* facilita a identificação do computador, para administradores de sistema quanto para outros softwares, pois ao alterar o endereço IP da máquina, o seu nome (*hostname*) continuará o mesmo, pois cada servidor possui um nome único na rede.

3.3.2.2. Configurando o arquivo *interfaces*

Existem duas formas para que os equipamentos estabeleçam conexão e acesso à rede: atribuição de endereçamento IP (*InternetProtocol*) de forma dinâmica fornecido pelo *DynamicHostConfigurationProtocol* (DHCP), ou com atribuição de endereço IP estático (TORRES, 2001).

A configuração estática garante que o adaptador de rede “eth0” comunique-se utilizando o endereço IP configurado no arquivo interfaces.

Define-se um endereço estático de IP para cada servidor do cluster, para garantir que não haja o risco de que uma atualização de servidor DHCP execute a reconfiguração automática dos endereços registrados para cada máquina.

Na Figura 4, podemos visualizar as configurações efetuadas no arquivo *interfaces.*

```
vi /etc/network/interfaces

#O conteúdo do arquivo deverá ser substituído por

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface

allow-hotplug eth0
iface eth0 inet static
address 192.168.1.100
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.254

#NetworkManager
#iface eth0 inet dhcp

#O valor do campo address deverá ser alterado para cada
#maquina seguindo a sequencia:
#192.168.1.100 lb1
#192.168.1.101 lb2
#192.168.1.103 http1
#192.168.1.104 http2
```

Figura 4 - Imagem da configuração do arquivo *interfaces.*

Fonte: Autoria Própria.

Cada adaptador é instalado em um computador e recebe um nome único. Esse nome é associado ao Endereço físico do adaptador (*MAC ADDRESS*).

Ao substituir ou adicionar um adaptador, onovo adaptador será chamado de “eth1”.Para que o novo adaptador receba as configurações de endereço IP, poderão ser efetuadas duas mudanças.

No arquivo interfaces, altera se os campos “eth0” para “eth1” ou deve-se efetuar a mudança do nome de interface para que este receba as configurações

definidas para “eth0”. Essas configurações são efetuadas no arquivo 70-persistent-net.rules. Representada na Figura 5 logo abaixo.

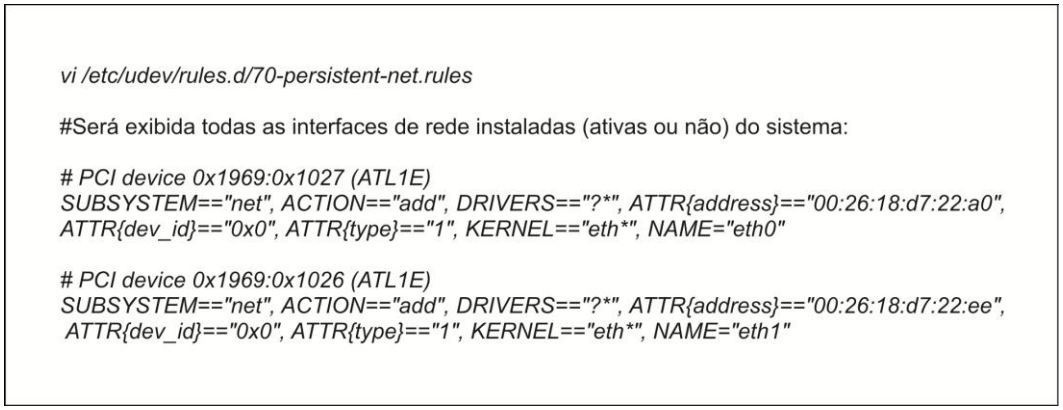


Figura 5 - Imagem do arquivo70-persistent-net.rules.
Fonte: Autoria Própria.

Deve-se excluir o primeiro conjunto de dados formado pelo “eth0” e renomeia-se o campo chamado “eth1” para “eth0”.

3.4. CONFIGURAÇÃO DO SERVIDOR WEB e B.D.

Na configuração do servidor web são instalados e configurados os softwares Apache, Open SSH, MySQL e o phpMyAdmin.

3.4.1. **Configuração do Servidor Apache**

Após efetuar o download dos arquivos do apache, é realizada a instalação do software para que o computador torne-se um servidor web. Após a instalação, é necessário configurar o arquivo de *log* do apache.

É criado um arquivo chamado *check.txt* utilizado para a gerência do *HAProxy*, e para evitar que o log receba informações irrelevantes sobre este arquivo, o mesmo é modificado para ignorar tais entradas. Como na Figura 6.

```
vi etc/apache2/apache2.conf

#O inserir o trecho de código:

[...]
#LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\"
 \"%{User-Agent}i\"" combined
[...]

touch /var/www/check.txt

vi /etc/apache2/sites-available/default

[...]
SetEnvIf Request_URI "^/check\.txt$" dontlog CustomLog
/var/log/apache2/access.log combined env=!dontlog
[...]
#Qualquer outra diretiva Customlog deverá ser comentada.

/etc/init.d/apache2 restart
```

Figura 6 - imagem de configuração do arquivo apache2.conf.
Fonte: Autoria Própria.

Para um servidor web ser membro de um ambiente gerenciado pelo HAProxy, um arquivo deve ser criado para servir de sinalizador de estado para que possa ser monitorado. Logo, é criado um arquivo chamado check.txt, que é atualizado periodicamente com informações do status de máquina.

Caso o arquivo check.txt seja excluído, o HAProxy ignora o servidor em questão e não o envia mais requisições. Este artifício é utilizado para efetuar manutenção no servidor e evitar a geração de um estado inconsistente de dados.

3.4.2. **Instalação do *MySQL***

Após executar o comando de instalação, Inicia-se o assistente de instalação do *MySQL*. É importante inserir uma senha de administrador do banco de dados. No caso do experimento, a senha é ”1234”. Esta senha é utilizada para configuração de acesso ao banco de dados. O acesso ao *MySQL* é feito através do seguinte comando:

mysql -u root -p

Onde root é o nome do usuário e após confirmar, será solicitado a senha do root que foi configurada durante a instalação do *MySQL*.

3.4.3. **Instalação do *PhpMyAdmin***

O *phpMyAdmin* é uma ferramenta desenvolvido em PHP, utilizada para administrar o *MySQL*. Várias operações são suportadas ao manipular banco de dados gerenciado pelo *MySQL*. Uma lista de operações utilizadas com maior frequência é suportada pela interface do usuário e também possui suporte a execução direta de declaração SQL (DELISLE, 2009).

Após efetuar a instalação do phpMyAdmin, executa-se o comando:

ln –s /usr/share/phpmyadmin /var/www/phpmyadmin

Este comando copia os arquivos necessários para que o ambiente de administraçãoweb seja disponibilizado.Como apresentado na figura 7 a seguir.

A Figura 7 apresenta a interface do phpMyAdmin.

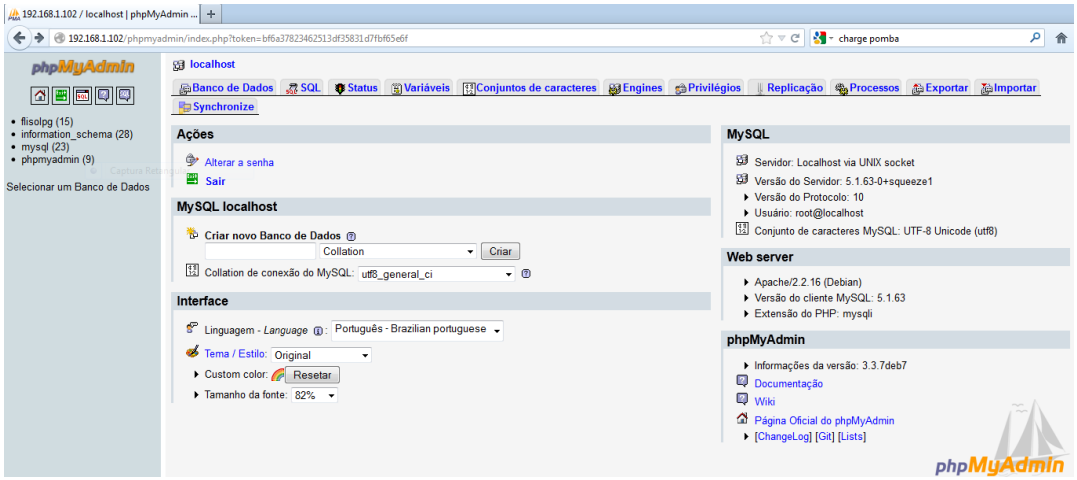


Figura 7 - Imagem da interface do phpMyAdmin.

3.4.4. **Instalação do PHP5**

PHP é uma linguagem web baseada em scripts, possui código fonte aberto. Ela é processada no servidor e embutida em HTML, que é compatível com

todos os grandes servidores. O PHP permite incorporar fragmentos de código em páginas HTML normais (CONVERSE et. al., 2004).

3.4.5. **Configuração do Site Hospedado**

Para efeito de testes do ambiente, o site criado para o evento de software livre FLISOL 2012, da UTFPR Ponta Grossa, foi configurado nos servidores web.O site foi desenvolvido pelos acadêmicos do quinto período do curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Copia-se a pasta do projeto web gerado para a seguinte pasta:

/var/www

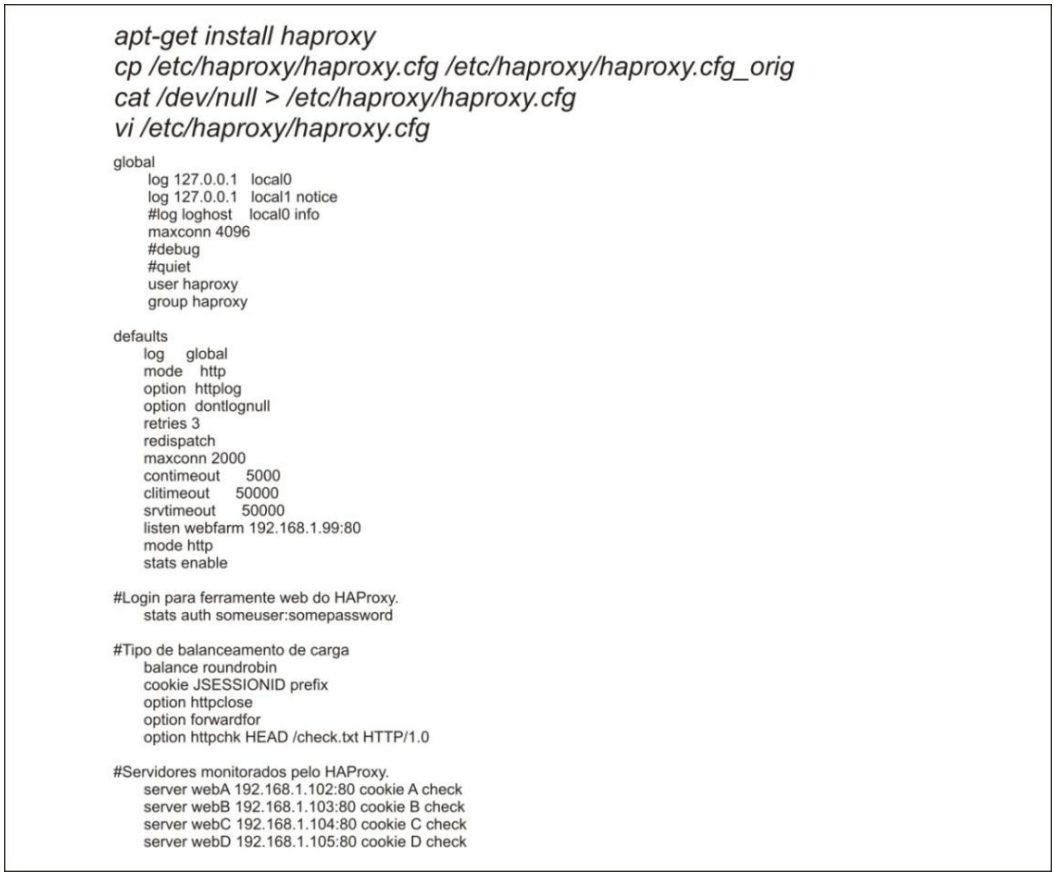
Com isto, conclui-se a configuração básica de um servidor web com conteúdo configurado.

3.5. CONFIGURAÇÃO DO BALANCEADOR DE CARGA

Para o funcionamento do balanceador de carga é necessária a instalação dos programas *Heartbeat* e *HAProxy,*juntamente com seus respectivos arquivos de configuração.

3.5.1. **Configuração do *HAProxy***

Após efetuar a instalação do HAProxy, é criado e editado o arquivo *haproxy.cfg,* como apresentado na Figura8.



```
apt-get install haproxy
cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg_orig
cat /dev/null > /etc/haproxy/haproxy.cfg
vi /etc/haproxy/haproxy.cfg

global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        #log loghost    local0 info
        maxconn 4096
        #debug
        #quiet
        user haproxy
        group haproxy

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        retries 3
        redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000
        listen webfarm 192.168.1.99:80
        mode http
        stats enable

#Login para ferramente web do HAProxy.
        stats auth someuser:somepassword

#Tipo de balanceamento de carga
        balance roundrobin
        cookie JSESSIONID prefix
        option httpclose
        option forwardfor
        option httpchk HEAD /check.txt HTTP/1.0

#Servidores monitorados pelo HAProxy.
        server webA 192.168.1.102:80 cookie A check
        server webB 192.168.1.103:80 cookie B check
        server webC 192.168.1.104:80 cookie C check
        server webD 192.168.1.105:80 cookie D check
```

Figura 8 - configuração do arquivo haproxy.cfg**.**

Fonte: Autoria Própria.

3.5.2. **Configurando o Arquivo haproxy.cfg**

Cria-se a o arquivo haproxy.cfg. O arquivo é composto por vários parâmetros, mas existem apenas três configurações mais importantes para o funcionamento: o tipo de balanceamento, configurado como *Round Robin*, a listagem dos servidores monitorados e o usuário e senha da ferramenta web, que permite monitorar visualmente o status de todos os servidores *web*.

Alteram-se os outros parâmetros somente para customizar o serviço, em busca de melhorias.

3.5.3. **Configurando O Arquivo *haproxy***

Cria-se também o arquivo *HAProxy* que conterá a configuração de uma *flag*para inicializar o script do *HAProxy*, como mostra a Figura 9.

```
vi /etc/default/haproxy

# Configuração da flag ENABLED= 1 para inicializar o script do haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

Figura 9 - conteúdo do arquivo haproxy.

Fonte: Autoria Própria.

A função do *HAProxy*é monitorar as requisições na interface virtual 192.168.1.99. A partir deste ponto o balanceamento de carga estará configurado**.**

3.5.4. **Configuração do *Heartbeat***

Após instalar o *Heartbeat*, adicionam-se ao arquivo *sysctl.conf* as informações "*net.ipv4.ipnonlocal_bind=1"* para que o *HAProxy* conecte-se ao endereço IP virtual compartilhado.

Criam-se três arquivos de configuração para o *Heartbeat*: o *authkeys, ha.cf e o haresources. O outhkeys*é responsável por conter a configuração de senhas, permitindo a comunicação entre os nós de forma segura. O *ha.cf* é responsável por especificar a porta e interface física de comunicação e troca de *traps*para verificação dos serviços. O *haresourses*é responsável por conter a configuração dos IPsVIPs do cluster e eleger o nó principal (Rôlla, p. 16-18).

3.5.5. **Configuração do Arquivo *authkeys***

O arquivo *authkeys* armazena a configuração do método de autenticação entre os balanceadores de carga, para garantir que os nós se comuniquem de forma segura.

```
vi /etc/ha.d/authkeys

   auth 3
   somerandomstring md5

chmod 600 /etc/ha.d/authkeys
```

Figura 10 - conteúdo do arquivo authkeys.

A stringsomerandomstring é uma sequência de caracteres utilizada como senha que será usada como modelo, para autenticação dos serviços *daemonsHeartbeat* LB1 e LB2. Pode-se utilizar uma sequência própria, no exemplo será utilizado a md5.

O arquivo a*uthkeys* deverá ser lido somente pelo usuário root por motivos de segurança do sistema, então a permissão chmod 600 (concede permissão de leitura e gravação apenas para o proprietário do arquivo) será aplicada ao arquivo.

3.5.6. **Configuração do *Arquivo ha.cf***

Armazenam-se no arquivo ha.cf todos os parâmetros de transferência dos pacotes entre o balanceador de carga*master* e o *slave*.O balanceador é responsável por monitorar a disponibilidade e evitar a perda de desempenho da rede.

vi /etc/ha.d/ha.cf

```
# Intervalo em segundos entre os pings
keepalive 2
# Intervalo em segundos para declarar uma maquina inativa
deadtime 30
# Tempo para notificar no log
warntime 10
# Permitir inicialização em todos os nodes
initdead 60
# Interface que fara broadcast de verificação
bcast eth0
# Configuração de Nodes
node lb1.example.com
node lb2.example.com
# Especifica o uso de um gerenciador de recursos externo
crm off
# Volta do node primário caso haja falha e depois volte a responder
auto_failback on
# Caminho do arquivo de Debug
debugfile /var/log/ha-debug.log
# Caminho do arquivo de Log
logfile /var/log/ha-log.log
```

Figura 11 - conteúdo do arquivo ha.cf.

3.5.7. **Configuração do Arquivo haresources:**

São configurados no arquivo *haresources* os parâmetros de todos os recursos que serão iniciados quando o balanceador de carga secundário assumir, na ocorrência de falha do primário.

O arquivo *haresources*é responsável também em apontar o endereço IP da interface virtual.

3.6. CRIAÇÃO DA NOVA DISTRIBUIÇÃO *DEBIAN GNU/LINUX SQUEEZE*

Paraser criada uma nova distribuição contendo as configurações realizadas até aqui, de maneira que já esteja tudo integrado ao instalar o sistema operacional em outro servidor, é necessário um software que consiga “salvar” o estado atual do sistema operacional e consiga criar a imagemISOdo mesmo. O *Remastersys* realiza tal tarefa de maneira eficiente.

3.6.1. **Configurando o*Remastersys***

Para a instalação do software *Remastersys*, será necessário atualizar alguns componentes do sistema operacional. No arquivo de fontes de instalação *sources.list* deve-se adicionar a fonte para o *download* da ferramenta, representada na Figura 12.

```
vi /etc/apt/sources.list


# Remastersys Squeeze
deb http://www.remastersys.com/repository squeeze/
```

Figura 12 - Fonte para download do Remastersys.

Para adicionar o *Remastersys* e as ferramentas necessárias para a criação da imagem ISO*,* executa-seos comandos apresentados na Figura 13.

```
# apt-get update
# apt-get install remastersys
#apt-get install module-assistant
#apt-get install openafs-modules-source
```

Figura 13 - comandos da ferramenta Remastersys para a criação da imagem ISO.

Após a instalação,as ferramentas gráficas do *Remastersys*são adicionadas na guia *sistema* do Sistema Operacional.

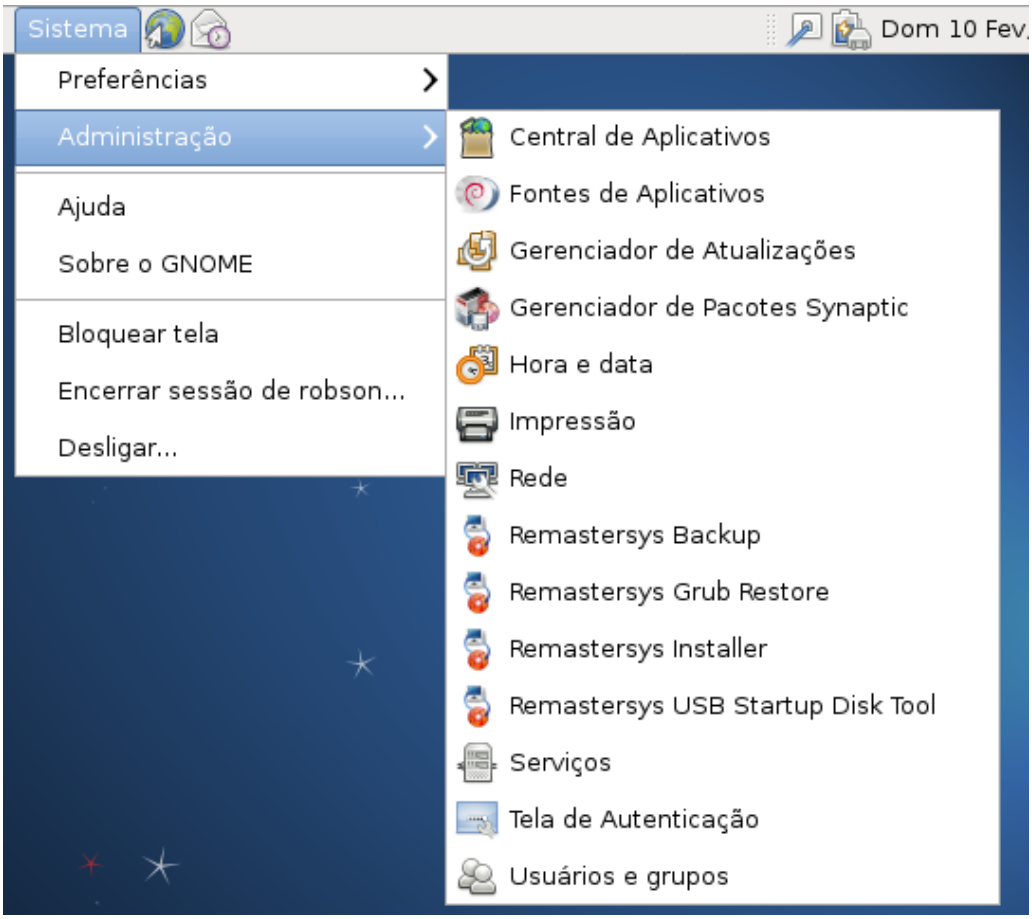


Figura 14 - Recursos gráficos da ferramenta remastersys no menu do Sistema Operacional.

Fonte: Autoria Própria.

3.7. CRIAÇÃO DA DISTRIBUIÇÃO

O processo de criação da distribuição pode ser feito por comandos *shell* ou via interface gráfica. É utilizado o processo gráfico pela agilidade disponível. No *menuSistema-Administração-RemastersysBackup* é possível escolher entre as modalidades disponíveis.

4. RESULTADOS

A rotina de testes para identificar funcionalidades do *Cluster* foi planejada e executada com o auxilio da ferramenta *JMeter*.O *JMeter* possui uma função chamada Servidor HTTP *Proxy*. Esta função permite que as operações de navegação em um site sejam registradas, de modo que seja possível reproduzi-las de maneira automatizada. Para configurar a funcionalidade, é necessário primeiramente, criar um grupo de usuários chamado Usuários *HACluster*. Após criar o grupo de usuários, são selecionados na área de trabalho do JMeter, nos elementos que não são de testes a funcionalidade Servidor HTTP *Proxy*. Ela funciona como um servidor *Proxy* de internet, mas com o intuito de coletar os dados para os testes. Na guia controlador alvo seleciona-se o grupo de usuários criadoPara a coleta de dados, configura-se o endereço de *Proxy* do *JMeter* em um navegador web. Para efetuar os testes de desempenho, utilizamos o website do FLISOL (<http://flisolpg.com.br/>) Uma cópia idêntica foi hospedada em cada um dos quatro servidores web do cluster. A seguir é demonstrada a lista de todos os arquivos capturados pelo *Proxy,* que são utilizados para o teste:

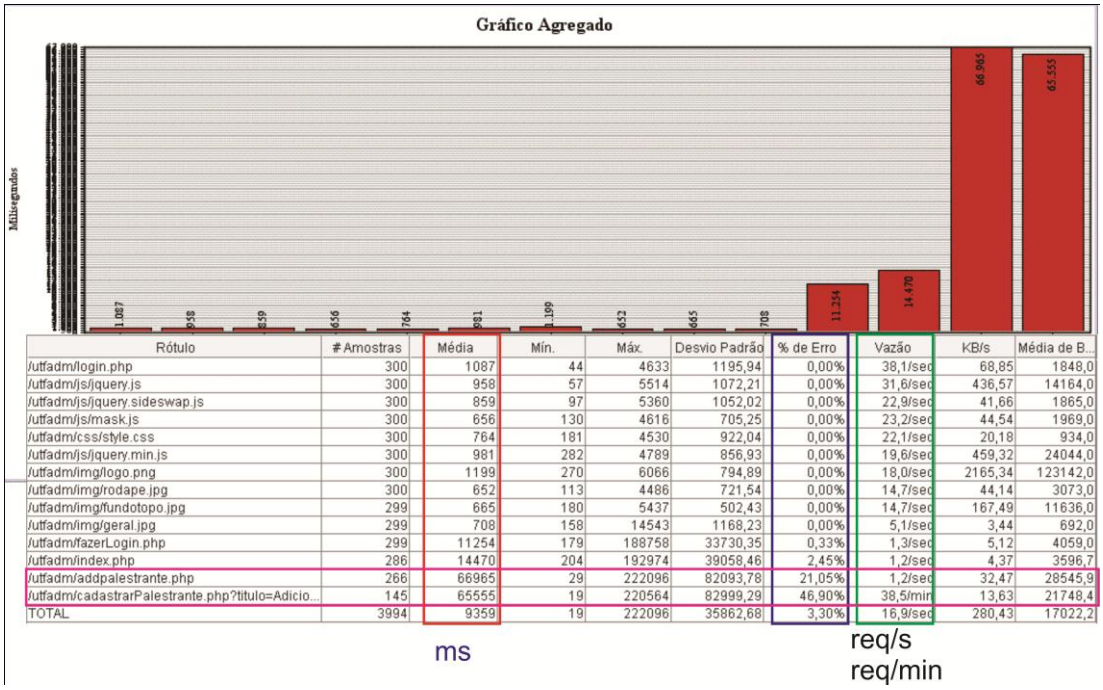
/
/utfadm/css/style.css
/utfadm/js/jquery.sideswap.js
/utfadm/js/mask.js
/utfadm/js/jquery.js
/utfadm/js/jquery.min.js
/utfadm/img/fundotopo.jpg
/utfadm/img/geral.jpg
/utfadm/img/rodape.jpg
/utfadm/img/logo.png
/utfadm/img/icone.ico
/utfadm/fazerLogin.php
/utfadm/index.php
/utfadm/img/lista.png
/utfadm/addpalestrante.php
/utfadm/cadastrarPalestrante.php?titulo=Adicionar&codigoParticipante=

/utfadm/addpalestra.php
/utfadm/cadastrarPalestra.php
/utfadm/addoficina.php
/utfadm/cadastrarOficina.php

O processo de navegação seguiu sequência a seguir:
Login;
Cadastrar palestrante (usuário: João Silva, instituição: UTFPR);
Cadastrar palestrante (usuário: Maria José, instituição: UTFPR);
Adicionar palestra (tema: HACluster,local: auditório palestrante:João);
Adicionar palestra (tema: Linux,local: sala 2,palestrante: Maria José);
Noticia evento (Vencedor do FretsOnFire);
Oficina (palestrante: João,tema: redes,local: sala 3);
Logoff.

Os testes foram divididos em duas etapas, 300 usuários e um servidor único e 300 usuários no cluster.

Cada etapa do teste foi repetida por um período de dois minutos. O primeiro teste foi realizado no servidor HTTP1, por se tratar do computador com melhor desempenho disponibilizado entre os servidores web.



Rótulo	# Amostras	Média	Mín.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Média de B...
/utfadm/login.php	300	1087	44	4633	1195,94	0,00%	38,1/sec	68,85	1848,0
/utfadm/js/jquery.js	300	958	57	5514	1072,21	0,00%	31,6/sec	436,57	14164,0
/utfadm/js/jquery.sideswap.js	300	859	97	5360	1052,02	0,00%	22,9/sec	41,66	1865,0
/utfadm/js/mask.js	300	656	130	4616	705,25	0,00%	23,2/sec	44,54	1969,0
/utfadm/css/style.css	300	764	181	4530	922,04	0,00%	22,1/sec	20,18	934,0
/utfadm/js/jquery.min.js	300	981	282	4789	856,93	0,00%	19,6/sec	459,32	24044,0
/utfadm/img/logo.png	300	1199	270	6066	794,89	0,00%	18,0/sec	2165,34	123142,0
/utfadm/img/rodape.jpg	300	652	113	4486	721,54	0,00%	14,7/sec	44,14	3073,0
/utfadm/img/fundotopo.jpg	299	665	180	5437	502,43	0,00%	14,7/sec	167,49	11636,0
/utfadm/img/geral.jpg	299	708	158	14543	1168,23	0,00%	5,1/sec	3,44	692,0
/utfadm/fazerLogin.php	299	11254	179	188758	33730,35	0,33%	1,3/sec	5,12	4059,0
/utfadm/index.php	286	14470	204	192974	39058,46	2,45%	1,2/sec	4,37	3596,7
/utfadm/addpalestrante.php	266	66965	29	222096	82093,78	21,05%	1,2/sec	32,47	28545,9
/utfadm/cadastrarPalestrante.php?titulo=Adicio...	145	65555	19	220564	82999,29	46,90%	38,5/min	13,63	21748,4
TOTAL	3994	9359	19	222096	35862,68	3,30%	16,9/sec	280,43	17022,2

ms

req/s
req/min

Figura 15 - gráfico de *requests* por segundo.

Fonte: Autoria Própria.

O gráfico da Figura 15 demonstra os dados em milissegundos, na coluna vermelha, o tempo médio de resposta para cada operação efetuada no servidor. As funções de adicionarPalestra e adicionarPalestrante, representadas no retângulo rosa, são as requisições com maior tempo de resposta. Elas executam inserção no banco de dados e também estão entre os itens com maior número de erros, pois o servidor não consegue executar com sucesso todas as modificações. As requisições a conteúdo estático mantêm uma linearidade no tempo de execução para este volume de acessos simultâneos. O tempo médio total foi contabilizado em 9359ms.

A Figura 16 evidencia a queda do servidor com 300 usuários.

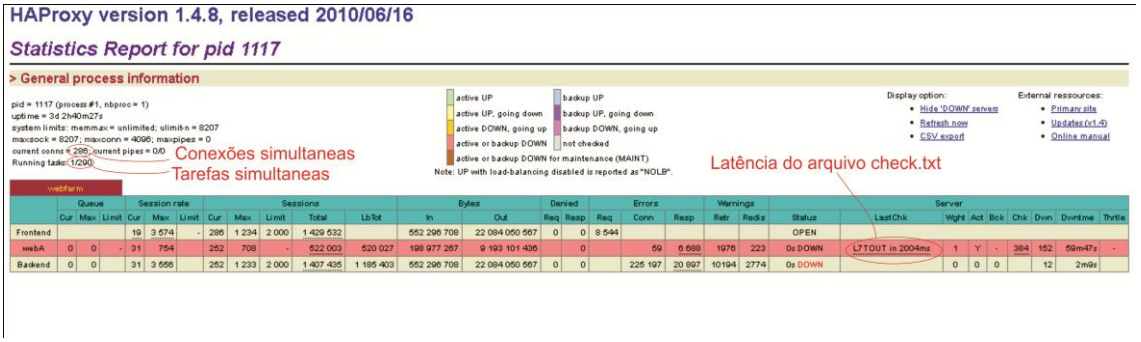
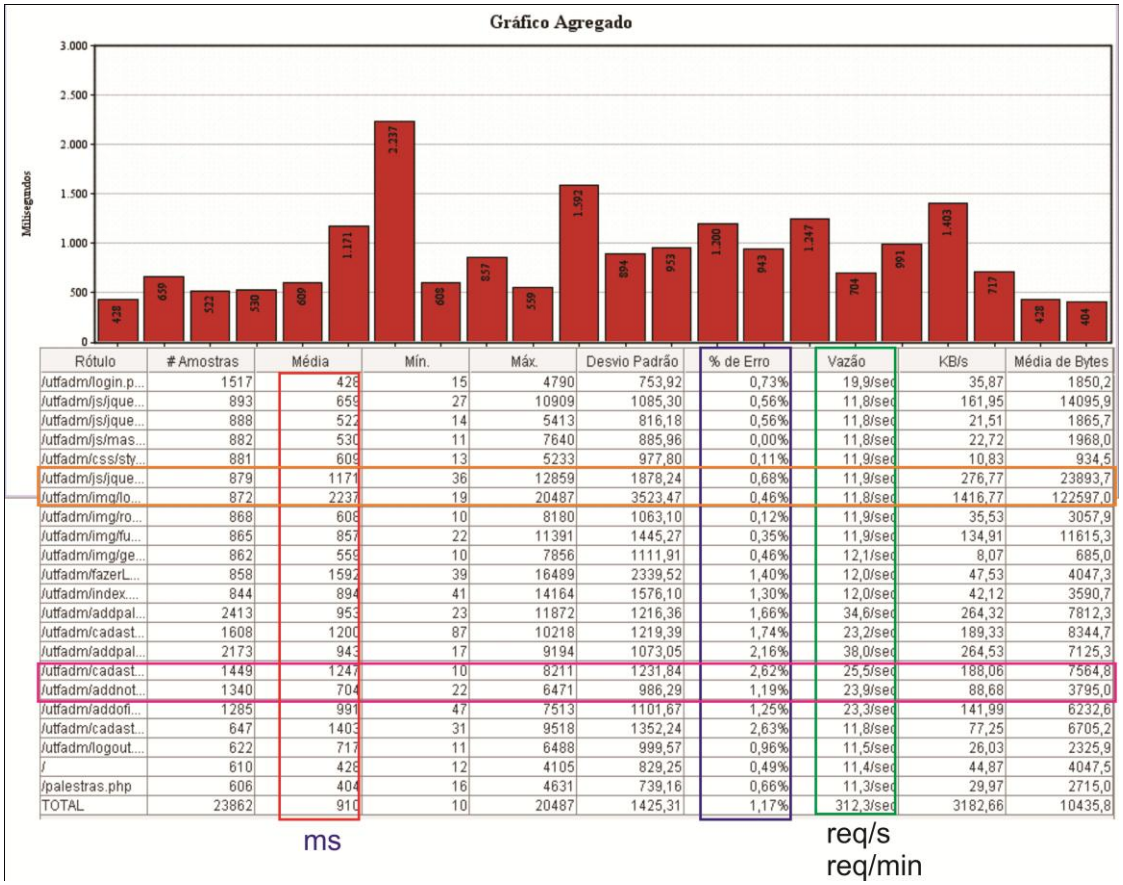


Figura 16 - Imagem da Ferramenta *HAProxy* que evidencia a queda do serviço.

Fonte: Autoria Própria.

A ferramenta de gerenciamento dos servidores web, disponibilizada pelo *HAproxy*, indica que para o teste de 300 usuários e apenas um servidor web, o número de requisições excede o limite suportado, ocasionando a queda do serviço.

No segundo experimento, foi utilizando o mesmo volume de dados utilizado no primeiro, mas com o diferencial de que ao invés de utilizar apenas um servidor web, foram conectados todos os quatro servidores disponíveis no cluster.



Rótulo	# Amostras	Média	Mín.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Média de Bytes
/utfadm/login.p...	1517	428	15	4790	753,92	0,73%	19,9/sec	35,87	1850,2
/utfadm/js/jque...	893	659	27	10909	1085,30	0,56%	11,8/sec	161,95	14095,9
/utfadm/js/jque...	888	522	14	5413	816,18	0,56%	11,8/sec	21,51	1865,7
/utfadm/js/mas...	882	530	11	7640	885,96	0,00%	11,8/sec	22,72	1968,0
/utfadm/css/sty...	881	609	13	5233	977,80	0,11%	11,9/sec	10,83	934,5
/utfadm/js/jque...	879	1171	36	12859	1878,24	0,68%	11,9/sec	276,77	23893,7
/utfadm/img/lo...	872	2237	19	20487	3523,47	0,46%	11,8/sec	1416,77	122597,0
/utfadm/img/ro...	868	608	10	8180	1063,10	0,12%	11,9/sec	35,53	3057,9
/utfadm/img/fu...	865	857	22	11391	1445,27	0,35%	11,9/sec	134,91	11615,3
/utfadm/img/ge...	862	559	10	7856	1111,91	0,46%	12,1/sec	8,07	685,0
/utfadm/fazerL...	858	1592	39	16489	2339,52	1,40%	12,0/sec	47,53	4047,3
/utfadm/index....	844	894	41	14164	1576,10	1,30%	12,0/sec	42,12	3590,7
/utfadm/addpal...	2413	953	23	11872	1216,36	1,66%	34,6/sec	264,32	7812,3
/utfadm/cadast...	1608	1200	87	10218	1219,39	1,74%	23,2/sec	189,33	8344,7
/utfadm/addpal...	2173	943	17	9194	1073,05	2,16%	38,0/sec	264,53	7125,3
/utfadm/cadast...	1449	1247	10	8211	1231,84	2,62%	25,5/sec	188,06	7564,8
/utfadm/addnot...	1340	704	22	6471	986,29	1,19%	23,9/sec	88,68	3795,0
/utfadm/addofi...	1285	991	47	7513	1101,67	1,25%	23,3/sec	141,99	6232,6
/utfadm/cadast...	647	1403	31	9518	1352,24	2,63%	11,8/sec	77,25	6705,2
/utfadm/logout....	622	717	11	6488	999,57	0,96%	11,5/sec	26,03	2325,9
/	610	428	12	4105	829,25	0,49%	11,4/sec	44,87	4047,5
/palestras.php	606	404	16	4631	739,16	0,66%	11,3/sec	29,97	2715,0
TOTAL	23862	910	10	20487	1425,31	1,17%	312,3/sec	3182,66	10435,8

ms

req/s
req/min

Figura 17 - Gráfico com representação do número de *requests* por segundo.

Fonte: Autoria Própria.

É possível visualizar no gráfico da Figura 17 primeiramente, que a média de resposta para as requisições decresceu de 9359ms para 910ms, ou seja, uma redução de aproximadamente 90% da carga do sistema.Observa-se também que as funções de *adicionarPalestra* e *adicionarPalestrante,* representadas no retângulo rosa , no primeiro teste foram as tarefas com maior tempo de resposta. No segundo teste a situação foi invertida, agora são as requisições com menor tempo de resposta. As requisições no retângulo laranja, */utfadm/js/jquery.min.js* e */utfadm/img/logo.png* são conteúdos estáticos e agora respondem com maior tempo ,1171ms e 2237ms respectivamente.

Por se tratar de conteúdo estático, em uma situação real, essas informações seriam carregadas uma única vez pelo navegador, permaneceriam em *cache e* só seriam recarregadas em caso de atualização no servidor. Testes

simulando *cache* de internet são possíveis de serem simulados, mas esta situação não foi aplicada no presente trabalho.

Outro item que apresentou melhorias em relação ao primeiro teste foi a vazão do sistema que passou de 16,9 req/s para 312.3req/s, aproximadamente 1800% de otimização.

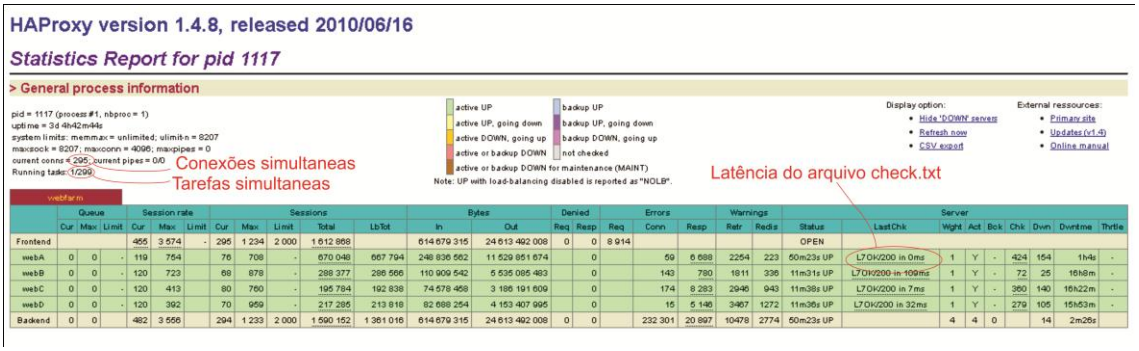


Figura 18 - representação do *Cluster* com 300 usuários.

Fonte: Autoria Própria.

A Figura 18 representa a situação do *Cluster*, com 300 usuários. Os tempos de latência entre os servidores web não atingem o valor de 2000ms, que é o tempo para que o servidor torne-se indisponível.

Diferenças de capacidade de processamento nos servidores web são visíveis ao analisar os tempos de latência do arquivo *check.txt.*

5. CONCLUSÃO

No ambiente de internet, sempre se está a procura por recursos computacionais que garantam a disponibilidade de serviços essenciais e que os executem com comportamento satisfatório. Para atender a essa demanda, asgrandes empresas normalmente investem em equipamentos isolados, “tops” de linha e de custo elevado e, muitas vezes de capacidade duvidosa.

Os *Clusters*de computadores podem suprir essa demanda, oferecendo soluções que a atendam por um preço mais acessível, muitas vezes menor que o de um supercomputador. Isso acontece pelo fato de quepodem ser aproveitados equipamentos já existentes nas organizações, dispensando novas aquisições.

Atualmente, grandes empresas, como o Google, por exemplo, já utilizam a tecnologia de cluster de computadores como infraestrutura para seus servidores, porém esta tecnologia pode atenderorganizações de qualquer porte, ou ainda, aquelas organizações que não dispõem de capital para o ambiente de TI.

Este trabalho teve como objetivo construir uma estrutura de alta disponibilidade e com balanceamento de carga. Aarquitetura já era existente, mas configurada de forma simples, mas ao mesmo tempo escalável para se desenvolver aplicações *Web* distribuídas. A facilidade para se adicionar novas máquinas no ambiente, ou substituir máquinas defeituosas se dá pela utilizaçãode novas distribuições geradas do sistema. Acredita-se que estes objetivos tenham sidoalcançados.

Testes foram realizados para comparar o desempenho da arquitetura,quando utilizado o balanceamento de carga entre os quatro servidorese comparado com apenas um servidor, notou-se uma melhora significativa de desempenho,comprovando assim as vantagens em distribuir a carga do sistema entre os servidores.

Foram encontradas diversas dificuldades na execução de testes, tais como: dificuldade de configurar a ferramenta *ApacheJMeter* para criar gráficos relevantes. O *JMeter* foi decisivo parasubmeter os servidores a uma carga significativa, pois com apenas um cliente gerando as *threads* simultâneas,observou-se que a capacidade de processamentopara simular um

ambiente com múltiplos usuários esgotava-se antes doservidor ficar sobrecarregado.

A ferramenta *JMeter* não permite que seja feita uma configuração de númerode *threads*por máquina de execução, mas obriga a utilizar-se sempre omesmo número de *threads* por máquina.

Limites de configurações nos servidores: alguns testes foram executadoscom grande quantidade de erros ou com tempos de execução suspeitos.Identificou-se que isso ocorria por estarem sendo atingidos limites deconfiguração para o número máximo de processos nos servidores Web eno servidor de banco de dados *MySql.*

A simulação criada gerava erro no banco de dados se esta fosse executada por mais de 5 minutos em um servidor web individual, pois o número de operações geradas não era suportada pelas configurações locais do serviço *MySql*.

Conclui-se que, se as falhas são inevitáveis, as suas conseqüências para a disponibilidade podem ser minimizadas, senão eliminadas. São necessários meios de prever e minimizar as paradas de sistema com ambientes de cluster de alta disponibilidade, e se necessário efetuar uma parada, prover meios para minimizar o tempo de substituição de servidores, utilizando imagens de sistema.

6. REFERÊNCIAS

AGARWAL, Anant; HOFFMANN, Henry; EASTEP, Jonathan; SANTANBROGIO, Marco D. ; MILLER, Jason E. A Generic Interface for Specifying Program Performance and Goals in Autonomous Computing Environments.**Application Heartbeats.**Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory**,** ICAC'10, Washington, USA.Jun. 7-11, 2010, pag.79-88.

ALVIM, Rômulo Miranda; GROSSMANN, Flávio Lúcio Leite; DANTAS, Marco Antônio Ribeiro. "Implementação de uma Arquitetura para Serviço Distribuído com Grande Disponibilidade em Ambiente Linux", **Revista Eletrônica de Iniciação Científica** – Sociedade Brasileira de Computação, Vol. II, No. III, 2002.

BOOKMAN, Charles. **Agrupamento de Computadores em Linux**. Rio de Janeiro: Ciência Moderna, 2003.

CAMPOS, Augusto. **O que é uma distribuição Linux**. BR-Linux. Florianópolis, mar. 2006. Disponível em: <http://br-linux.org/linux/faq-distribuicao>. Acessado em: 13 mar. 2013.

CONVERSE, Tim; PARK, Joyce; MORGAN, Clark.**PHP5 and MySQL bible**. Wiley, 2004

COSTA, Hebert Luiz Amaral. **Alta Disponibilidade e Balanceamento de Carga para Melhoria de Sistemas Computacionais Críticos usando Software Livre: Um Estudo de Caso**. 2009. 126 f. Dissertação (Mestrado em Ciência da Computação) – Faculdade de Engenharia Mecânica, Universidade Federal de Viçosa, Viçosa, Minas Gerais, 2009.

DELISLE, Marc. **Mastering phpMyAdmin 3.1 for effective MySQL management**.Packt Pub Limited, 2009.

FERREIRA, Edmar. **Escolhendo entre escalabilidade horizontal eescalabilidade vertical**. 2010. Disponível em: < http://escalabilidade.com/2010/09/21/escolhendo-entre-escalabilidade-horizontal-e-escalabilidade-vertical/>. Acesso em: 09 mar. 2013.

GOMES,ChristianLyra.**GuiadoServidorConectivaLinux7.0**.ConectivaS.A .cap.11. Disponível em <http://www.dimap.ufrn.br/~aguiar/Manuais/Servidor/index.html>.2001.Acessoem02nov.2012.

HENDERSON, Cal. **Building Scalable Web Sites**.2006. O'Reilly Media, Inc, 2006,1ª Edição. ISBN-10: 0-596-10235-6. ISBN-13: 978-0-596-10235-7.

JAKARTA, Apache. **Apache JMeter**. 2011. Disponível em: <http://jakarta.apache.org/jmeter/>. Acessoem: 02 Mar. 2013.

KAUSHAL, Vishonika; BALA, Anju.**Autonomic Fault Tolerance using HAProxy in Cloud Environment**, International Journal Of Advanced Engineering Sciences And Technologies, Patiala – Índia, 2011. Pag. 55.

LACERDA, A; QUINTÃO, F; SABINO, V. **Arquitetura do Google Cluster. Programa de Residência em Tecnologia**. UFMG. LINUX VIRTUAL SERVER. 2007.

LOPES FILHO, E. D. M. O. **Arquitetura de alta disponibilidade para firewall e IPS baseada em SCTP**. Diss. Universidade Federal de Uberlândia, 2008.

MAJÓ, Aniol Oliver i. **OpenERPFiscalfàcil**,UniversitatOberta de Catalunya, 2012.

NATÁRIO, Rui. **Balanceamento de Carga.** 2011. Disponível em:http://redes-e-servidores.blogspot.com.br/2011/03/balanceamento-de-carga-i.html/>. Acessoem 09 mar. 2013.

NEVEDROV, Dmitri. **Using JMeter to Performance Test Web Services**.Dev2dev**.** Disponivel em: <http://dev2dev.bea.com/pub/a/2006/08/jmeter-performance-testing.html>. Acesso em: 04 mar. 2013.

PATTERSON, David A. & HENNESSY, John L**. Organização e Projeto de Computadores**: A interface Hardware/Software. 2 ed. Rio de Janeiro, LTC – Livros Técnicos e Científicos Editora S.A., 2000.

PITANGA, Marcos. **Computação em cluster**: o estado da arte da computação. Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2003. 322p.

PITANGA, Marcos. **Construindo Supercomputador com Linux**. 2. Ed. Rio de Janeiro: Brasport, 2004.

REIS, Christian Robottom. **Caracterização de um Processo de Software para Projetos de Software Livre.** Fev. 2003. 247p. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) Instituto de Ciências Matemáticas e de Computação. Universidade São Paulo. São Carlos – SP, 2003.

RÔLLA, Felipe Martins. **Desenvolvendo Firewalls Seguros Em Ambientes De Alta Disponibilidade Utilizando Software Livre**. Universidade Federal do Rio de Janeiro. Núcleo De Computação Eletrônica Pós-Graduação Em Gerência De Segurança Da Informação (MSI). 2009. 31p.

RUSCHEL, André Guedes. **Terminais sem Hard Disk**: Aprenda a configurar um servidor de boot remoto no Windows Server. Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2009.

TEIXEIRA, Mário Antonio Meireles. **Suporte a serviços diferenciados em servidores web:** modelos e algoritmos. Diss. PhD thesis, CMC-USP, São Carlos-SP, 2004.

Disponível em http://www.deinf.ufma.br/~mario/producao/tese_swds.pdf

TORRES, Gabriel. **Redes de computadores**: curso completo. Rio De Janeiro: Axcel Books, 2001.