



**CURSO SUPERIOR DE TECNOLOGIA  
EM REDES DE COMPUTADORES**

**DIEGO DE SÁ  
VINICIUS HIURI NEVES**

**CLUSTER DE ALTA DISPONIBILIDADE EM LINUX**

**Santa Bárbara d'Oeste  
2012**

**FACULDADE POLITEC**  
**CURSO SUPERIOR DE TECNOLOGIA**  
**EM REDES DE COMPUTADORES**

**DIEGO DE SÁ**  
**VINICIUS HIURI NEVES**

**CLUSTER DE ALTA DISPONIBILIDADE EM LINUX**

Trabalho apresentado à Faculdade Politec,  
como exigência para obtenção do grau de  
Tecnólogo em Redes de Computadores, sob a  
orientação do prof. Mario Luiz Bernardinelli.

**Santa Bárbara d'Oeste**  
**2012**

**Aprovação**

**DIEGO DE SÁ**

**VINICIUS HIURI NEVES**

**CLUSTER DE ALTA DISPONIBILIDADE EM LINUX**

Trabalho aprovado em: \_\_\_\_/\_\_\_\_/\_\_\_\_

Nota: \_\_\_\_ (\_\_\_\_)

---

Nome do primeiro examinador

---

Nome do segundo examinador

---

Mario Luiz Bernardinelli

**Santa Bárbara d'Oeste**

**2012**

## DEDICATÓRIA

Dedico este trabalho aos meus pais, irmãos, minha futura esposa Priscila Mazer Tagliapietra, Deus e amigos. (Diego de Sá)

Dedico este trabalho a minha família, que sempre que eu pensava em desistir me deram forças para continuar, acreditando em mim e me orientando para seguir o caminho correto, dedico também ao meu companheiro de trabalho de conclusão, pela dedicação e o esforço prestado. (Vinicius Hiuri Neves)

## **AGRADECIMENTOS**

Gostaria de agradecer a minha mãe que sempre acreditou no meu potencial e ajudou para que esse sonho se tornasse realidade, a minha futura esposa pela ajuda, compreensão e paciência e a meu orientador Mario Luiz Bernardinelli pela motivação e orientação, pessoas que contribuíram na conclusão de mais uma jornada em minha vida, pois sem o apoio deles não teria chegado até aqui. (Diego de Sá)

Agradeço a Deus, pela oportunidade que tive de me preparar para o ensino, pela sabedoria obtida através da Faculdade Politec, aos professores, coordenadores, amigos que tanto me ajudaram e apoiaram e minha família que me cativou e apoiou. (Vinicius Hiuri Neves)

*Software* é como sexo: é melhor quando é de graça.  
(Linus Torvalds)

## RESUMO

*Cluster* é um conjunto de computadores interconectados entre si que trabalham de maneira conjunta para realizar grandes processamentos. Tem como finalidade melhorar o desempenho de aplicações e criar redundância, formando um ambiente de alta disponibilidade. Disponibilidade é a probabilidade de se encontrar recursos e/ou aplicações em perfeito estado e condições de uso; Alta Disponibilidade é a capacidade de um sistema de ficar ininterrupto. Esses tipos de *clusters* são usados em aplicações de missão crítica, pois são eficientes na proteção e detecção de falhas. Este trabalho tem por objetivo demonstrar a construção de um cluster de alta disponibilidade em Linux utilizando-se de ferramentas de código aberto, dentre as quais se destacam o *Distributed Replicated Block Device (DRBD)*, *Heartbeat* e *Mon*. Será demonstrado um estudo de caso no qual serão realizadas a instalação e configuração destas ferramentas em sistemas Debian. A solução será baseada em: replicação de discos, monitoramento dos servidores e alta disponibilidade em um servidor de arquivos Samba, bem como os resultados obtidos.

**Palavras-chave:** Cluster, Alta Disponibilidade, DRBD, Heartbeat e Mon.

## ABSTRACT

Cluster is an ensemble of computers with interconnection between them which work altogether in order to make huge processing. Its aim is to improve the performance of applications and to create redundancy, forming a high availability environment. Availability is the probability of finding resources and/or applications in perfect state and use conditions; High Availability is the capacity of a system of staying uninterrupted. These kinds of clusters are used in applications of critical goal because they are efficient in protection and detection of failures. The aim of this job is to demonstrate the construction of a cluster of high availability in Linux using open source tools, in which are highlighted the Distributed Replicated Block Device (DRBD), Heartbeat and Mon. It will be shown a case study in which will be performed the installation and configuration of these tools on Debian systems. The solution will be based in: replication of disks, monitoring of the servers and high availability in a Samba-based server, as well as the obtained results.

**Keywords:** Cluster, High Availability, DRBD, Heartbeat and Mon.



## Lista de Abreviaturas e Siglas

DRBD	Distributed Replicated Block Device
DHCP	Dynamic Host Configuration Protocol
FTP	File Transfer Protocol
GNU	Gnu Is Not Unix
HA	High Availability
HASP	Houston Automated Spooling Program
HD	Hard Disk
IBM	International Business Machines
IP	Internet Protocol
JES	Job Entry System
LBC	Load Balancing Cluster
LVS	Linux Virtual Server
MS-DOS	Microsoft Disk Operating System
MTBF	Mean Time Between Failure
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
NIC	Network Interface Card
PDC	Primary Domain Controller
RAID	Redundant Array of Inexpensive Disks
SMB	Server Message Block
SSH	Secure shell
TCP/IP	Transmission Control Protocol/Internet Protocol

TI Tecnologia da Informação

XP eXPerience

## LISTA DE FIGURAS

FIGURA 1: CLUSTER ALTA DISPONIBILIDADE (HIGH AVAILABILITY (HA)).....	18
FIGURA 2: BALANCEAMENTO DE CARGA ( <i>LOAD BALANCING</i> ).....	20
FIGURA 3: COMBINAÇÃO DE ALTA DISPONIBILIDADE E BALANCEAMENTO DE CARGA .....	21
FIGURA 4: CUSTOS DA DISPONIBILIDADE. ....	24
FIGURA 5: ALTERNÂNCIA ENTRE OS PERÍODOS DE FUNCIONAMENTO E REPARO. ....	27
FIGURA 6: FUNCIONAMENTO DO <i>HEARTBEAT</i> . ....	31
FIGURA 7: DIFERENÇA ENTRE O DRBD E RAID-1 .....	33
FIGURA 8: FUNCIONAMENTO DO DRBD. ....	35
FIGURA 9: ARQUITETURA DO ESTUDO DE CASO.....	38
FIGURA 10: ARQUIVO DE CONFIGURAÇÃO <i>AUTHKEYS</i> DO <i>HEARTBEAT</i> . ....	42
FIGURA 11: ARQUIVO DE CONFIGURAÇÃO <i>HARESOURCES</i> DO <i>HEARTBEAT</i> . ....	43
FIGURA 12: IP VIRTUAL EM FUNCIONAMENTO NO SERVIDOR PRIMÁRIO.....	44
FIGURA 13: IP VIRTUAL EM FUNCIONAMENTO NO SERVIDOR SECUNDÁRIO. ....	45
FIGURA 14: DISPOSITIVOS DO DRBD.....	46
FIGURA 15: RETIRANDO A MONTAGEM AUTOMÁTICA DO DISPOSITIVO NO ARQUIVO <i>FSTAB</i> . ....	49
FIGURA 16: CRIAÇÃO DO DISCO VIRTUAL.....	50
FIGURA 17: INÍCIO DA SINCRONIZAÇÃO ENTRE OS DISPOSITIVOS VIRTUAIS. ....	51
FIGURA 18: SINCRONIZAÇÃO EM 56,1% .....	51
FIGURA 19: SINCRONIZAÇÃO TERMINADA. ....	51
FIGURA 20: FORMATAÇÃO DO DISPOSITIVO DO DRBD TERMINADA. ....	52
FIGURA 21: DIRETÓRIO CRIADO NO SERVIDOR PRIMÁRIO.....	53
FIGURA 22: PASTA REPLICADA NO SERVIDOR SECUNDÁRIO.....	54
FIGURA 23: TRECHO DO ARQUIVO DE CONFIGURAÇÃO DO <i>MON</i> , ONDE INDICA O EMAIL ONDE SERÁ ENVIADO ALERTAS .....	57
FIGURA 24: <i>HEARTBEAT.ALERT</i> .....	58
FIGURA 25: ARQUIVO <i>SUDOERS</i> .....	59
FIGURA 26: SAÍDA DO COMANDO <i>MONSHOW</i> . ....	59
FIGURA 27: SAÍDA DO COMANDO <i>MONSHOW</i> EM CASO DE FALHA DO SERVIÇO. ....	60
FIGURA 28: ARQUIVO DE CONFIGURAÇÃO DO SAMBA. ....	61
FIGURA 29: TESTANDO ARQUIVO DE CONFIGURAÇÃO DO SAMBA COM O COMANDO <i>TESTPARM</i> . ....	62
FIGURA 30: TELA DE <i>LOGIN</i> . ....	63
FIGURA 31: MAPEAMENTO DO COMPARTILHAMENTO NO CLIENTE XP.....	64
FIGURA 32: ARQUIVO DE TEXTO CRIADO NO CLIENTE XP, ENQUANTO O NODE1 RESPONDIA PELO <i>CLUSTER</i> .....	65
FIGURA 33: ACESSO SSH NO IP DO CLUSTER, ENQUANTO O NODE1 RESPONDIA PELO MESMO. ....	65
FIGURA 34: MOMENTO DA PARADA DO HEARTBEAT NO SERVIDOR PRIMÁRIO (NODE1). ....	66
FIGURA 35: MOMENTO QUE O SERVIDOR SECUNDÁRIO (NODE2), COMEÇA A RESPONDER COMO PRIMÁRIO. ....	67
FIGURA 36: ACESSO SSH NO IP DO CLUSTER, ENQUANTO O NODE2 RESPONDIA PELO MESMO. ....	67
FIGURA 37: ARQUIVO DE TEXTO REPLICADO E NOVO ARQUIVO CRIADO NO NODE2. ....	68

## LISTA DE TABELAS

TABELA 1: ATRIBUTOS DA DEPENDABILIDADE .....	26
TABELA 2: MEDIDAS DE DISPONIBILIDADE .....	27
TABELA 3: MEDIDAS DE CONFIABILIDADE. ....	28
TABELA 4: HARDWARE DOS SERVIDORES. ....	40
TABELA 5: DESCRIÇÃO DAS MÁQUINAS UTILIZADAS. ....	41

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>13</b>
<b>2 DEFINIÇÃO DE CLUSTER.....</b>	<b>14</b>
2.1 Aplicações para clusters.....	15
2.2 Vantagens .....	16
<b>3 TIPOS DE CLUSTER .....</b>	<b>17</b>
3.1 Cluster de Alta Disponibilidade (High Availability - HA).....	17
3.2 Cluster de balanceamento de carga LBC (Load Balancing Cluster) .....	19
3.3 Combinação alta disponibilidade com balanceamento de carga .....	21
3.4 Clusters de processamento distribuído ou processamento paralelo .....	22
<b>4 ALTA DISPONIBILIDADE.....</b>	<b>23</b>
4.1 Como medir a disponibilidade .....	24
4.2 Dependabilidade .....	25
4.2.1 Disponibilidade.....	26
4.2.2 Confiabilidade.....	27
4.3 Níveis de alta disponibilidade .....	28
4.3.1 Disponibilidade Básica .....	28
4.3.2 Alta disponibilidade .....	29
4.3.3 Disponibilidade Continuada .....	29
<b>5 FERRAMENTAS UTILIZADAS.....</b>	<b>30</b>
5.1 Heartbeat .....	30
5.2 Distributed Replicated Block Device (DRBD).....	32
5.2.1 Ferramentas de administração.....	35
5.2.2 Sincronismo .....	36
5.3 Mon.....	37
<b>6 ESTUDO DE CASO: CONFIGURANDO UM SERVIDOR SAMBA COM ALTA DISPONIBILIDADE .....</b>	<b>38</b>
6.1 Samba .....	39
6.2 Hardware Utilizado .....	40
6.3 Configurando o Heartbeat .....	41
6.4 Configurando o DRBD .....	45
6.5 Configurando o Mon .....	54
6.6 Configurando o Samba.....	60
<b>7 TESTANDO A FUNCIONALIDADE DO CLUSTER DE ALTA DISPONIBILIDADE .....</b>	<b>64</b>
<b>8 CONSIDERAÇÕES.....</b>	<b>70</b>
<b>ANEXOS.....</b>	<b>74</b>
A. Vídeo de demonstração do funcionamento da alta disponibilidade.....	74

## 1 INTRODUÇÃO

Atualmente para uma empresa que pretenda manter-se competitiva no mercado, tornou-se de vital importância possuir uma estratégia envolvendo alta disponibilidade em seus recursos e/ou aplicações, e eventualmente estar preparada para possíveis falhas de *hardware* ou *software*. “Quanto mais os negócios demandam que um serviço ou uma aplicação estejam mais disponíveis, maior será a necessidade de se considerar um ambiente altamente disponível.” (CHRISTIANINI, 2011).

A principal finalidade de um sistema de alta disponibilidade é a capacidade de combater falhas de forma automática, a fim de garantir que os serviços prestados fiquem o maior tempo possível disponíveis ao usuário. É possível alcançar um ambiente altamente disponível através da configuração de dois ou mais servidores que, trabalhando em conjunto e com monitoração entre si; com o conceito de primário e secundário; em caso de falhas um seja capaz de assumir os serviços que ficaram indisponíveis, formando assim um *cluster* de alta disponibilidade.

Um *cluster* é composto por questões complexas de pesquisa em computação tais como escalabilidade, disponibilidade, tolerância a falhas, desempenho e a relação custo benefício. Neste trabalho será apresentada uma solução com custos relativamente baixos se comparados aos seus benefícios, com uso de ferramentas de código aberto utilizando o DRDB, *Heartbeat*, Mon e Samba.

O DRBD terá a função de espelhar os dados entre um servidor e outro, usando a rede como meio de transmissão. O *Heartbeat* é responsável pela verificação dos servidores e tomada de decisões. O Mon fica responsável pela monitoração dos serviços, para em caso de falhas, enviar alertas tanto para o administrador da rede quanto para o *Heartbeat*. O Samba fará o compartilhamento dos arquivos entre os clientes da rede. Ele não faz parte da implementação do *cluster* em si, mas será utilizado para comprovar na prática o seu funcionamento.

## 2 DEFINIÇÃO DE CLUSTER

*Cluster* é um conjunto de computadores ou sistemas interconectados entre si que, trabalhando em conjunto, têm como finalidade melhorar o desempenho de aplicações e criar redundâncias para que, em caso de falhas de *hardware* ou *software*, os recursos e aplicativos de missão crítica continuem disponíveis para o usuário final.

Os computadores presentes no *cluster* permanecem em constante sincronismo através de troca de mensagens periódicas, sendo um com o papel de mestre (*master*) e os demais como escravos, ou nós. Caso algum nó falhe ou entre em manutenção, outro nó deve assumir imediatamente o papel da máquina principal, mantendo assim o serviço em operação.

*Cluster* é um sistema que compreende dois ou mais computadores ou sistemas que trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma que os usuários do agrupamento de máquinas tenham a impressão que somente um único sistema responde para eles, criando assim uma ilusão de um recurso único (computador virtual). Esse conceito é denominado transparência do sistema. Como características fundamentais para a construção das plataformas incluem-se elevação da confiança, distribuição de carga e performance. (PITANGA, 2003).

De acordo com (KOPPER, 2005), um *cluster* compõem-se de quatro características básicas:

- Transparência para os usuários;
- Os computadores do *cluster* não têm ideia que fazem parte de um *cluster*;
- As aplicações ou recursos em execução, não têm ideia que fazem parte de um *cluster*;
- Os computadores que fazem parte do *cluster* têm que ser vistos como clientes comuns.

Uma rede de computadores não é caracterizada como um sistema *cluster*, pois nela os computadores operam de forma autônoma, fugindo assim do conceito de imagem única.

*Cluster* nasceu na década de 60 desenvolvido primeiramente pela IBM (*International Business Machines*) com a ideia de se fazer conexões entre *mainframes*, com o intuito de entrar no mercado de paralelismo. Até então o sistema HASP (*Houston Automated Spooling Program*) desenvolvido pela própria IBM e seu sucessor o JES (*Job Entry System*) tinham o trabalho de fornecer esta interconexão de tarefas nos *mainframes*.

Com o passar do tempo o sistema *cluster* foi ganhando força até que, na década de 80, surgiram quatro técnicas convergentes:

- Necessidade de microprocessadores de alto desempenho;
- Redes de alta velocidade;
- Padronização das ferramentas para computação distribuída de alto desempenho;
- Maior processamento para aplicações e recursos científicos.

Foi então que em 1993 que Donald Becker e Thomas Sterling deram início a um projeto com o intuito de construir um sistema de processamento distribuído com *hardware* comum, combatendo assim o alto custo dos supercomputadores. Já em 1994 nasceu o primeiro projeto deste tipo, chamado *beowulf*, um sistema *cluster* composto por 16 processadores que funcionavam ligados através de dois canais *ethernet*.

Com o sucesso do projeto, não só o meio acadêmico, mas também as comunidades de pesquisa e a NASA adotaram a ideia.

## 2.1 Aplicações para clusters

Uma aplicação ou recurso para ser implantado num sistema *cluster* de alta disponibilidade primeiramente precisa satisfazer alguns pré-requisitos (PFISTER, 2006):

- Para iniciar, interromper ou parar a aplicação deve conter suporte a linha de comando e ou *script* capaz de controlar sua inicialização;



- Suporte a múltiplas instâncias da mesma aplicação;
- A aplicação deve ter suporte a armazenamento compartilhado;
- Possuir a capacidade de salvar o estado do seu armazenamento compartilhado e restaurar em outro nó o estado anterior a ocorrência da falha;
- Não corromper os dados na ocorrência da falha.

## 2.2 Vantagens

Com o aumento constante da demanda de recursos, se faz necessária uma grande capacidade computacional, requisito este que os sistemas de *clusters* preenchem perfeitamente, seja para melhorar o desempenho, seja para garantir a disponibilidade de aplicações de missão crítica.

Segundo (PITANGA, 2008), ainda podemos destacar mais algumas vantagens:

- **Escalabilidade:** Possibilidade de que novos componentes sejam adicionados à medida que cresce a carga de trabalho;
- **Tolerância a falhas:** Aumento de confiabilidade do sistema, caso alguma parte falhe;
- **Baixo custo:** Obtenção de processamento de alto desempenho com um baixo custo;
- **Independência de fornecedores:** Utilização de hardware aberto, software livre e independência de fabricantes e licenças de uso.

### 3 TIPOS DE CLUSTER

Existem alguns tipos de *clusters* dos quais se destacam: *cluster* de alta disponibilidade, *cluster* de balanceamento de carga e combinação de alta disponibilidade com balanceamento de carga.

#### 3.1 Cluster de Alta Disponibilidade (High Availability - HA)

O *Cluster* de alta disponibilidade tem como finalidade deixar um sistema no ar trezentos e sessenta e cinco dias por ano, evitando assim quedas de serviços que podem causar até mesmo a parada de uma empresa inteira, provocando prejuízos incalculáveis para a mesma.

De acordo com (PITANGA, 2008) nos *clusters* de alta disponibilidade o que garante um serviço ininterrupto é o trabalho das máquinas configuradas para realizar em conjunto a replicação de aplicações e servidores. A ideia central é que, se um nó do *cluster* vier a falhar (*failover*), outro já esteja pronto para assumir imediatamente o papel principal, de forma tão transparente que chega a ser imperceptível para o usuário.

Servidores de missão crítica não podem ficar fora de serviço em momento algum, pois falhas podem provocar prejuízos incalculáveis para a empresa.

Em 1995, dois estudos patrocinados pela Oracle Corporation e a Datamation mostraram que empresas diversas por todo o Estados Unidos perderam, em média, US\$ 80 mil a US\$ 350 mil por hora de paralisação não planejada. Vale lembrar que prevenir é melhor do que remediar, mesmo que nem sempre saia mais barato, mas há a vantagem de se saber quanto se está gastando. (Ruiz, 2000).

Imagine *sites* de comércio eletrônico e instituições financeiras fora do ar? Seria como doar clientes para a concorrência. Através da alta disponibilidade é possível atingir 99,9% de

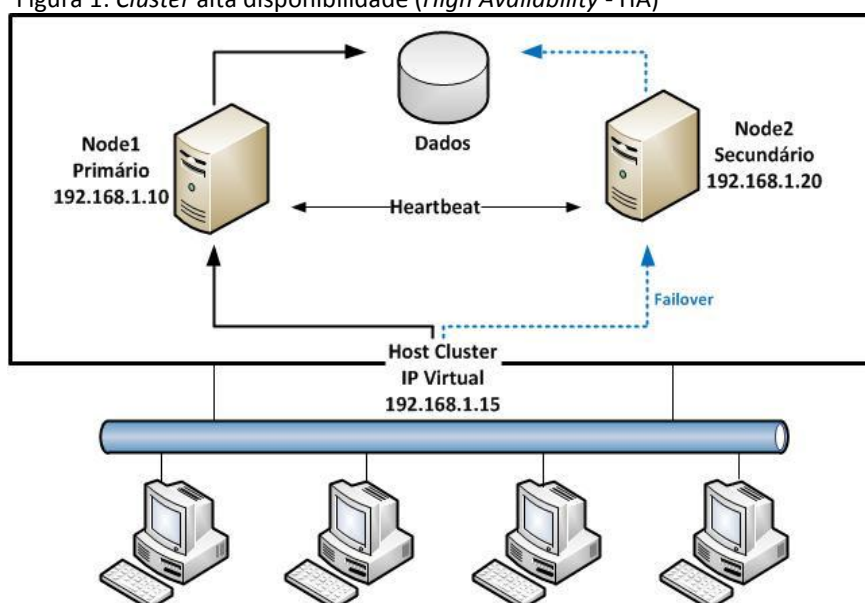
disponibilidade (mais conhecido como “*three nines*”) significando que o servidor não fica fora do ar por mais que 43 minutos por mês.

O risco de se manter apenas um computador realizando uma tarefa importante e falhar é muito alto, pois problemas de *hardware* e/ou aplicações podem causar a interrupção do serviço.

A Figura 1 demonstra o funcionamento de um *cluster* de alta disponibilidade entre dois servidores de arquivos com o monitoramento do *Heartbeat* e espelhamento de dados. Cada servidor tem seu endereço IP (*Internet Protocol*) fixo, o conjunto como um todo forma um servidor virtual que atende com o IP 192.168.1.15. O servidor chamado node1 é o servidor primário, funcionando assim como servidor que está em produção.

A implementação da disponibilidade é realizada pelo *software Heartbeat* e funciona assim: cada servidor possui um endereço IP específico e um endereço IP virtual é utilizado pelo servidor principal. O *software Heartbeat* de cada um dos servidores monitora o servidor ativo e, caso ocorra falha, o servidor secundário assume o IP virtual. Quando o servidor principal voltar a operar, o *software Heartbeat* volta o controle para o servidor principal, que assume o IP virtual.

Figura 1: Cluster alta disponibilidade (*High Availability - HA*)



Fonte: Autoria própria.

Exemplos de alta disponibilidade usando código aberto (*Open-source*) em ambiente GNU/Linux:

- O *Linux Virtual Server (LVS)*: É uma solução *open source* avançada de balanceamento de carga para sistemas *Linux*, desenvolvida por Wensong Zhang em maio de 1998.
- *Ediware*: Faz roteamento de tráfego, atuando com escalabilidade de servidores *web*, operando em duas camadas: *frontend* e *backend*.
- *TurboLinux Cluster*: Provê alta disponibilidade para roteadores e servidores.
- *Heartbeat*, *Mon*, *DRBD*: Provêm alta disponibilidade e serão abordados mais adiante.

### 3.2 Cluster de balanceamento de carga LBC (Load Balancing Cluster)

De acordo com (Morimoto, 2007) os *clusters* de balanceamento de carga, *Load Balance Clusters* (LBC), funcionam redirecionando o tráfego gerado pelos clientes para vários servidores, com a finalidade de combater os gargalos criados por vários acessos simultâneos a um mesmo servidor.

O servidor que faz o balanceamento da carga também é chamado de servidor virtual que, configurado para operar em conjunto com os servidores, tem a finalidade de melhorar o tempo de acesso aos dados e aumentar a disponibilidade das aplicações. Este computador é colocado no “*front-end*” (na fente), recebe as conexões e as repassa de uma maneira balanceada para os nós que executam os mesmos aplicativos, compondo assim o *cluster*. A alta disponibilidade é alcançada através do monitoramento dos nós que compartilham a responsabilidade de controlar os pedidos, e se algum nó falhar, ele é excluído da relação de máquinas receptoras. Isto pode provocar uma queda no desempenho, mas garante a disponibilidade do serviço.

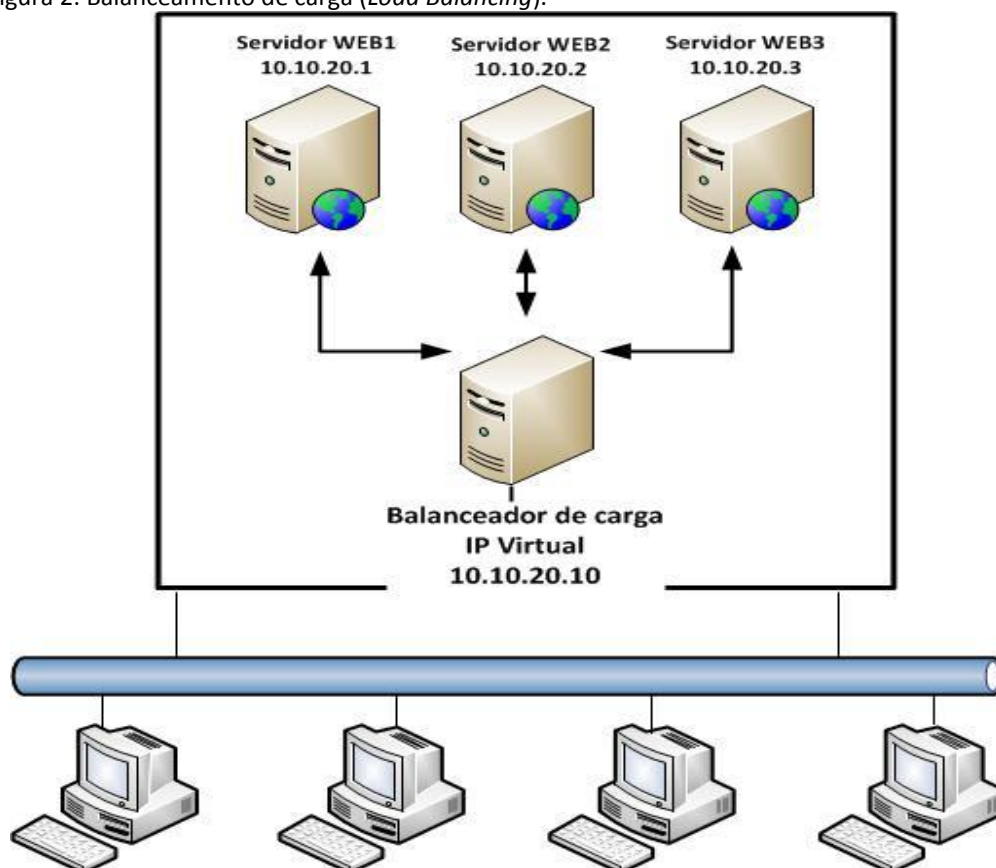
A vantagem deste sistema é que a qualquer momento se pode adicionar novos

computadores ao *cluster*, evitando assim a necessidade de *upgrade* dos já existentes. Os pedidos de requisição podem ser divididos de acordo com a capacidade de processamento dos servidores equilibrando assim a diferença entre eles.

Como exemplos típicos de uso de *clusters* de balanceamento de carga estão as empresas de *e-commerce*, que possuem servidores com altos índices de acesso.

A Figura 2 demonstra um conjunto de servidores que formam um *cluster* LBC, e o balanceador de carga distribui as requisições dos usuários, o conjunto como um todo é visto como se fosse apenas um servidor.

Figura 2: Balanceamento de carga (*Load Balancing*).



Fonte: Autoria própria.

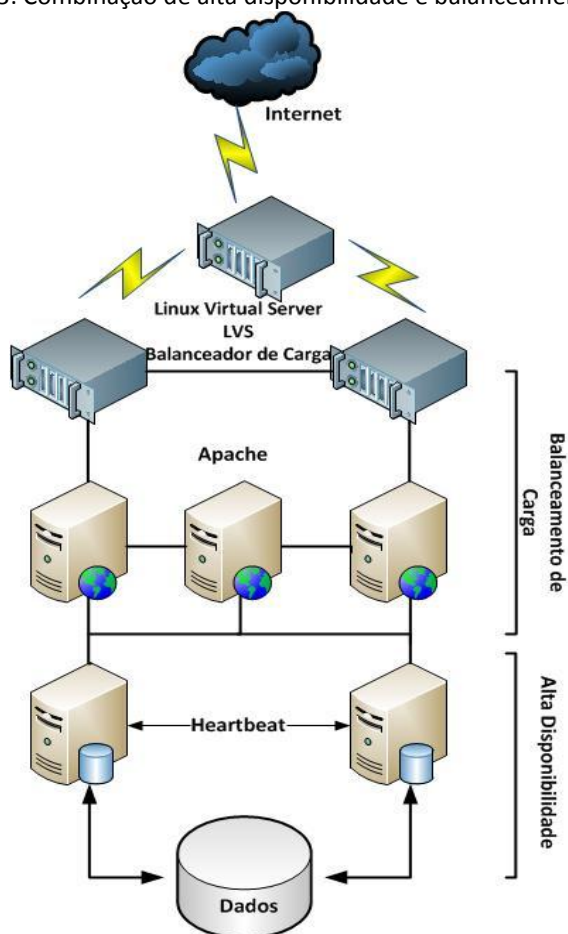
### 3.3 Combinação alta disponibilidade com balanceamento de carga

Como o próprio nome diz, *cluster* de alta disponibilidade e balanceamento de carga é a combinação entre a alta disponibilidade e balanceamento de carga. Consiste num conjunto de *clusters* trabalhando em sincronia aumentando assim não somente a disponibilidade, mas oferecendo a distribuição de carga de recursos e aplicações (PITANGA, 2008).

Este tipo de *cluster* é muito utilizado em serviços *web*, *mail*, *news* e *FTP*.

A Figura 3 demonstra o funcionamento do *cluster* que combina a alta disponibilidade e balanceamento de carga.

Figura 3: Combinação de alta disponibilidade e balanceamento de carga



Fonte: Autoria própria.

### 3.4 Clusters de processamento distribuído ou processamento paralelo

Este tipo de *cluster* tem como finalidade aumentar o desempenho e a disponibilidade de recursos e aplicações que exigem alto poder de processamento. Segundo (TANEMBAUM, 2007) o paralelismo é: “o uso de várias unidades de processamento para executar uma tarefa de forma mais rápida”.

É uma tecnologia que agrupa computadores formando um supercomputador virtual, onde um processo de grande consumo é dividido em pequenas tarefas e distribuído através dos nós que fazem parte do *cluster*. Funciona como uma linha de produção, onde cada funcionário desempenha um papel específico, de forma simultânea, porém todos com a mesma finalidade que é obter o produto final da melhor forma e o mais rápido possível.

Estes *clusters* são utilizados para computação científica, análises financeiras, prospecção de petróleo, cálculos aeroespaciais etc, que normalmente são processos que exigem enorme capacidade de processamento.

## 4 ALTA DISPONIBILIDADE

Alta disponibilidade é a capacidade de um sistema ou serviço de tecnologia da informação (TI) ficar ininterrupto para o usuário por um período de tempo superior ao que seus componentes seriam capazes de suportar sozinhos.

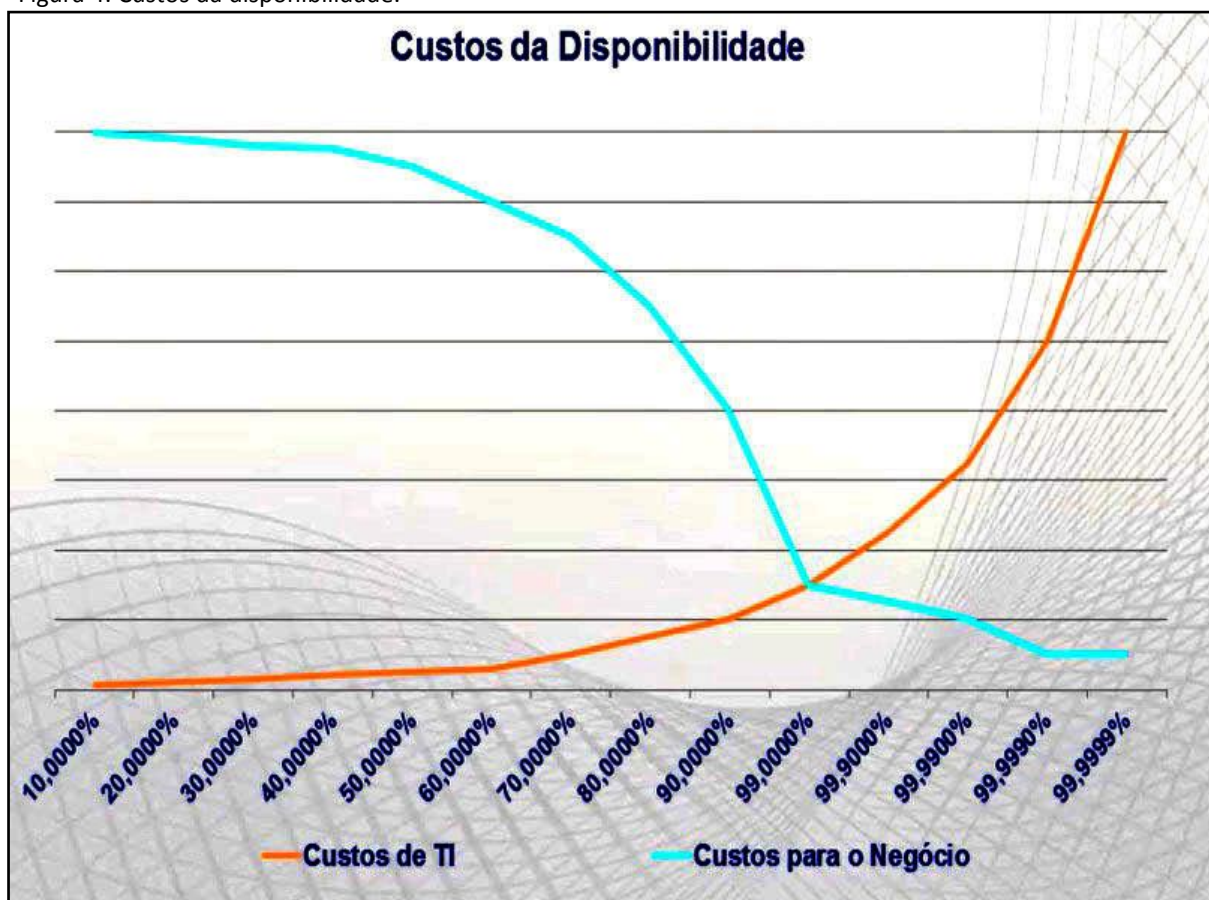
Os sistemas de alta disponibilidade têm como objetivo proteger as empresas contra fatores que possam vir a causar falhas nos mesmos, como por exemplo: falhas de *hardware*, falhas de *software*, falhas de rede, falta de energia e desastres naturais.

Como demonstrado na Figura 4, os custos adicionais para a implantação de um sistema de alta disponibilidade se tornam quase que irrelevantes se forem comparados aos seus benefícios, pois combatem prejuízos não somente internos como também externos. A falta de disponibilidade pode vir a causar queda na produtividade, perda da confiança nos sistemas e aplicações, perda de vendas, perdas de clientes e até mesmo perda de espaço de mercado, pois falhas nos serviços prejudicam a imagem institucional.

A Alta Disponibilidade está ligada diretamente à crescente dependência dos computadores, pois atualmente eles possuem um papel crítico, principalmente em empresas cuja maior funcionalidade é exatamente a oferta de alguns serviços computacionais, como *ebusiness*, notícias, sites *web*, banco de dados, dentre outros.(PITANGA, 2008).



Figura 4: Custos da disponibilidade.



Fonte: AGUIAR, 2012.

#### 4.1 Como medir a disponibilidade

Segundo (Brandão, 2011) calcular a disponibilidade de um servidor existe uma fórmula muito simples: mede-se o tempo que os serviços ficaram disponíveis (*uptime*) e tempo pelo qual os mesmos estavam fora do ar (*downtime*). A disponibilidade é dada por:

$$\text{Disponibilidade (\%)} = (\text{Uptime} - \text{Downtime}) / \text{Uptime}.$$

A título de exemplo, vamos considerar que, num período de dois anos, um servidor esteve fora do ar por cinco horas. Neste caso, a disponibilidade foi:

$$\text{Uptime} = 2 \text{ anos } (365 \times 24 \times 2 = 17.520 \text{ horas})$$

*Downtime* = 5 horas.

Disponibilidade =  $(17520 \text{ horas} - 5 \text{ horas}) / 17520 \text{ horas} = 99,9715 \%$

## 4.2 Dependabilidade

Para alcançar um ambiente com tolerância a falhas também é preciso pensar em dependabilidade, que é uma tradução literal da língua inglesa referente ao termo *dependability*. O objetivo, é fornecer a entrega de um serviço de forma confiável, evitando os defeitos mais frequentes e medindo a qualidade e a confiança depositada nele.

Segundo (AVIZIENIS ET al, 2004) os principais atributos da dependabilidade são: confiabilidade, disponibilidade, segurança de funcionamento (*safety*), segurança (*security*) e performance (*performability*), demonstrados na Tabela 1.

Dependendo das aplicações pretendidas pelo sistema, diferentes tipos de atributos da dependabilidade podem ser definidos, tais como:

- A prontidão de uso leva à disponibilidade;
- Continuidade do serviço leva à confiabilidade;
- O impedimento de revelações desautorizadas de informações leva à confidencialidade;
- A proteção contra alterações impróprias nas informações leva à integridade;
- A proteção contra catástrofes ao ambiente e ao usuário leva à segurança;
- A possibilidade de submeter o sistema a reparos e evoluções leva à sustentabilidade.

Tabela 1: Atributos da dependabilidade

<b>Dependabilidade (<i>dependability</i>)</b>	Qualidade do serviço fornecido por um dado sistema.
<b>Confiabilidade (<i>reliability</i>)</b>	Capacidade de atender a especificação, dentro de condições definidas, durante certo período de funcionamento e condicionado a estar operacional durante o período.
<b>Disponibilidade (<i>availability</i>)</b>	Probabilidade de o sistema estar operacional num instante de tempo determinado; alternância de períodos de funcionamento e reparo.
<b>Segurança (<i>safety</i>)</b>	Probabilidade de o sistema estar operacional e executar sua função corretamente ou descontinuar suas funções de forma a não provocar danos a outros sistema ou pessoas que dele dependam.
<b>Segurança (<i>security</i>)</b>	Proteção contra falhas maliciosas, visando privacidade, autenticidade, integridade e não repúdio dos dados.

Fonte: WEBER, 2002.

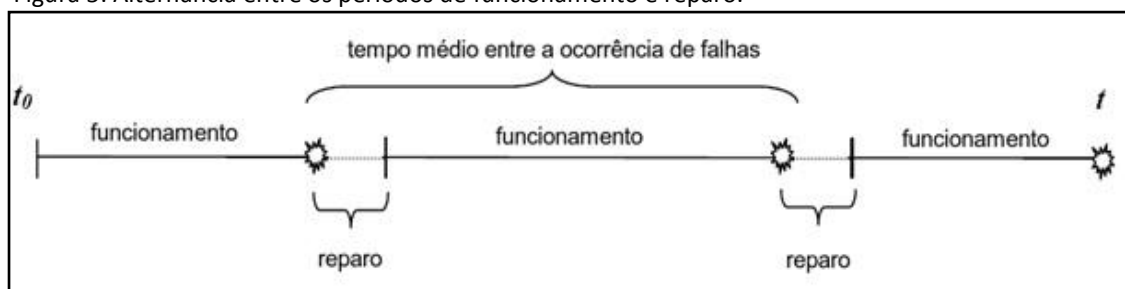
#### 4.2.1 Disponibilidade

Disponibilidade é a probabilidade de se encontrar recursos e/ou aplicações em perfeito estado e prontas para serem usadas em um determinado momento. De acordo com (PITANGA, 2008) máquinas que apresentam um percentual de disponibilidade entre quatro e cinco nozes (99.99% e 99.999%), tendem a ficar inativas somente cinco minutos à uma hora dentro de um ano de funcionamento.

Disponibilidade é o fator mais buscado e desejado dentro de um sistema de missão crítica. Na Tabela 2 estão algumas representações percentuais dos nozes de disponibilidade.

A disponibilidade também pode ser representada através da probabilidade, dada pela função  $A(t)$ , onde  $t$  representa o tempo operante de um sistema, como demonstrado na Figura 5. Um servidor mesmo que apresente vários nozes de disponibilidade, não seria viável se o *downtime* fosse apenas um instante e não curtos períodos, pois quanto maior o período inoperante, maior é o comprometimento da qualidade do serviço.

Figura 5: Alternância entre os períodos de funcionamento e reparo.



Fonte: WEBER, 2002.

Tabela 2: Medidas de disponibilidade

Código	Uptime	Downtime	Downtime por semana	Downtime por ano
1	90%	10%	16 horas, 48 minutos	36,5 dias
2	98%	2%	3 h, 21m, 36s	7,3 dias
3	99%	1%	1 hora, 41 minutos	3,65 dias
4	99,8%	0.2%	20 minutos, 9 segundos	17 horas, 31 minutos
5	99,9%	0.1%	10 minutos, 5 segundos	8 horas, 45 minutos
6	99,99%	0.01%	1 minuto	52,6 minutos
7	99,999%	0.001%	6 segundos	5,26 minutos
8	99,9999%	0.0001%	0,6 segundos	31,5 segundos

Fonte: PEREIRA, 2004.

Deste modo, um servidor com quatro nozes de disponibilidade significa que ele esteve ativo 99,99% do tempo e inativo por 0.01%, ou seja, esteve fora do ar 52,6 minutos em um intervalo de um ano ou aproximadamente um minuto por semana.

#### 4.2.2 Confiabilidade

Confiabilidade é a probabilidade de um sistema executar determinados processos e/ou aplicações sem falhas, de forma que se torne confiável para o usuário.

Segundo (WEBER, 2002) existem quatro medidas principais de confiabilidade: taxa de defeito, *mean time to failure* (MTTF), *mean time to repair* (MTTR), *mean time between failure* (MTBF), definidas na Tabela 3.

Tabela 3: Medidas de confiabilidade.

<b>Medida</b>	<b>Significado</b>
Taxa de defeitos - <i>failure rate, hazard function, hazard rate</i>	Número esperado de defeitos em um dado período de tempo, assumido um valor constante durante o tempo de vida útil do componente.
MTTF - <i>mean time to failure</i> (Tempo médio para falha)	Tempo esperado até a primeira ocorrência de defeito.
MTTR - <i>mean time to repair</i> (Tempo médio para reparo)	Tempo médio para reparo do sistema.
MTBF - <i>mean time between failure</i> (Tempo médio entre falha)	Tempo médio entre as falhas do sistema.

Fonte: WEBER, 2002.

É obrigação do fabricante fornecer tais medidas para os seus produtos, que são determinadas estatisticamente de acordo com os componentes e dispositivos fabricados, sejam eles eletrônicos ou sistemas computacionais mais complexos.

### 4.3 Níveis de alta disponibilidade

O autor (RESNICK, 1996) define uma divisão quanto aos níveis de alta disponibilidade: disponibilidade básica, alta disponibilidade e disponibilidade continuada.

#### 4.3.1 Disponibilidade Básica

Disponibilidade básica é o primeiro nível de alta disponibilidade e encontra-se em máquinas comuns, implementadas com componentes (*hardware* e *software*) suficientes para satisfazer as necessidades de seu uso. Não possui habilidade de mascarar falhas, processo este que consegue impedir a visualização de uma falha por um observador externo (ou usuário), ou seja, não chega a impedir a falha e nem a sua observação. Este nível apresenta um percentual de disponibilidade 99% a 99,9%, ou seja, os equipamentos ou

sistemas desta categoria podem ficar fora do ar oito horas e quarenta e cinco minutos por ano.

### **4.3.2 Alta disponibilidade**

Alta disponibilidade encontra-se no segundo nível que, assim como a disponibilidade básica, consiste em máquinas compostas por componentes suficientes para o seu funcionamento, porém com o diferencial de possuírem redundância, adicionando assim a capacidade de mascarar algumas falhas. Este nível apresenta um percentual de disponibilidade de 99,99% a 99,999% podendo, portanto ficar fora do ar por um período de 5 a 52 minutos por ano.

### **4.3.3 Disponibilidade Continuada**

Com o aumento dos níveis, é possível obter disponibilidades cada vez mais próximas dos 100%. O diferencial da disponibilidade continuada é que todas as paradas, sejam elas planejadas ou não, são mascaradas através de redundância e replicação deixando assim o sistema sempre disponível. A disponibilidade continuada apresenta um percentual de 99,9999 ou mais, o que significa no máximo trinta e dois segundos de parada por ano.

## 5 FERRAMENTAS UTILIZADAS

Neste capítulo são abordados aplicativos e ferramentas auxiliares que possibilitam a criação de um ambiente de alta disponibilidade. São enfatizados o *Heartbeat*, que possibilita monitoração entre os servidores primários e secundários, para que em caso de falhas haja a mudança entre servidor falho e operante, o DRBD que permite espelhamento entre partições de dados via rede, e Mon que é um serviço de monitoramento da disponibilidade que têm como objetivo enviar alarmes em caso de falhas.

### 5.1 Heartbeat

O *Heartbeat* nasceu de um projeto chamado *High Availability Linux Project* (Projeto Alta Disponibilidade *Linux*), cujo maior objetivo é desenvolver soluções GNU/Linux com o intuito de promover confiabilidade, disponibilidade e suportabilidade (*LINUX-HA*, 2011).

O *Heartbeat* é responsável por monitorar os servidores presentes no *cluster* para que, em caso de falhas no nó primário, sejam elas de *hardware* ou *software*, seja realizado automaticamente os procedimentos necessários para disponibilizar os recursos e/ou aplicativos utilizados no nó secundário. Preservando assim a alta disponibilidade do *cluster*.

O *software Heartbeat* monitora os servidores trocando mensagens periódicas com eles, verificando assim se estão ativos. Este funcionamento lembra a monitoração dos batimentos cardíacos, daí o nome *Heartbeat*.

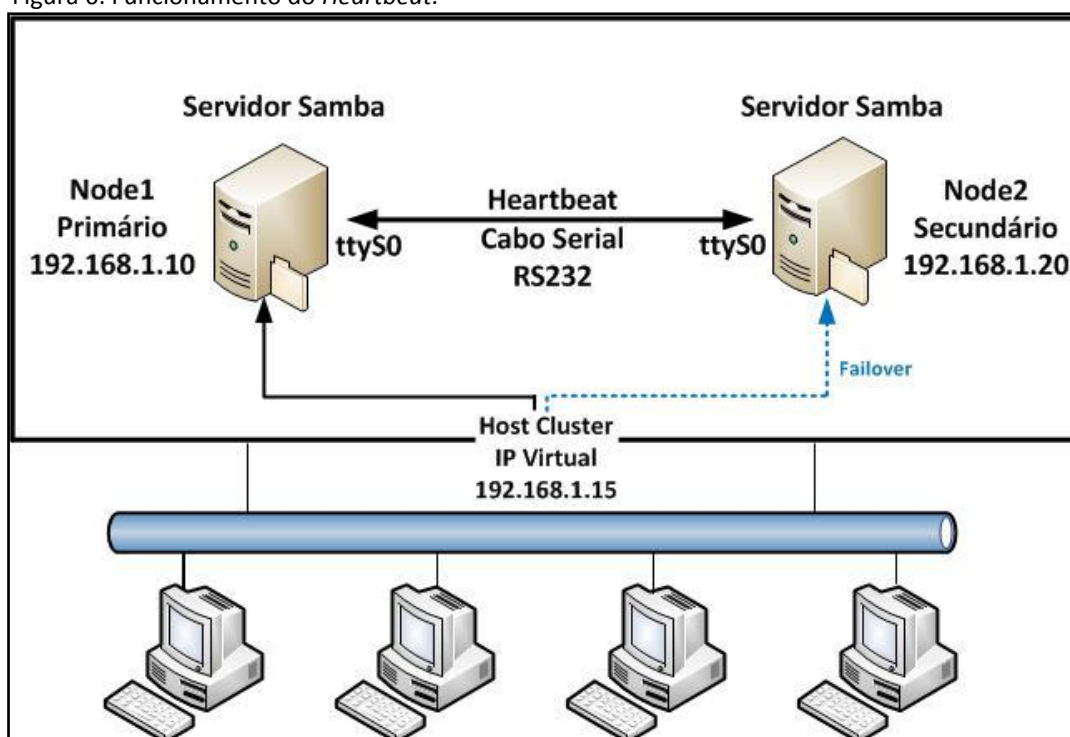
Para tal serviço pode ser utilizado uma ou mais conexões físicas entre os servidores, sejam elas conexões *ethernet* normais, interfaces dedicadas ligadas por cabo *crossover* ou interfaces seriais.

Utilizando-se do conceito primário e secundário, um servidor atende as requisições como ativo (primário) e o outro, como passivo (secundário), que fica em espera para assumir

os serviços caso o “coração” do nó primário pare de bater.

Para tal monitoramento é altamente recomendado uma conexão dedicada com cabo *crossover* ou cabo serial, pois através de uma comunicação normal passa muito tráfego de dados das aplicações que ele suporta o que poderia ocasionar interferências indesejadas ou até mesmo gargalos, fazendo assim com que a conexão se torne lenta.

Figura 6: Funcionamento do *Heartbeat*.



Fonte: Autoria própria.

Na Figura 6, o endereço IP 192.168.1.15 é um IP virtual gerenciado pelo *Heartbeat*. É por este IP que os clientes têm acesso ao serviço Samba do servidor em produção. Os endereços IPs 192.168.1.10 e 192.168.1.20 são fixos e usados apenas para permitir o gerenciamento remoto, independente de qual servidor estiver com o IP virtual configurado.

Com base na Figura 6, é possível detalhar os passos de funcionamento do *Heartbeat*:

- O *Heartbeat* do **servidor secundário** “conversa” com o **servidor primário** enviando uma mensagem através da conexão serial assíncrona RS232;



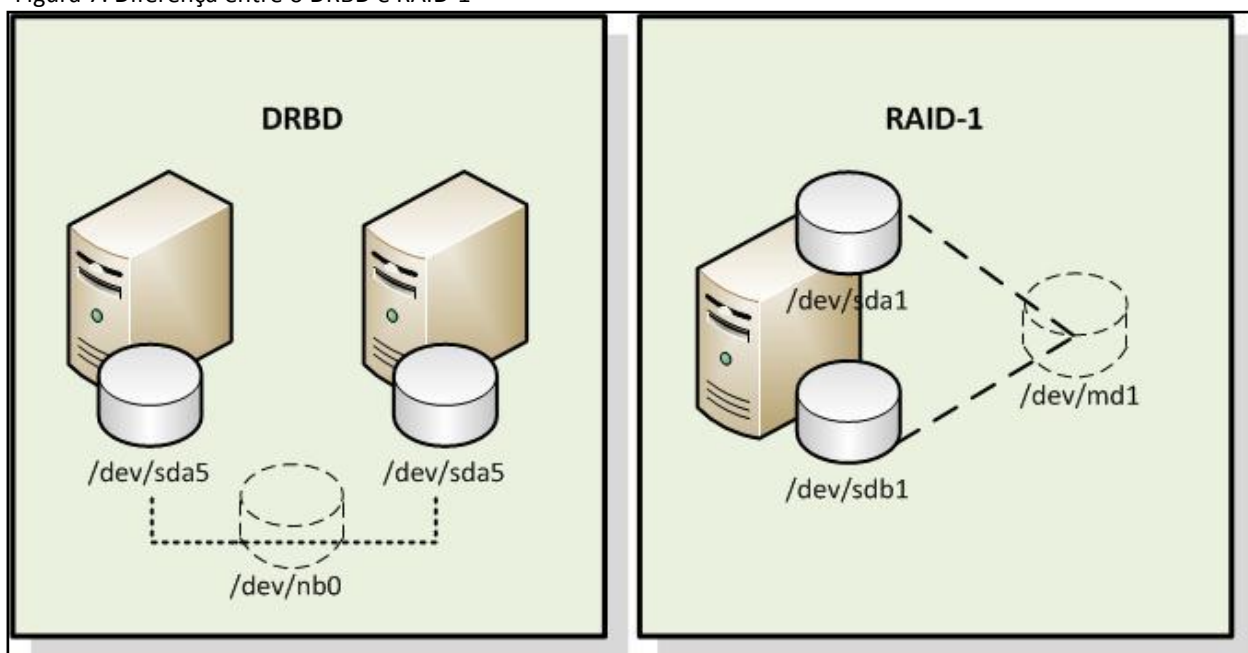
- Caso ocorra alguma falha na comunicação entre os servidores, automaticamente o *Heartbeat* do **servidor secundário** inicia o serviço Samba e configura o IP **192.168.1.15**, passando assim a funcionar como o servidor em produção;
- Quando o **servidor primário** estiver novamente disponível, através da comunicação serial irá enviar um pacote avisando sobre sua disponibilidade para o **servidor secundário**;
- O **servidor secundário** então para o serviço Samba, libera o IP **192.168.1.15**, configurado anteriormente e envia uma mensagem para o **servidor primário**, avisando-o sobre a possibilidade de ele assumir novamente os serviços;
- O **servidor primário** inicia o Samba e configura novamente o IP **192.168.1.15**, voltando assim a ser o servidor em produção.

## 5.2 Distributed Replicated Block Device (DRBD)

Segundo o (DRBD, 2008) o dispositivo de bloco distribuído replicado (DRBD) é um *software* baseado em replicação e espelhamento de conteúdo de dispositivos de bloco como, discos rígidos, partições, volumes lógicos etc; entre servidores. O DRBD resume-se em um módulo do *kernel* do sistema operacional *Linux* que, em conjunto com *scripts* e programas, disponibilizam blocos de armazenamento distribuídos, altamente utilizado em *clusters* de alta disponibilidade.

Semelhante ao funcionamento do RAID-1 (*Redundant Array of Inexpensive Disks*) que trabalha com espelhamento entre dois discos localizados em uma mesma máquina (MORIMOTO, 2007), o DRBD também opera espelhando dados, porém entre servidores interligados em rede, como ilustrado na Figura 7.

Figura 7: Diferença entre o DRBD e RAID-1



Fonte: Autoria própria.

Segundo (DRBD, 2008b), o DRBD pode replicar os dados de três formas:

- **Tempo real (*real time*):** A replicação dos dados acontece em tempo real ou seja, à medida que são modificados em um dos dispositivos, automaticamente ocorre a replicação dos mesmos para os demais dispositivos.
- **Transparente:** Aplicações não precisam estar cientes de que os dados são armazenados em vários *hosts*.
- **Síncrona ou Assíncrona:** Através do espelhamento síncrono, as aplicações são notificadas assim que o processo de gravação for concluído em todos os *hosts*. Já no espelhamento assíncrono, a notificação acontece logo após a conclusão das gravações obterem sucesso localmente, o que geralmente ocorre antes que tenham se propagado para os demais *hosts*.

O DRBD utiliza-se do protocolo *Transmission Control Protocol/Internet Protocol* (TCP/IP) para realizar transferências entre os servidores, recurso este que acaba se tornando uma vantagem, pois permite a sua instalação em máquinas geograficamente afastadas.

Os servidores que fazem parte do DRBD são configurados de maneira que cada um receba um estado, que pode ser definido como primário ou secundário, em ambos é criado uma conexão entre um dispositivo virtual (/dev/nbX) e uma partição local (/dev/hdaX) que é o dispositivo de bloco de nível mais baixo (físico).

A partição local não é acessada diretamente, a escrita dos dados é realizada no dispositivo virtual do servidor primário, que tem o trabalho de realizar a transferência dos mesmos para uma partição local e replicá-los para os demais servidores que se encontram no nível secundário. Imediatamente ao receberem as informações, os servidores secundários fazem a escrita dos dados em suas partições locais (dispositivos físicos).

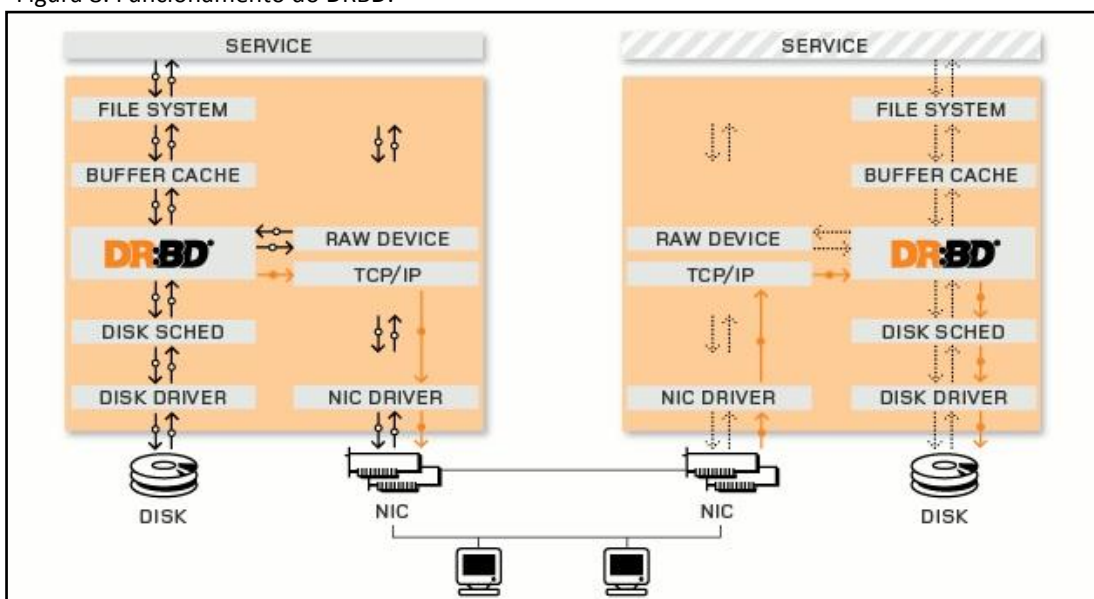
Resumidamente o que o DRBD faz é criar um espelho entre os dispositivos usando a rede como meio de transmissão. O dispositivo primário é utilizado sem restrições tanto para leitura e escrita de operações, já o secundário recebe todas as atualizações do nó primário não podendo ser usado por aplicativos, pois há a necessidade de se manter uma coerência de *cache*, o que seria impossível, se um recurso secundário fosse utilizado por outras aplicações.

Se o servidor primário vier a falhar, de imediato o *Heartbeat* avisará o DRBD sobre a falha e fará a mudança do servidor primário para secundário e todo o processo começará a ser realizado no servidor secundário.

Assim que o servidor que falhou voltar à ativa, ele assumirá o papel de servidor secundário e começará a sincronização dos dados com o servidor agora primário. Todo o processo é realizado em segundo plano, e sem a necessidade de interrupção do serviço.

O *Heartbeat* também é responsável por montar o sistema de arquivos na nova máquina que se tornou primária.

Figura 8: Funcionamento do DRBD.



Fonte: DRBD, 2008.

Na Figura 8, as duas caixas representam dois servidores que formam um cluster de HA. As caixas contêm os componentes habituais do *kernel* Linux (sistema de arquivos), memória *cache*, disco, *drivers* de disco, pilha TCP/IP e *Network Interface Card* (NIC). As setas pretas mostram o fluxo de dados entre esses componentes.

As setas laranja mostram o fluxo de dados, como DRBD espelha os dados do serviço altamente disponível do servidor primário e do servidor secundário do grupo HA.

### 5.2.1 Ferramentas de administração

O DRBD possui um conjunto de ferramentas que ajudam o usuário a administrar e configurar recursos (DRBD, 2008c):

- **Drbdadm:** Ferramenta de administração de alto nível do conjunto de programas DRBD utilizado para administração dos dispositivos. Permite que o espelhamento seja iniciado e parado, além de fornecer estatísticas sobre o espelhamento;

- **Drbdsetup:** Permite aos usuários configurar o módulo DRBD que foi carregado no *kernel*. É a ferramenta de baixo nível dentro do conjunto de programas DRBD. Ao usar este programa, todos os parâmetros de configuração devem ser entregues diretamente na linha de comando;
- **Drbdmeta:** Permite aos usuários criar, gravar, restaurar e modificar estruturas de meta-dados do DRBD;
- **/etc/drbd.conf:** Arquivo onde são armazenadas todas as configurações do aplicativo.

### 5.2.2 Sincronismo

Conforme (DRBD, 2008a) o DRBD suporta três modos de replicação, permitindo três diferentes tipos de sincronismo. Para tanto, utiliza-se de três protocolos, chamados A, B, C, conforme segue:

- **Protocolo A:** O servidor primário realiza uma operação de entrada/saída no disco local, e notifica os servidores secundários sobre a mudança. Este tipo de protocolo apresenta o menor nível de confiabilidade entre os três, pois ele não procura saber se o sincronismo foi ou não bem sucedido. A escrita é considerada completa logo que o dado é escrito no disco e enviado pela rede.
- **Protocolo B:** Apresenta um nível de confiabilidade um pouco maior, se comparado ao protocolo A. Após o término de uma execução de entrada/saída ser realizado no servidor primário, é enviado um aviso aos servidores secundários e o processo só é dado como completo quando confirmado o recebimento pelos demais servidores, porém o servidor primário não aguarda que a gravação seja realizada nos discos locais dos servidores secundários.

- **Protocolo C:** De longe o protocolo que apresenta o maior nível de confiabilidade, pois as operações são realizadas de modo sincronizado. O servidor primário somente dá como completo uma operação de entrada/saída no disco local, quando receber uma notificação dos outros servidores dizendo que também terminaram a gravação dos dados em seus discos locais.

### 5.3 Mon

De acordo com (TROCKI, 2008) Mon é um *software* de código aberto que tem como finalidade monitorar a disponibilidade de serviços. Caso seja detectada alguma falha é possível enviar um alerta para o administrador da rede, caso haja um *e-mail* configurado, e até mesmo acionar o *Heartbeat* para que ele tome alguma decisão para a solução do problema.

O Mon possui métodos de alertas através de uma interface comum, que são facilmente implementadas através de *scripts* escritos em *Perl*, *shell* ou até mesmo em linguagem C.

## 6 ESTUDO DE CASO: CONFIGURANDO UM SERVIDOR SAMBA COM ALTA DISPONIBILIDADE

Neste estudo de caso será abordado a implementação de um sistema de alta disponibilidade usando o sistema operacional *Linux*.

Para as empresas certos arquivos e informações são praticamente a alma do negócio, se tornam o diferencial de sucesso de cada uma. São de extrema importância para a introdução de novas tecnologias e se exploradas de maneira correta se transformam em oportunidades de novos investimentos.

Portando este estudo de caso tem como intuito demonstrar uma proposta que se torna atraente para as empresas que desejam obter tal disponibilidade em seus arquivos, aplicações e/ou recursos, utilizando ferramentas *open-source*, ou seja, de baixo custo.

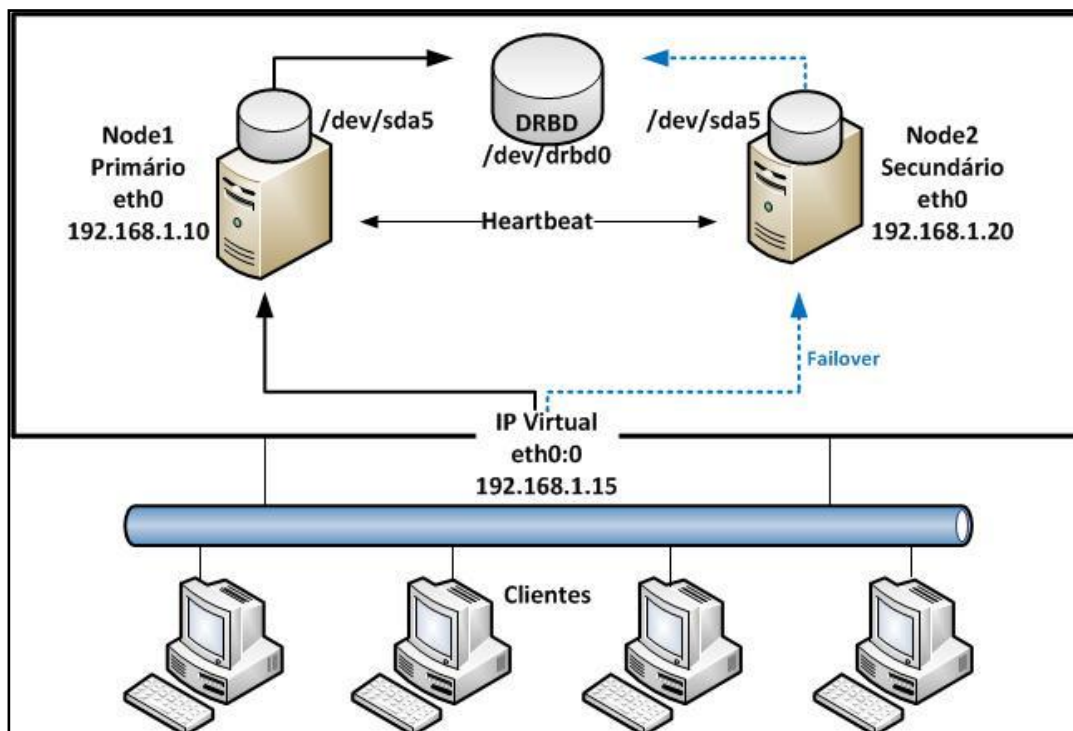
A fim de testar o *cluster* de alta disponibilidade iremos utilizar a ferramenta Samba, de forma a comprovar o funcionamento das demais ferramentas utilizadas para formar o *cluster* de alta disponibilidade. Como por exemplo:

- Testar o sincronismo do DRBD;
- Testar o *failover* do *Heartbeat*;
- Testar a monitoração do Mon;

Ou seja, o Samba necessariamente não faz parte do *cluster* de HA, mas será utilizado para demonstrar o funcionamento das ferramentas que são utilizadas para formar um ambiente altamente disponível.

A Figura 9 demonstra a arquitetura utilizada.

Figura 9: Arquitetura do estudo de caso.



Fonte: Autoria própria.

## 6.1 Samba

Segundo (MORIMOTO, 2010) o projeto Samba nasceu em 1991 de forma acidental. Desenvolvido por um australiano chamado Andrew Tridgell que na época era estudante do curso de ciências da computação da Universidade Nacional da Austrália.

Ele precisava executar um *software* chamado “*excursion*” em conjunto com um *software* chamado *patchworks*, especializado em compartilhamento de arquivos. O problema era que o *patchworks* utilizava um protocolo com pouquíssima documentação disponível.

Foi então que ele começou a estudar o tal protocolo e desenvolveu um servidor que pudesse ser executado em seu computador, que era capaz de comunicar-se com os clientes executando o *patchworks*. Primeiramente o objetivo era interligar dois computadores, um com suporte a *MicroSoft Disk Operating System* (MS-DOS) e outro com Solaris.



Após algum tempo, Andrew recebeu um *e-mail* afirmando que seu programa também era capaz de executar junto de um *software* da *Microsoft* chamado *LanManager*, permitindo assim compartilhar arquivos entre servidores Unix com computadores com MS-DOS.

O protocolo usado pelo *patchworks* que até então se mostrava “obscuro”, se revelou uma implementação do protocolo *Server Message Block* (SMB), utilizado pela *Microsoft* em seu produto *LanManager*.

Foi assim que nasceu o Samba, que é um serviço de compartilhamento de arquivos para sistemas operacionais *Unix* e *Linux*, mas que permite o gerenciamento e compartilhamento de recursos em redes formadas por computadores *Microsoft Windows*.

Além de compartilhar arquivos e impressoras, o Samba também pode atuar como *Primary Domain Controller* (PDC), que é um serviço de controle de domínios, permitindo a autenticação e compartilhamento de recursos com computadores com o *Windows*.

## 6.2 Hardware Utilizado

O estudo de caso foi projetado em 3 máquinas virtuais, 2 servidores *Linux* e um cliente *Windows XP*. A Tabela 4 mostra o *hardware* dos servidores e do cliente *Windows XP*, e a Tabela 5 mostra a descrição dos mesmos.

Tabela 4: Hardware dos servidores.

Hardware	Servidor primário	Servidor secundário	Cliente Windows XP
CPU	Intel core i3 M350 2,27 GHz	Intel core i3 M350 2,27 GHz	Intel core i3 M350 2,27 GHz
Memória RAM	512 MB	512 MB	512 MB
HD	10GB	10GB	10GB

Fonte: Autoria própria.

Tabela 5: Descrição das máquinas utilizadas.

Descrição	Servidor primário	Servidor secundário	Cliente
Sistema Operacional	Debian GNU/Linux 6.0	Debian GNU/Linux 6.0	Windows XP
Hostname	Node1	Node2	Cliente
HD	5,3GB para a partição / 3,7GB para a partição /cluster 1GB para swap	5,3GB para a partição / 3,7GB para a partição /cluster 1GB para swap	10GB
IP	eth0 – 192.168.1.10 eth0:0 – 192.168.1.15 (IP virtual quando respondendo pelo cluster) eth1 – DHCP através da máquina hospedeira, usado para se comunicar com a internet.	eth0 – 192.168.1.10 eth0:0 – 192.168.1.15 (IP virtual quando respondendo pelo cluster) eth1 – DHCP através da máquina hospedeira, usado para se comunicar com a internet.	192.168.1.30
Serviços	DRBD, Heartbeat, Mon, Samba	DRBD, Heartbeat, Mon, Samba	

Fonte: Autoria própria.

### 6.3 Configurando o Heartbeat

Todos os arquivos de configuração do *Heartbeat* devem ser idênticos em ambos os servidores. Para a instalação do *Heartbeat* é usado o comando a seguir.

```
#apt-get install heartbeat
```

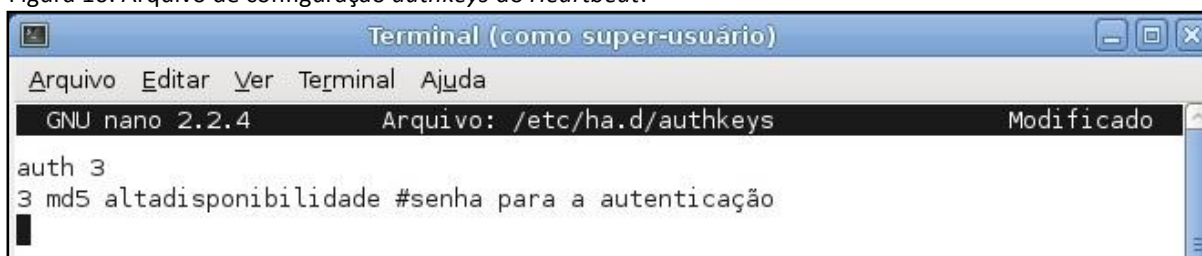
Após a finalização da instalação é necessário copiar os arquivos de configuração da área de documentação do *Heartbeat*, pois no *Debian* esses arquivos não vêm no diretório de configuração por *default*.

```
#cd /usr/share/doc/heartbeat
#cp authkeys ha.cf.gz haresources.gz /etc/ha.d
#cd /etc/ha.d
#gunzip ha.cf.gz haresources.gz
```

Em seguida vamos configurar o arquivo *authkeys*, que é responsável pela autenticação entre os servidores. Como demonstrado na Figura 10.

```
#nano /etc/ha.d/authkeys
```

Figura 10: Arquivo de configuração *authkeys* do *Heartbeat*.



```
Terminal (como super-usuário)
Arquivo  Editar  Ver  Terminal  Ajuda
GNU nano 2.2.4  Arquivo: /etc/ha.d/authkeys  Modificado
auth 3
3 md5 altadisponibilidade #senha para a autenticação
```

Fonte: Autoria própria.

É necessário atribuir permissão de leitura e escrita para o arquivo *authkeys*.

```
#chmod 600 authkeys
```

O principal arquivo de configuração do *Heartbeat* é o *ha.cf*, onde é atribuído qual *interface* será responsável pela comunicação entre os servidores, os diretórios de *logs*, tempo mínimo para declarar a outra máquina como inativa etc.

```
#nano /etc/ha.d/ha.cf
```

Arquivo *ha.cf*:

```
node node1 #Servidor primário
node node2 #Servidor secundário

udp eth0 #Interface que faz comunicação de verificação entre os servidores

debugfile /var/log/ha-debug #Caminho do arquivo Debug
logfile /var/log/ha-log #Caminho do arquivo de Log

keepalive 1 #Intervalo em segundos entre os pings

deadtime 3 #Intervalo em segundos para declarar uma máquina inativa

warntime 3 #Tempo para notificar no log

initdead 90 #Permite inicialização em todos os nodes

udpport 694 #Porta utilizada
```

```

auto_failback on #Volta do servidor primário caso haja falha e depois volte a
responder

compression bz2 #Compactação dos dados

compression_threshold 2 #Compactação dos dados

```

O *Heartbeat* possui um terceiro arquivo de configuração, que gerencia os *daemons* de responsabilidade do *Heartbeat*, como por exemplo: atribuir um IP virtual para o *cluster* e ativar os serviços no servidor secundário em caso de falha do servidor primário. Arquivo demonstrado na Figura 11.

```
#nano /etc/ha.d/haresources
```

Figura 11: Arquivo de configuração *haresources* do *Heartbeat*.



Fonte: Autoria própria.

O significado dos parâmetros são os seguintes:

- node1 - nome da máquina principal
- drbddisk - utilitário do *Heartbeat* para gerenciar o DRBD
- dados - nome do dispositivo do DRBD (configurado no *drbd.conf*)
- *filesystem* - utilitário para montagem de partição
- /dev/drbd0 - nome da unidade do DRBD
- /cluster - nome do local de montagem do disco do DRBD
- 192.168.1.15 - IP virtual
- samba - *script* do *init.d* para o samba

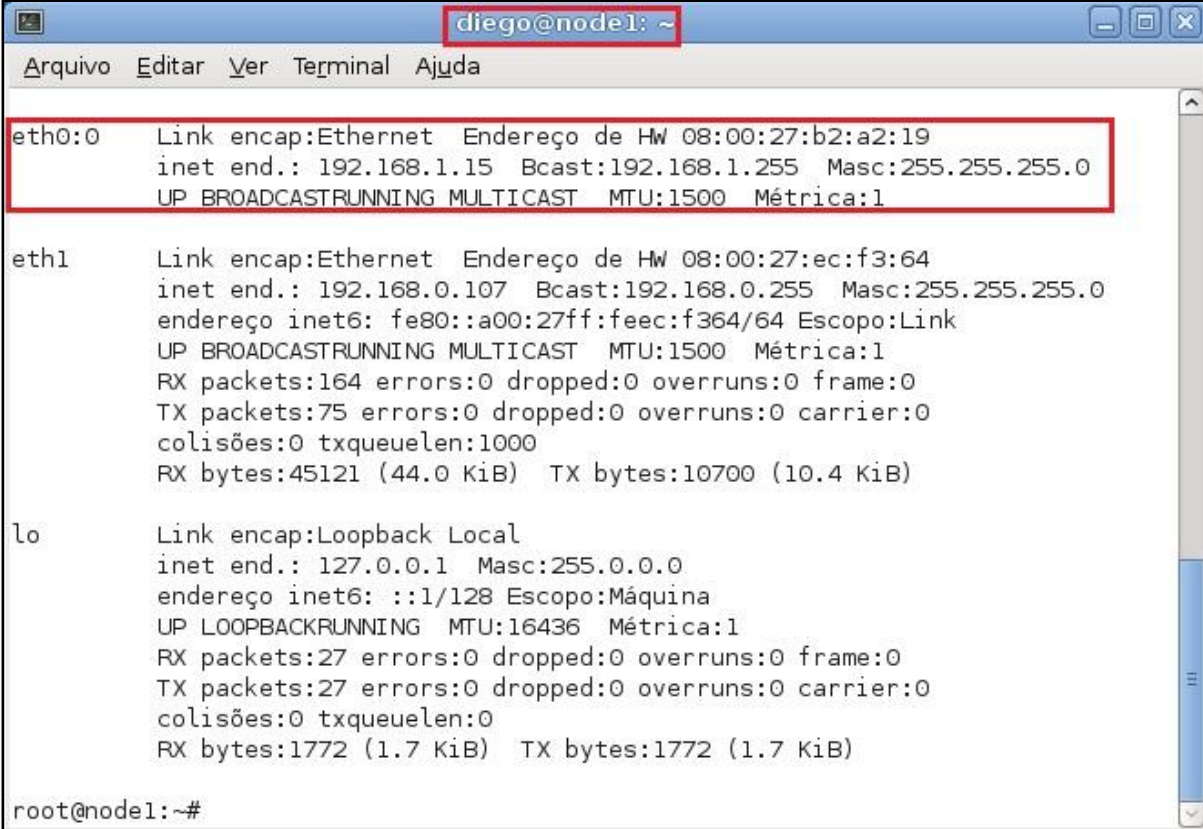
Logo após o termino da configuração é necessário reiniciar o serviço do *Heartbeat*.

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

Assim que o serviço estiver ativo novamente, o IP virtual do *cluster* de alta disponibilidade atribuído no arquivo */etc/ha.d/haresources*, já estará em uso no servidor

primário, como demonstrado na Figura 12.

Figura 12: IP virtual em funcionamento no servidor primário.



```
diego@node1: ~
Arquivo  Editar  Ver  Terminal  Ajuda

eth0:0  Link encap:Ethernet  Endereço de HW 08:00:27:b2:a2:19
        inet end.: 192.168.1.15  Bcast:192.168.1.255  Masc:255.255.255.0
        UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1

eth1    Link encap:Ethernet  Endereço de HW 08:00:27:ec:f3:64
        inet end.: 192.168.0.107  Bcast:192.168.0.255  Masc:255.255.255.0
        endereço inet6: fe80::a00:27ff:feec:f364/64  Escopo:Link
        UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
        RX packets:164 errors:0 dropped:0 overruns:0 frame:0
        TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
        colisões:0 txqueuelen:1000
        RX bytes:45121 (44.0 KiB)  TX bytes:10700 (10.4 KiB)

lo      Link encap:Loopback Local
        inet end.: 127.0.0.1  Masc:255.0.0.0
        endereço inet6: ::1/128  Escopo:Máquina
        UP LOOPBACKRUNNING  MTU:16436  Métrica:1
        RX packets:27 errors:0 dropped:0 overruns:0 frame:0
        TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
        colisões:0 txqueuelen:0
        RX bytes:1772 (1.7 KiB)  TX bytes:1772 (1.7 KiB)

root@node1:~#
```

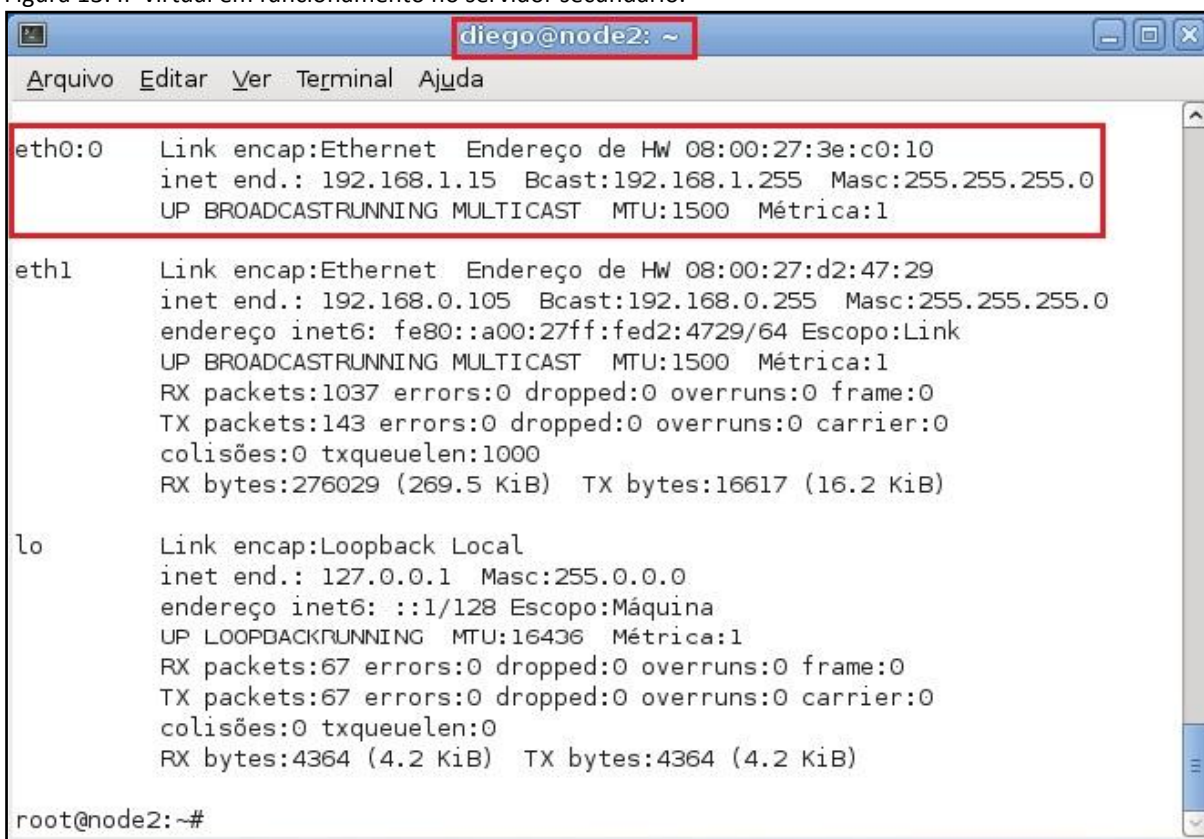
Fonte: Autoria própria.

A fim de realizar um teste, vamos desligar o servidor primário e conferir se o servidor secundário assumirá o IP virtual.

```
#shutdown -h now
```

Como demonstrado na Figura 13, o servidor secundário assumiu o IP virtual em poucos segundos.

Figura 13: IP virtual em funcionamento no servidor secundário.



```

diego@node2: ~
Arquivo Editar Ver Terminal Ajuda

eth0:0   Link encap:Ethernet  Endereço de HW 08:00:27:3e:c0:10
         inet end.: 192.168.1.15  Bcast:192.168.1.255  Masc:255.255.255.0
         UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1

eth1     Link encap:Ethernet  Endereço de HW 08:00:27:d2:47:29
         inet end.: 192.168.0.105  Bcast:192.168.0.255  Masc:255.255.255.0
         endereço inet6: fe80::a00:27ff:fed2:4729/64  Escopo:Link
         UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
         RX packets:1037 errors:0 dropped:0 overruns:0 frame:0
         TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
         colisões:0 txqueuelen:1000
         RX bytes:276029 (269.5 KiB)  TX bytes:16617 (16.2 KiB)

lo       Link encap:Loopback Local
         inet end.: 127.0.0.1  Masc:255.0.0.0
         endereço inet6: ::1/128  Escopo:Máquina
         UP LOOPBACKRUNNING  MTU:16436  Métrica:1
         RX packets:67 errors:0 dropped:0 overruns:0 frame:0
         TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
         colisões:0 txqueuelen:0
         RX bytes:4364 (4.2 KiB)  TX bytes:4364 (4.2 KiB)

root@node2:~#

```

Fonte: Autoria própria.

## 6.4 Configurando o DRBD

O comando a seguir é usado para a instalação do DRBD e deverá ser executado tanto no servidor primário como no secundário.

```
#apt-get install drbd8-utils
```

Depois de instalado é necessário conferir se os dispositivos do DRBD foram criados em /dev, pois nem todas as distribuições criam os mesmos por *default*. Para tal conferência usamos o comando:

```
# ls /dev/drbd*
```

Se o resultado for como demonstrado na Figura 14 os dispositivos estão presentes, se não estiverem lá será necessários criar manualmente, o que pode ser feito com o comando a seguir (cria 15 dispositivos por padrão).

```
# for i in $(seq 0 15) ; do mknod /dev/drbd$i b 147 $i;done
```

Figura 14: Dispositivos do drbd.



```
Terminal (como super-usuário)
Arquivo  Editar  Ver  Terminal  Ajuda
root@node1:/home/diego# ls /dev/drbd*
/dev/drbd0  /dev/drbd11  /dev/drbd14  /dev/drbd3  /dev/drbd6  /dev/drbd9
/dev/drbd1  /dev/drbd12  /dev/drbd15  /dev/drbd4  /dev/drbd7
/dev/drbd10 /dev/drbd13  /dev/drbd2   /dev/drbd5  /dev/drbd8

/dev/drbd:
by-disk by-res
root@node1:/home/diego#
```

Fonte: Autoria própria.

O principal arquivo de configuração do DRBD é o `/etc/drbd.conf`, que contém informações como o endereço IP da máquina principal e secundária, a taxa de transferência da sincronização, qual protocolo do DRBD será utilizado, endereço do dispositivo virtual, endereço do dispositivo a ser replicado etc. O arquivo `drbd.conf` deve ser exatamente igual nos dois servidores, para configurá-lo usamos o comando:

```
#nano /etc/drbd.conf
```

Arquivo de configuração deve conter as seguintes linhas:

```
global { usage-count yes; }

# Velocidade de transferencia (utilize em torno de 40% a 60% da sua banda total)

comMon { syncer { rate 100M; } }

# Nome do resource em questao (sera utilizado como referencia nos comandos
posteriores)

resource dados

    protocol C;
    handlers {
        pri-on-incon-degr "echo o > /proc/sysrq-trigger ; halt -
f";
        pri-lost-after-sb "echo o > /proc/sysrq-trigger ; halt -
```

```
f";
    local-io-error "echo o > /proc/sysrq-trigger ; halt -f";
    pri-lost "echo primary DRBD lost | mail -s 'DRBD Alert'
email@dominio.com";

    split-brain "echo split-brain. drbdadm -- --discard-my-
data connect $DRBD_RESOURCE ? | mail -s 'DRBD Alert'
pager@braslink.com";

        }

startup { degr-wfc-timeout 120; # 2 minutes.
        }

disk { on-io-error detach;
        }

net {
    sndbuf-size 512k;
    timeout 60; # 6 seconds (unit = 0.1 seconds)
    connect-int 10; # 10 seconds (unit = 1 second)
    ping-int 10; # 10 seconds (unit = 1 second)
    ping-timeout 5; # 500 ms (unit = 0.1 seconds)
    max-buffers 20480;
    cram-hmac-alg "sha1";
    shared-secret "dfadspuy234523n"; # esta chave é uma senha de
conexão
    after-sb-0pri discard-older-primary;
    after-sb-1pri violently-as0p;
    after-sb-2pri disconnect;
    rr-conflict disconnect;
}

syncer {
    rate 100M; # taxa de transferência de sincronização
    al-extents 257;
```



```

    }

on node1 {
    device /dev/drbd0; # endereço do dispositivo (disco) virtual do DRBD
    disk /dev/sda5; # partição do disco que será replicada
    address 192.168.1.10:7788; #IP do node1
    meta-disk internal; #define o meta-disk do drbd (junto com o resto do
sistema)
}

on node2 {
    device /dev/drbd0; # endereço do dispositivo (disco) virtual do DRBD
    disk /dev/sda5; # partição do disco que será replicada
    address 192.168.1.20:7788; #IP do node2
    meta-disk internal; #define o meta-disk do drbd (junto com o resto do
sistema)
}

```

Próximo passo é desmontar as partições nos dois servidores.

```
#umount /cluster
```

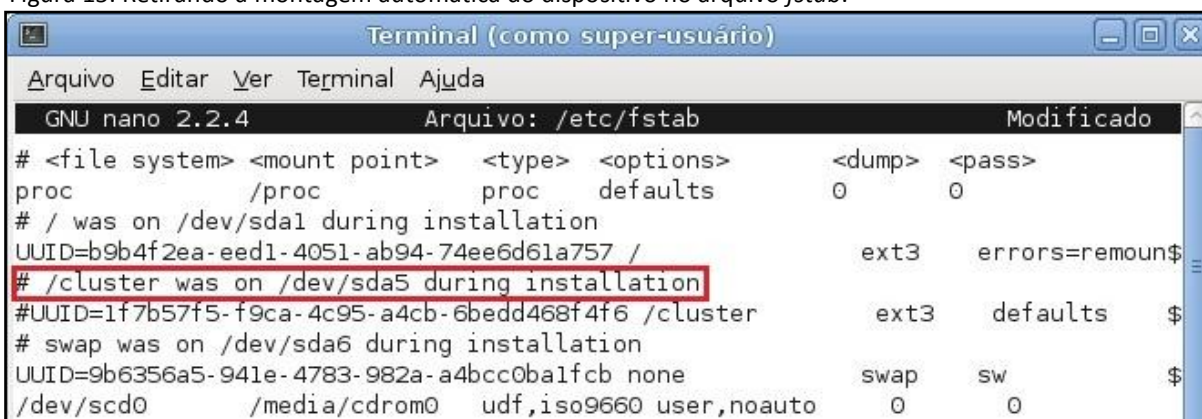
Em seguida é necessário retirar a montagem automática da partição, editando o arquivo *fstab*, usando o seguinte comando:

```
#nano /etc/fstab
```

A linha referente a montagem da partição */cluster*, que conterà os dados sincronizados pelo DRBD, deverá ser comentada no arquivo */etc/fstab*, conforme ilustrado a seguir:

```
/cluster
```

Figura 15: Retirando a montagem automática do dispositivo no arquivo *fstab*.



```

Terminal (como super-usuário)
Arquivo Editar Ver Terminal Ajuda
GNU nano 2.2.4      Arquivo: /etc/fstab      Modificado
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# / was on /dev/sda1 during installation
UUID=b9b4f2ea-eed1-4051-ab94-74ee6d61a757 / ext3 errors=remoun$
# /cluster was on /dev/sda5 during installation
#UUID=1f7b57f5-f9ca-4c95-a4cb-6bedd468f4f6 /cluster ext3 defaults $
# swap was on /dev/sda6 during installation
UUID=9b6356a5-941e-4783-982a-a4bcc0ba1fcb none swap sw $
/dev/scd0 /media/cdrom0 udf,iso9660 user,noauto 0 0

```

Fonte: Autoria própria.

Após a retirada do ponto de montagem automático, é necessário zerar a partição que será replicada nos dois servidores. Para tal utilizamos o comando a seguir:

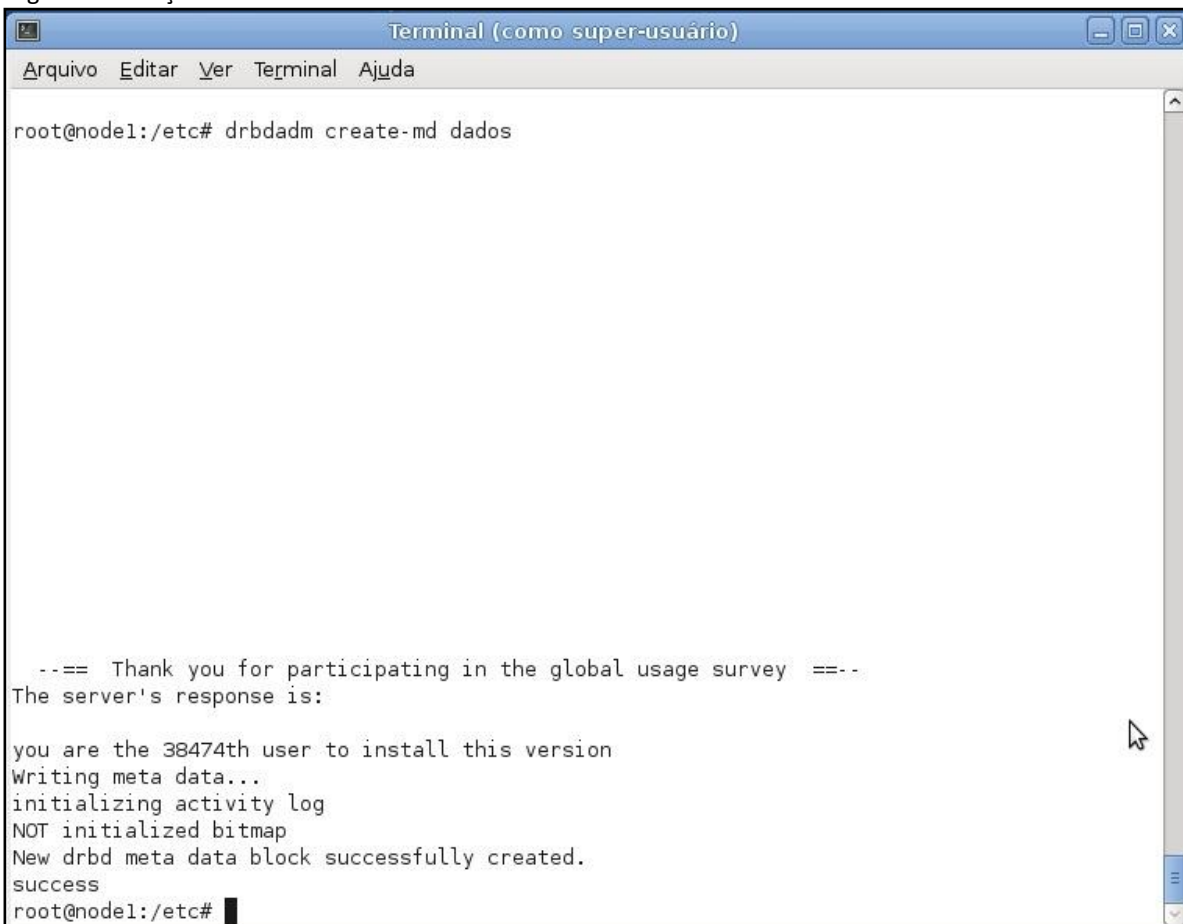
```
#dd if=/dev/zero of=/dev/sda5 bs=1M count=128
```

O parâmetro `bs=1M` indica o tamanho dos blocos a serem gravados na partição (disco) enquanto que o parâmetro `count=128` indica o número de blocos a serem gravados. Estes parâmetros devem ser ajustados de acordo com o tamanho da partição que, no exemplo, é de 128MB. Também é possível omitir o parâmetro `count=128`, o que fará que toda a partição seja escrita, independente de seu tamanho.

O comando a seguir cria o disco virtual, como demonstrado na Figura 16.

```
# drbdadm create-md dados
```

Figura 16: Criação do disco virtual.



```
Terminal (como super-usuário)
Arquivo  Editar  Ver  Terminal  Ajuda

root@node1:/etc# drbdadm create-md dados

--== Thank you for participating in the global usage survey ==--
The server's response is:

you are the 38474th user to install this version
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
success
root@node1:/etc#
```

Fonte: Autoria própria.

Ative os discos nas duas máquinas:

```
# drbdadm attach dados
```

Sincronize os discos nas duas máquinas:

```
# drbdadm syncer dados
```

As Figura 17, Figura 18 e Figura 19 demonstram a replicação dos dados entre os discos virtuais, após a realização do comando para realizar a replicação (Comando deve ser realizado somente no servidor primário) e reiniciar o serviço nas duas máquinas.

```
# drbdadm -- --overwrite-data-of-peer primary dados
```

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

Figura 17: Início da sincronização entre os dispositivos virtuais.

```

Terminal (como super-usuário)
Arquivo Editar Ver Terminal Ajuda
root@node1:/# cat /proc/drbd
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
0: cs:SyncSource ro:Secondary/Secondary ds:UpToDate/Inconsistent C r----
   ns:346368 nr:0 dw:0 dr:427808 al:0 bm:20 lo:2 pe:69 ua:2545 ap:0 ep:1 wo:b oos:3561220
   [>.....] sync'ed: 9.0% (3561220/3905380)K
   finish: 0:01:12 speed: 49,164 (49,164) K/sec
root@node1:/# █

```

Fonte: Autoria própria.

Figura 18: Sincronização em 56,1%

```

Terminal (como super-usuário)
Arquivo Editar Ver Terminal Ajuda
root@node1:/# cat /proc/drbd
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
0: cs:SyncSource ro:Secondary/Secondary ds:UpToDate/Inconsistent C r----
   ns:2189116 nr:0 dw:0 dr:2270144 al:0 bm:133 lo:1 pe:73 ua:2533 ap:0 ep:1 wo:b oos:171859
   6
   [=====>.....] sync'ed: 56.1% (1718596/3905380)K
   finish: 0:00:37 speed: 46,032 (40,496) K/sec
root@node1:/# █

```

Fonte: Autoria própria.

Figura 19: Sincronização terminada.

```

Terminal (como super-usuário)
Arquivo Editar Ver Terminal Ajuda
root@node1:/# cat /proc/drbd
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
0: cs:Connected ro:Secondary/Secondary ds:UpToDate/UpToDate C r----
   ns:3905380 nr:0 dw:0 dr:3905380 al:0 bm:239 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0
root@node1:/# █

```

Fonte: Autoria própria

O comando a seguir é utilizado para definir o servidor primário, observando que este comando deve ser realizado somente no servidor primário:

```
# drbdadm primary all
```

O comando a seguir é utilizado para definir o servidor secundário e deve ser executado somente no servidor secundário.

```
# drbdadm secondary all
```

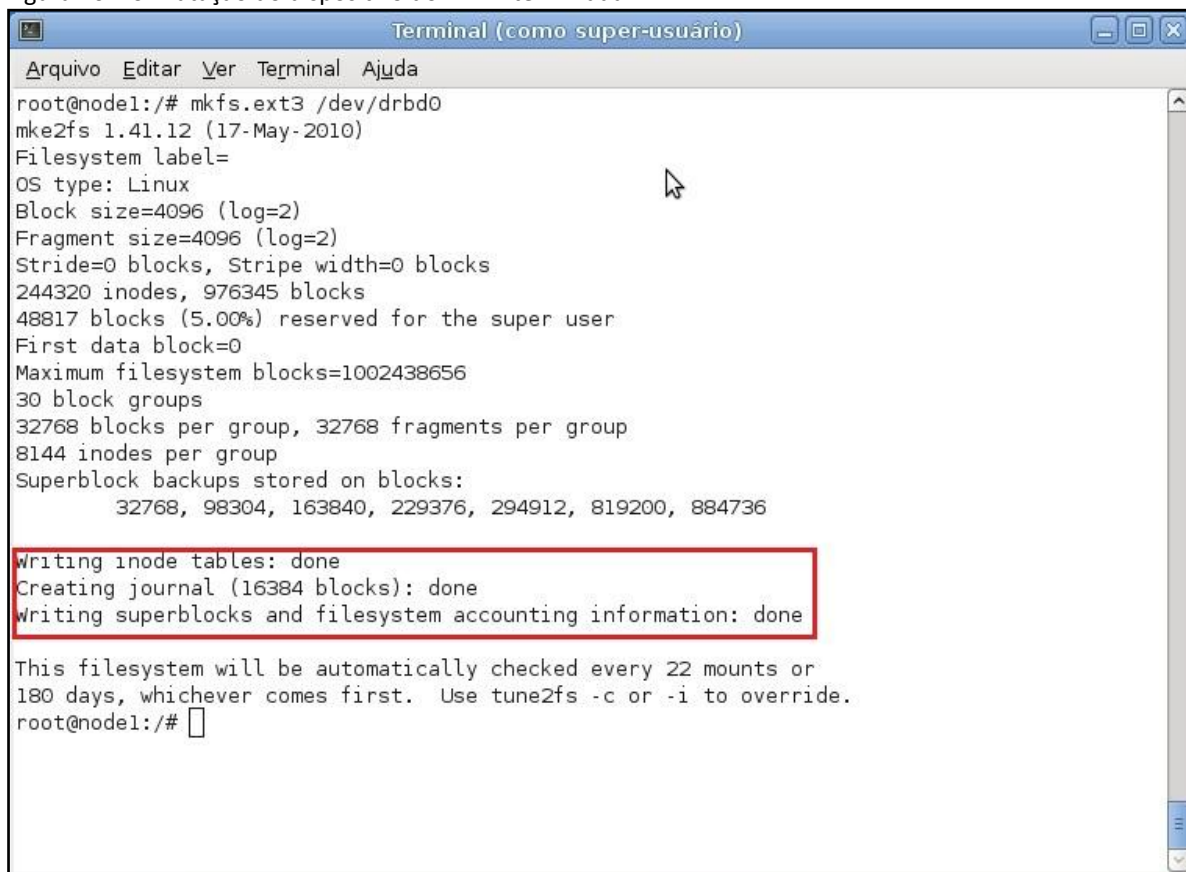
Agora é necessário formatar o dispositivo criado pelo DRBD no servidor primário, através

do comando:

```
# mkfs.ext3 /dev/drbd0
```

O resultado da execução da formatação é apresentado na Figura 20.

Figura 20: Formatação do dispositivo do DRBD terminada.



```
Terminal (como super-usuário)
Arquivo  Editar  Ver  Terminal  Ajuda
root@node1:/# mkfs.ext3 /dev/drbd0
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
244320 inodes, 976345 blocks
48817 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1002438656
30 block groups
32768 blocks per group, 32768 fragments per group
8144 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 22 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
root@node1:/#
```

Fonte: Autoria própria

Caso seja necessário alterar o servidor primário e secundário manualmente, para troca de alguma peça (*hardware*) ou qualquer outro tipo de manutenção que necessite desligar a máquina use os seguintes comandos:

```
# drbdadm secondary all
```

Para definir o servidor primário.

```
# drbdadm primary all
```

Para definir o servidor secundário, que neste caso poderá ser desligado sem afetar o

funcionamento do *cluster*.

Adicione no arquivo `/etc/fstab` dos dois servidores os parâmetros de montagem da partição a ser sincronizada, da seguinte forma:

```
/dev/drbd0 /cluster ext3 noauto 0 0
```

Assim foi finalizada a configuração do DRBD. Para testar o seu funcionamento, realizamos os seguintes testes:

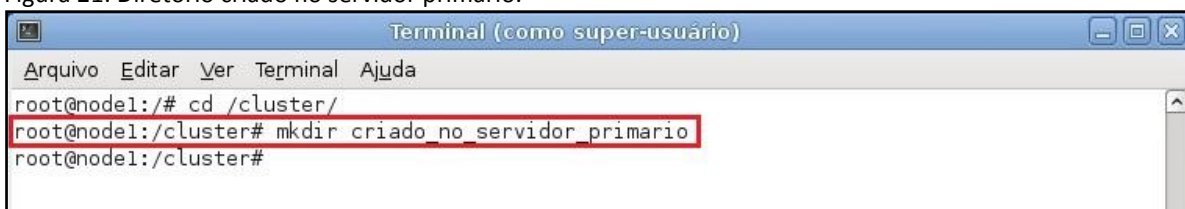
Montamos a partição no servidor primário.

```
# mount /cluster
```

Criamos um diretório vazio dentro da partição montada, como demonstrado na Figura 21.

```
# cd /cluster
# mkdir criado_no_servidor_primario
```

Figura 21: Diretório criado no servidor primário.



Fonte: Autoria própria

Verificamos se o diretório foi criado.

```
# ls /cluster
```

Desmontamos o diretório.

```
# umount /cluster
```

Realizamos o seguinte comando no `node1` para definir o até então servidor primário, para secundário:

```
# drbdadm secondary all
```

Realizamos o seguinte comando no node2 para definir o até então servidor secundário, para primário:

```
# drbdadm primary all
```

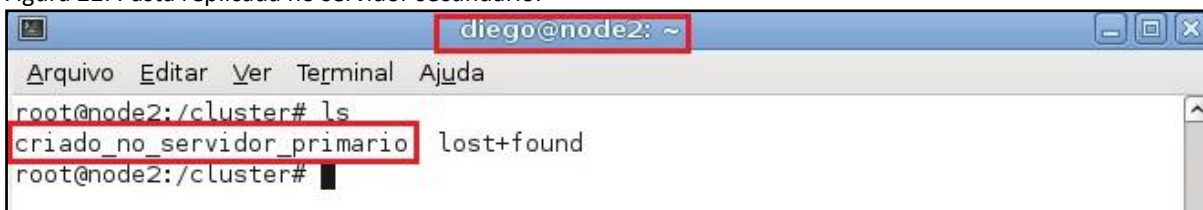
Montamos a partição.

```
# mount /cluster
```

Verificamos se o diretório criado no servidor primário (node1) foi replicado no servidor secundário (node2), como demonstrado na Figura 22.

```
# ls /cluster
```

Figura 22: Pasta replicada no servidor secundário.



Fonte: Autoria própria

## 6.5 Configurando o Mon

O comando usado para instalar o Mon na distribuição *Debian Linux* é demonstrado a seguir:

```
#apt-get install mon
```

Por padrão a instalação do Mon não cria diretório e nem o arquivo de *log*, portando é preciso criá-los manualmente. Para criar o diretório onde vamos armazenar o arquivo de *log* usamos o seguinte comando:

```
mkdir -p /var/log/mon
```

Para permitir que o *software* Mon possa registrar (gravar) dados no arquivo de *log*, é

preciso ajustar as permissões de acesso ao diretório de *log*, lembrando que o *software* Mon utiliza o usuário *mon* e o grupo *mon* durante a sua execução. Para alterar as permissões ao *log*, o seguinte comando deve ser executado:

```
chown -R mon.mon /var/log/mon
```

Através do mesmo processo criamos e fizemos as mudanças necessárias no arquivo que armazenará os logs do Mon.

```
#touch /var/log/mon/heartbeat.alert.log  
#chown mon.mon /var/log/mon/heartbeat.alert.log
```

A configuração do Mon é feita em apenas um arquivo, encontrado em */etc/mon/mon.cf* que será demonstrado a seguir:

```
alerkdir      = /usr/lib/mon/alert.d  
mondir        = /usr/lib/mon/mon.d  
maxprocs      = 20  
histlength    = 100  
historicfile  = /var/log/mon.log  
randstart     = 60s  
  
hostgroup cluster localhost  
watch cluster  
    service smb  
    interval 10s  
    monitor smb.monitor  
    allow_empty_group  
    period wd {Sun-Sat}  
    alert heartbeat.alert  
    alert file.alert -d /var/log/mon heartbeat.alert.log  
    alert mail.alert -S "Serviço Samba parou de funcionar"  
root@localhost  
    upalert mail.alert -S "Servio Samba está OK"  
root@localhost  
    alertevery 1m
```



Descrição dos parâmetros de configuração.

- *alertdir* – diretório onde são armazenados os scripts de alerta
- *mondir* – diretório onde são armazenados os scripts de monitoração
- *maxprocs* – número máximo de processos
- *histlength* - tamanho da lista do histórico
- *randstart* - tempo de inicialização
  
- *hostgroup* - define um nome de grupo de *hosts* a serem monitorados
  
- *watch* - define quais serviços serão monitorados. O nome definido no parâmetro *service* dentro do *watch* deve ser o nome do *script* monitor, sem a extensão *.monitor* e deve constar no *mondir*, definido acima. Ou seja, em */usr/lib/mon/mon.d* deve existir um script chamado *smb.monitor*
- *service* - este parâmetro é exatamente o nome do script existente sob */usr/lib/mon/mon.d*, sem a extensão *.monitor*
- *interval* - intervalo de tempo entre as checagens
- *monitor* – o script de monitoração
- *period* - período em que a checagem deve ser executada
- *alert* – aqui são definidos os alertas a serem gerados quando a programação de algum *script* de monitoração de algum serviço retornar um valor diferente de zero, ou seja, caso algum serviço monitorado pare de funcionar, automaticamente o Mon dispara os alertas citados neste parâmetro

O *Heartbeat* detecta falhas entre o servidor primário e o servidor secundário e faz a troca de papéis somente se houver uma falha de comunicação entre os mesmos, porém ele não consegue identificar falhas dos serviços usados na alta disponibilidade, que no nosso caso seria o Samba. Essa monitoração é feita através do Mon, que programado através de *scripts* de monitoração consegue monitorar os serviços e, em caso de falha, automaticamente desativa o *Heartbeat* no servidor primário, forçando assim que o servidor secundário assuma o papel de primário.

Por padrão o Mon não possui um *script* de monitoração do Samba, tornando necessária a sua criação e gravação no diretório */usr/lib/mon/mon.d* que é o diretório padrão do Mon para *scripts* de monitoração. O exemplo a seguir ilustra um *script* de monitoração do processo Samba, que deve ser gravado em */usr/lib/mon/mon.d/*:

```
#!/bin/bash

DRBDADM="/sbin/drbdadm"
GREP="/bin/grep"
PS="/bin/ps"

$PS aux | $GREP smbd | $GREP -v $GREP >> /dev/null
if [ $? = 0 ];then
    echo "ok"
else
    echo "samba ESTA FORA DO AR, VERIFIQUE!"
    exit 1
fi
```

Depois de criado é necessário atribuir permissão de execução para o *script*.

```
#chmod +x /usr/lib/mon/mon.d/smb.monitor
```

É importante salientar que, caso o Mon necessite encerrar o funcionamento do *Heartbeat*, devido alguma falha de serviço, o próprio Mon não poderá ativa-lo novamente, e esse trabalho fica a cargo do administrador da rede. Porém para auxiliar o administrador, se configurado, o Mon consegue enviar mensagens avisando que o serviço monitorado parou de funcionar, através do email especificado no arquivo de configuração `/etc/mon/mon.cf`. Como demonstrado na Figura 23.

Figura 23: Trecho do arquivo de configuração do Mon, onde indica o email onde será enviado alertas

```
alert mail.alert -S "Serviço Samba parou de funcionar" root@localhost
upalert mail.alert -S "Serviço Samba está OK" root@localhost
```

Fonte: Autoria própria

Note que, para que o envio de *e-mails* funcione, é preciso um serviço de *e-mails* disponível na rede.

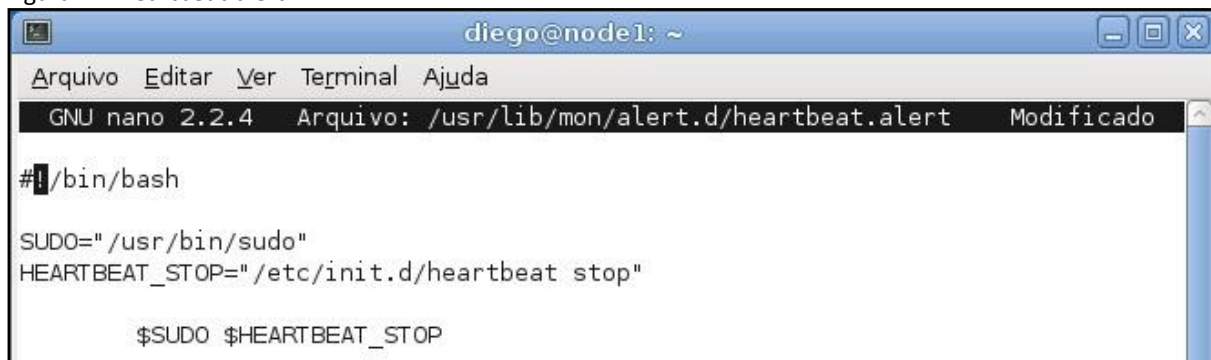
Os *scripts* de alertas do Mon ficam no diretório `/usr/lib/mon/alert.d`. Por padrão já existem alguns alertas criados, porém o *heartbeat.alert* que terá o trabalho de desligar o

*Heartbeat* deve ser criado manualmente. Arquivo demonstrado na Figura 24, para criação e atribui permissão de execução do arquivo usa-se o comando a seguir:

```
#touch /usr/lib/mon/alert.d/heartbeat.alert
```

```
#chmod +x /usr/lib/mon/alert.d/heartbeat.alert
```

Figura 24: *Heartbeat.alert*



```
diego@node1: ~
Arquivo  Editar  Ver  Terminal  Ajuda
GNU nano 2.2.4 Arquivo: /usr/lib/mon/alert.d/heartbeat.alert Modificado
#!/bin/bash
SUDO="/usr/bin/sudo"
HEARTBEAT_STOP="/etc/init.d/heartbeat stop"
$SUDO $HEARTBEAT_STOP
```

Fonte: Autoria própria

O próximo passo será fazer com que o Mon tenha permissão de desativar o *Heartbeat* sem a necessidade de senha administrativa, para tal será necessário editar o arquivo */etc/sudoers*, através do comando:

```
#nano /etc/sudoers
```

Neste arquivo adicione a linha demonstrada na Figura 25. Isto permitirá que o usuário *mon* use o comando */etc/init.d/heartbeat stop* sem a necessidade de ser *root* ou ter senha do mesmo.

Após a finalização, o serviço *Mon* deve ser reiniciado com o comando:

```
#!/etc/init.d/mon restart
```

O comando *monshow* mostra o estado dos serviços monitorados pelo *Mon*, conforme ilustrado na Figura 26.

```
#monshow
```

Figura 25: Arquivo sudoers

```

Terminal (como super-usuário)
Arquivo  Editar  Ver  Terminal  Ajuda
GNU nano 2.2.4      Arquivo: /etc/sudoers

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
mon     ALL=(root) NOPASSWD: /etc/init.d/heartbeat stop

# Allow members of group sudo to execute any command
# (Note that later entries override this, so you might need to move
# it further down)
%sudo  ALL=(ALL) ALL
#
#includedir /etc/sudoers.d

^G Ajuda      ^O Gravar    ^R Ler o Arq ^Y Pág Anter ^K Recort Txt ^C Pos Atual
^X Sair      ^J Justificar ^W Onde está? ^V Próx Pág  ^U Colar Txt ^T Para Spell

```

Fonte: Autoria própria

Figura 26: Saída do comando monshow.

```

Terminal (como super-usuário)
Arquivo  Editar  Ver  Terminal  Ajuda
root@node1:/# monshow

server: localhost
time: Sat Nov 3 16:31:06 2012 1
state: scheduler running

GROUP  SERVICE  STATUS  LAST  NEXT  ALERTS  SUMMARY
cluster  smb      -       5s    4s    none    ok
2
Nothing is disabled.
root@node1:/#

```

Fonte: Autoria própria

Para melhor entendimento da Figura 26 os elementos são descritos a seguir:

- 1) Descrição do servidor, hora e data da realização do teste;
- 2) Grupo de trabalho ou domínio;

- 3) Nome do serviço que está sendo monitorado (não aparece o nome do serviço propriamente dito e sim o mesmo nome definido no *script* monitor, no caso smb.monitor);
- 4) Status do serviço na ocasião do teste: o hífen indica que nenhuma falha foi detectada;
- 5) Tempo que foi realizada a última monitoração do serviço;
- 6) Tempo restante até a próxima verificação do serviço;
- 7) Total de alertas enviados (no caso até o momento nenhum);
- 8) Descrição configurada na programação do *script* monitor;

Na Figura 27 é apresentado um exemplo de saída do comando *monshow* na ocorrência de uma falha no serviço monitorado.

Figura 27: Saída do comando *monshow* em caso de falha do serviço.



```

Terminal (como super-usuário)
Arquivo Editar Ver Terminal Ajuda
root@node1:/# monshow

server: localhost
time: Sat Nov 3 16:50:55 2012
state: scheduler running

GROUP          SERVICE    STATUS    LAST    NEXT    ALERTS SUMMARY
R cluster      smb        FAIL      4s      5s      1      samba ESTA
FORA DO ...

Nothing is disabled.

root@node1:/# █

```

Fonte: Autoria própria

## 6.6 Configurando o Samba

O Samba é configurado no arquivo `/etc/samba/smb.conf`. O serviço Samba é instalado através do comando a seguir:

```
#apt-get install samba
```

A Figura 28 apresenta a configuração de um servidor Samba para fins de compartilhamento dos arquivos no diretório /cluster.

```
#nano /etc/samba/smb.conf
```

Figura 28: Arquivo de configuração do Samba.

```
diego@node2: ~
Arquivo  Editar  Ver  Terminal  Ajuda
GNU nano 2.2.4  Arquivo: /etc/samba/smb.conf

[global]
workgroup = cluster
netbios name = node1
server string = Servidor de Arquivos
security = user
interfaces = 192.168.1.15

[cluster]
path = /cluster/
read only = No
create mask = 0777
force create mode = 0777
directory mask = 0777
force directory mode = 0777
guest ok = yes

[ 16 linhas lidas ]
^G Ajuda      ^O Gravar     ^R Ler o Arq  ^Y Pág Anter  ^K Recort Txt ^C Pos Atual
^X Sair       ^J Justificar ^W Onde está? ^V Próx Pág   ^L Colar Txt  ^T Para Spell
```

Fonte: Autoria própria

Os parâmetros definidos no arquivo smb.conf são:

- *workgroup* - nome do domínio ou grupo de trabalho
- *netbios name* – nome do servidor
- *server string* – descrição do servidor
- *security* – nível de segurança
- *interfaces* – IP do Samba
- *path* – caminho do diretório a ser compartilhado
- *read only* – usado para permitir ou não gravações no diretório
- *create mask* - modo padrão para criação de arquivos no compartilhamento
- *force create mode* - força a criação de arquivos nos compartilhamentos com a permissão definida
- *directory mask* - modo padrão para a criação de diretórios no compartilhamento
- *force directory mode* - Solicita que o samba force o tipo de permissão para a criação dos diretórios no compartilhamento
- *guest ok* – indica se será ou não permitido o acesso de outros usuários

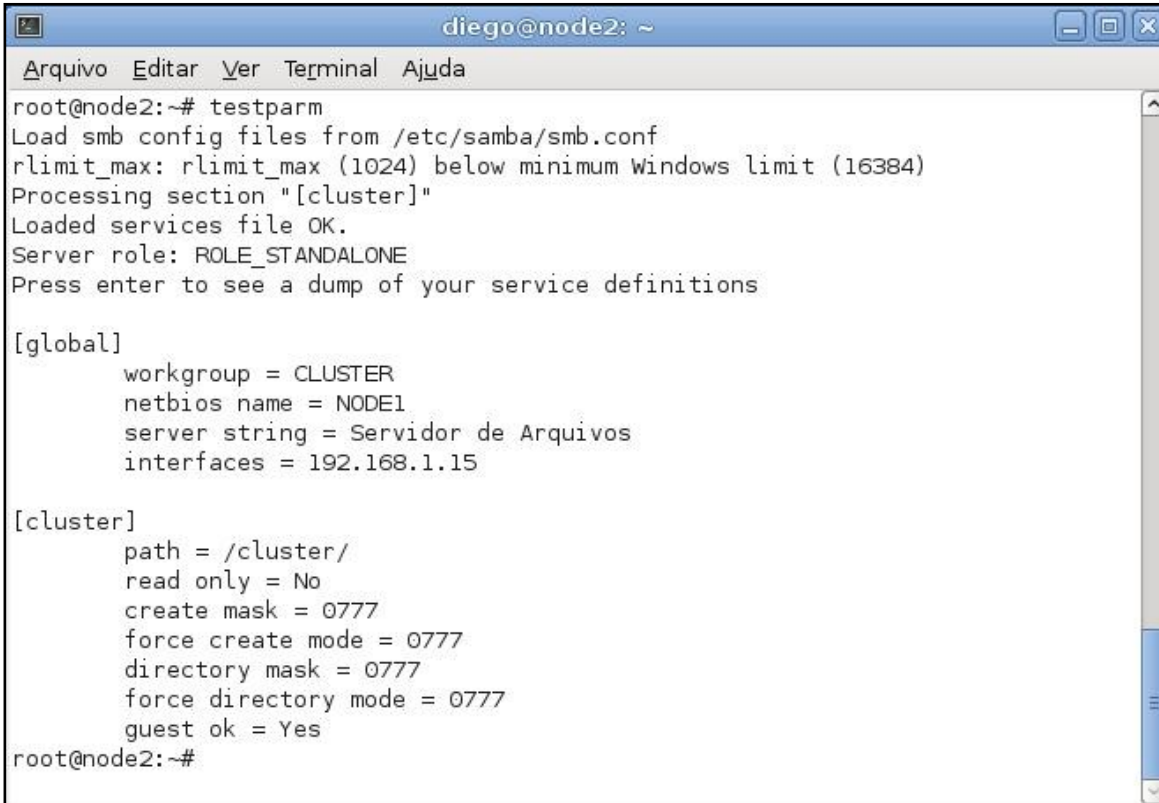
Após finalizar a instalação é necessário reiniciar o serviço através do seguinte comando:

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

Depois de reiniciar o serviço, e testado com sucesso através do comando *testparm* (usado para testar a configuração do Samba) como demonstrado na Figura 29, o próximo passo é cadastrar os *logins* e senhas das pessoas que terão acesso ao servidor de arquivos Samba.

```
#testparm
```

Figura 29: Testando arquivo de configuração do samba com o comando testparm.



```
diego@node2: ~
Arquivo Editar Ver Terminal Ajuda
root@node2:~# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: rlimit_max (1024) below minimum Windows limit (16384)
Processing section "[cluster]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

[global]
  workgroup = CLUSTER
  netbios name = NODE1
  server string = Servidor de Arquivos
  interfaces = 192.168.1.15

[cluster]
  path = /cluster/
  read only = No
  create mask = 0777
  force create mode = 0777
  directory mask = 0777
  force directory mode = 0777
  guest ok = Yes
root@node2:~#
```

Fonte: Autoria própria

Para adicionar usuários no Samba primeiramente é necessário criar o usuário no sistema *Linux*, caso o mesmo já não exista. Para tal usa-se o comando a seguir.

```
#adduser maria
```

Após criado usuário com sucesso, adiciona-se o usuário no Samba com o comando:

```
#smbpasswd -a maria
```

Caso os *logins* usados pelos usuários no *Windows* sejam os mesmos que os do *Linux*, o acesso ao compartilhamento será automático, do contrário será necessário fornecer as credenciais do Samba ao tentar acessar o compartilhamento, conforme ilustrado na Figura 30.

Figura 30: Tela de login.



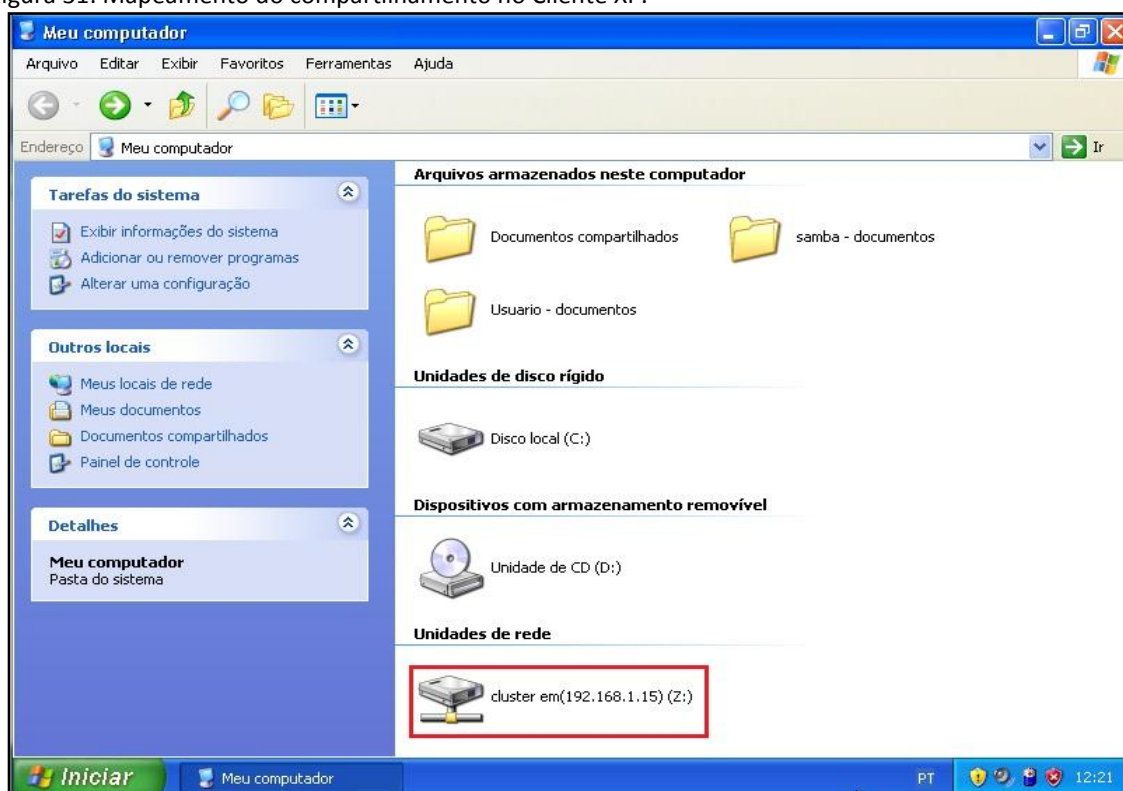
Fonte: Autoria própria



## 7 TESTANDO A FUNCIONALIDADE DO CLUSTER DE ALTA DISPONIBILIDADE

Para testar a solução, foi criado um mapeamento do diretório compartilhado pelo servidor Samba de alta disponibilidade no cliente XP, através do IP do *cluster* (192.168.1.15), conforme ilustrado na Figura 31.

Figura 31: Mapeamento do compartilhamento no Cliente XP.



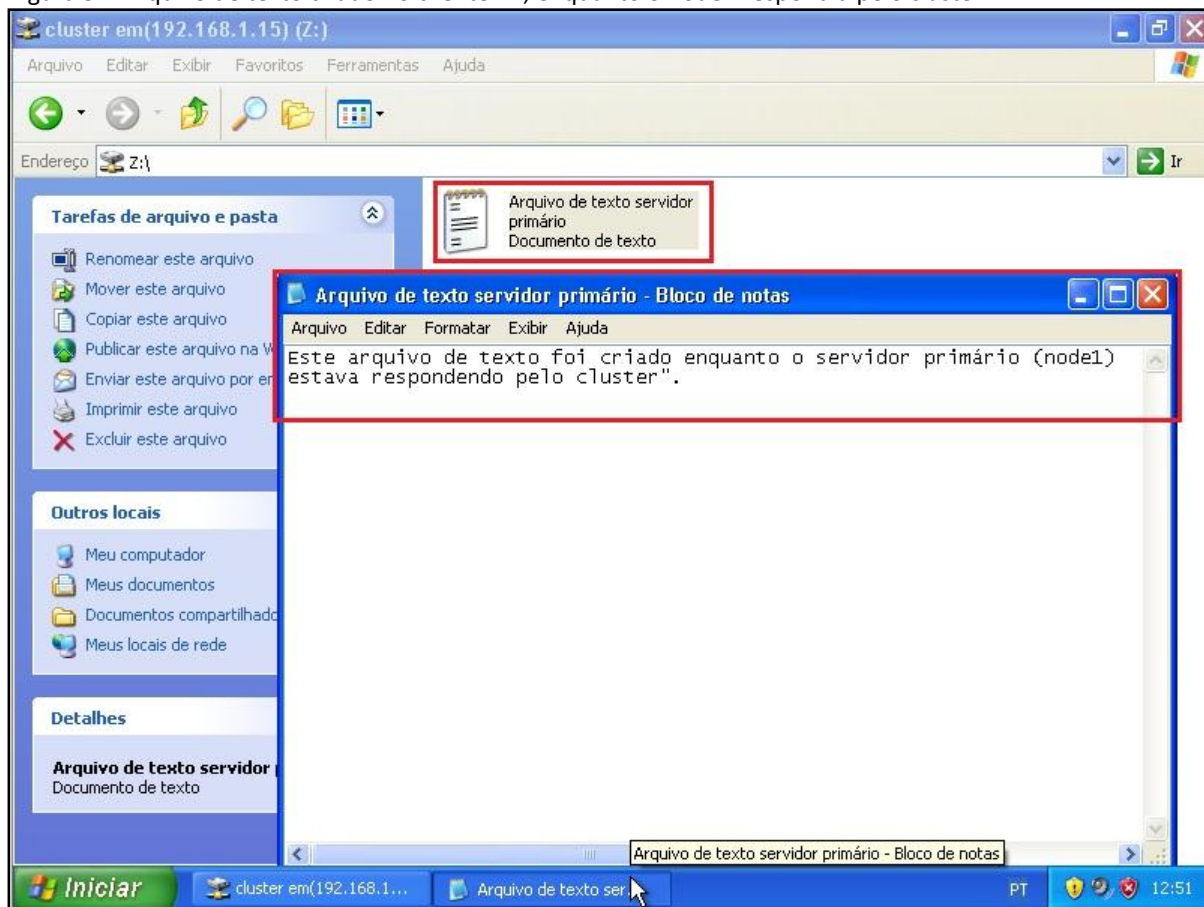
Fonte: Autoria própria

Depois de mapeado, foi acessado o compartilhamento e criou-se um arquivo de texto com o nome “Arquivo de texto servidor primário”, e dentro do mesmo arquivo foi gravada a seguinte frase “Este arquivo de texto foi criado enquanto o servidor primário (node1) estava respondendo pelo *cluster*”, conforme ilustrado na Figura 32.

A fim de comprovar que era o servidor primário (node1) que estava respondendo pelo *cluster* no momento da criação do arquivo, estabeleceu-se uma conexão *secure shell* (SSH) para o endereço IP do *cluster*, conforme ilustrado na Figura 33. Observe que, ao ser acessado o IP do *cluster*, o servidor node1 foi acessado, indicando que este servidor era o

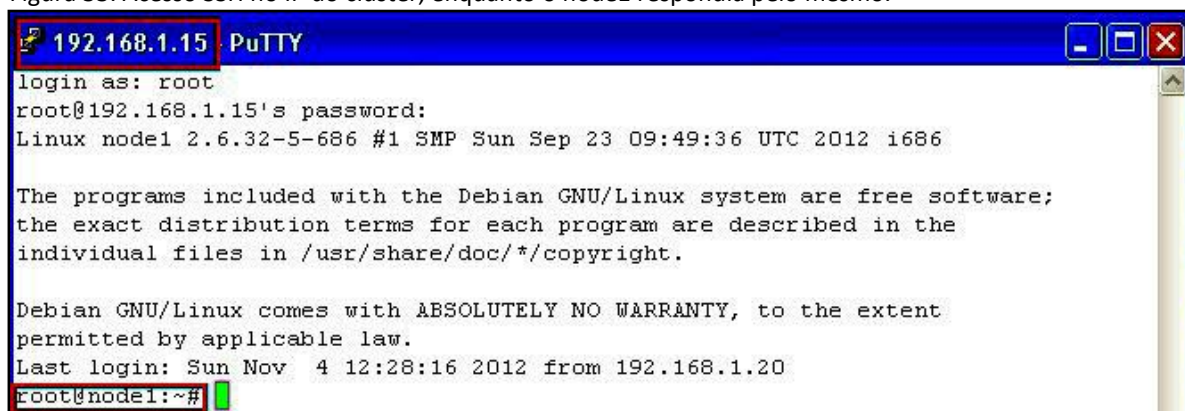
servidor primário do *cluster*, indicado no *prompt* do acesso SSH.

Figura 32: Arquivo de texto criado no cliente XP, enquanto o node1 respondia pelo *cluster*.



Fonte: Autoria própria

Figura 33: Acesso SSH no IP do cluster, enquanto o node1 respondia pelo mesmo.



Fonte: Autoria própria

Para verificar se o *failover* ocorreria com sucesso, o serviço do *Heartbeat* foi encerrado no servidor primário (node1).

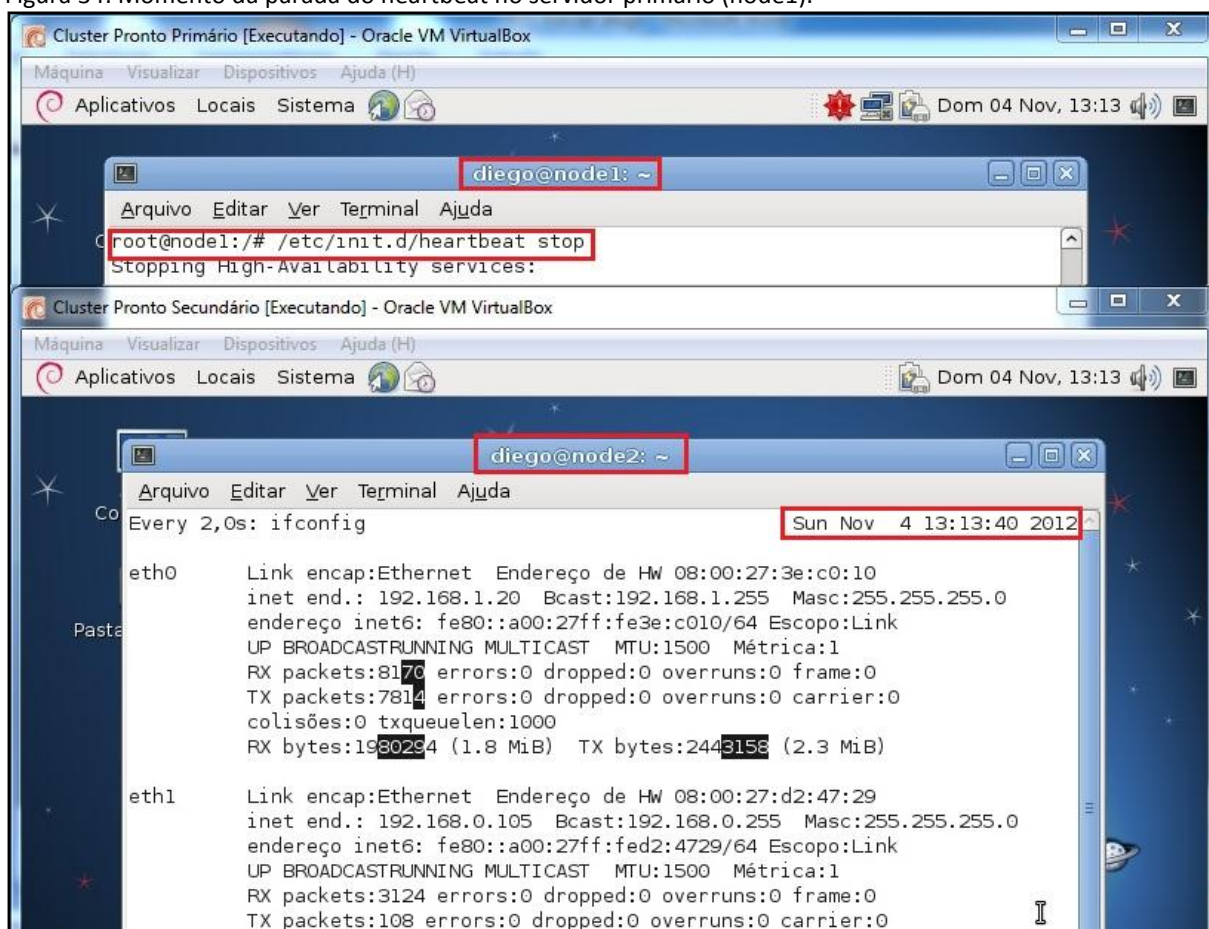
```
#/etc/init.d/heartbeat stop
```

A fim de conferir o tempo que levaria para que o servidor secundário (node2) assumisse a responsabilidade de responder pelo cluster, transformando-se assim em servidor primário, foi executado no mesmo o comando *watch* com o parâmetro *-d*, de forma a conferir a saída do comando *ifconfig* a cada 2 segundos:

```
#watch -d ifconfig
```

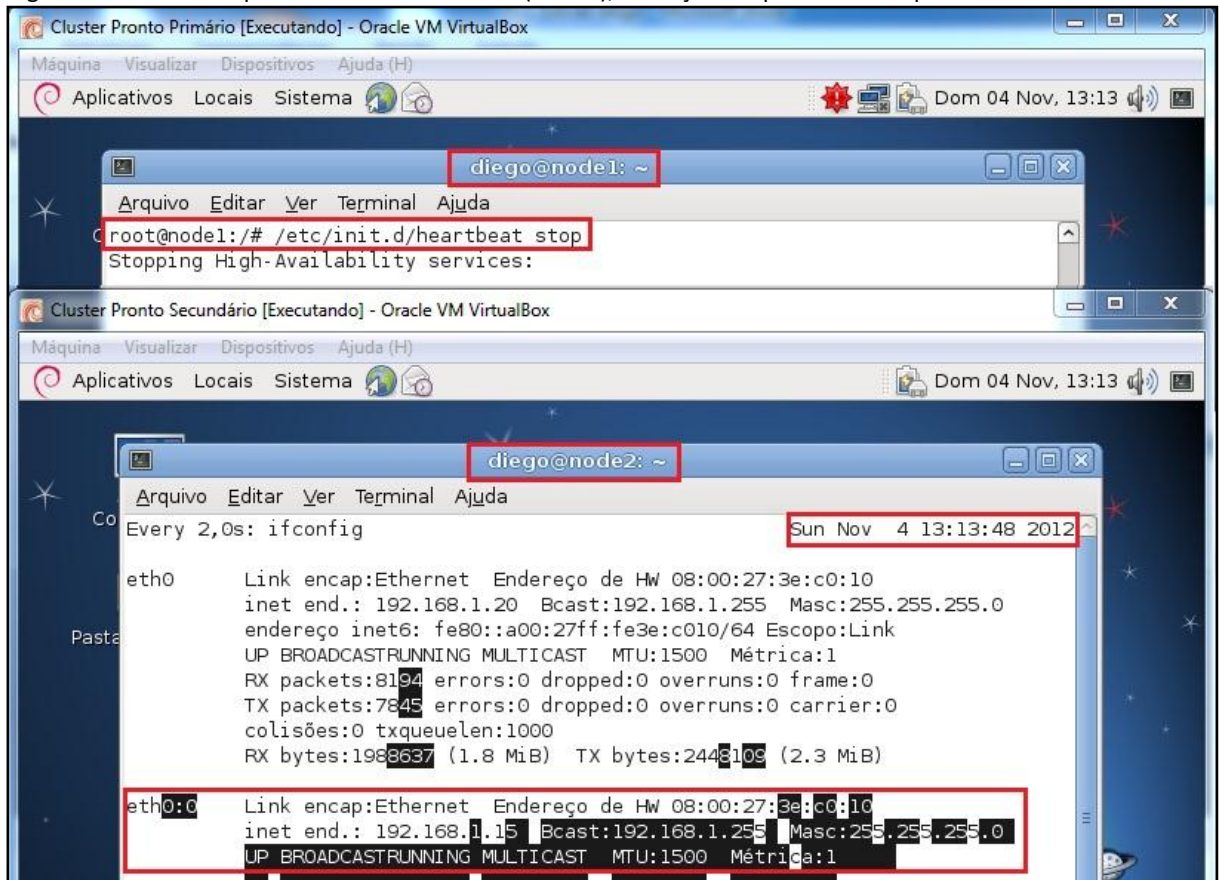
A Figura 34 e a Figura 35 ilustram, respectivamente, a finalização do *Heartbeat* no servidor node1, e a ativação do IP virtual do *cluster* no servidor node2. Observe que, no teste, o servidor node2 assumiu o IP virtual após 8 segundos. O tempo pode ser observado no relógio apresentado no canto superior da saída do comando *watch*. Observe que os dois servidores possuem relógios sincronizados.

Figura 34: Momento da parada do heartbeat no servidor primário (node1).



Fonte: Autoria própria

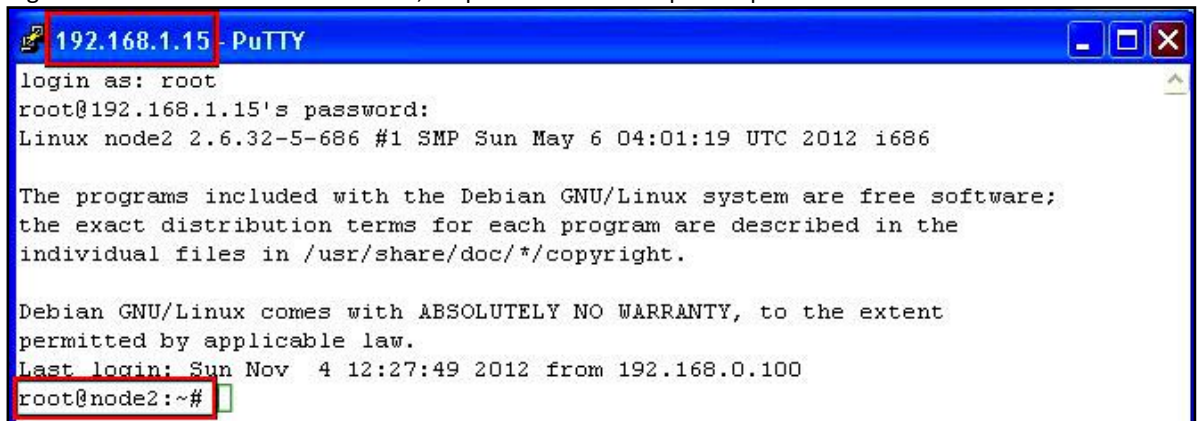
Figura 35: Momento que o servidor secundário (node2), começa a responder como primário.



Fonte: Autoria própria

A fim de comprovar que de fato o node2 começou a responder como servidor primário, um novo acesso SSH foi executado no IP do *cluster* (192.168.1.15). Como demonstrado na Figura 36, observe que o acesso foi obtido com sucesso e quem está respondendo é realmente o node2.

Figura 36: Acesso SSH no IP do cluster, enquanto o node2 respondia pelo mesmo.

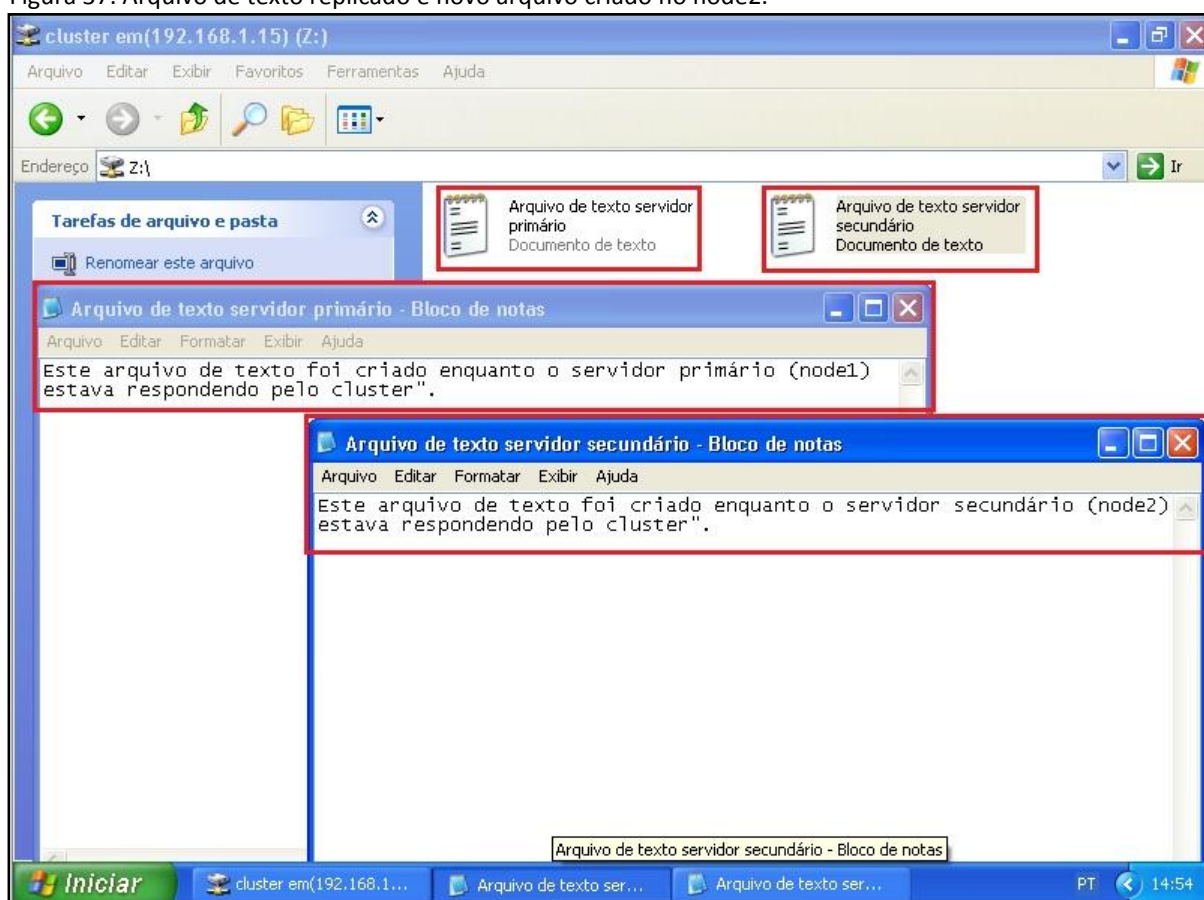


Fonte: Autoria própria



Com o intuito de verificar se houve a replicação de arquivos para o servidor secundário (node2), foi executado um novo acesso ao servidor Samba através do mapeamento criado no cliente *Windows XP*, e constatou-se que o arquivo criado no node1 estava disponível no node2, que neste momento respondia pelo *cluster*. Foi criado então outro arquivo de texto, agora com o nome de “Arquivo de texto servidor secundário”, com o seguinte conteúdo “Este arquivo de texto foi criado enquanto o servidor “secundário” (node2) estava respondendo pelo *cluster*”. Todo este processo pode ser observado na Figura 37.

Figura 37: Arquivo de texto replicado e novo arquivo criado no node2.



Fonte: Autoria própria

O *Heartbeat* foi configurado com a opção *auto\_failback* ativada, que tem como objetivo devolver o papel de servidor primário para o node1, caso o mesmo volte a responder, o serviço do *Heartbeat* foi reativado no node1 a fim de testar o *failback*.

```
#/etc/init.d/heartbeat start
```

Uma nova conexão SSH no IP do *cluster* foi efetuada e constatou-se que o node1

voltou a responder como servidor primário, comprovando assim o funcionamento do *failback*.

Para novamente testar se a replicação de dados foi realizada com sucesso o mapeamento do cliente *Windows XP* foi acessado para conferir a existência do arquivo de texto que foi criado enquanto o *node2* respondia como servidor primário. Novamente os arquivos estavam da mesma maneira que se encontravam quando houve a transição entre os servidores.

Continuando, para testar o monitoramento do serviço Samba pelo Mon, o serviço Samba foi encerrado no *node1* com o comando:

```
# /etc/init.d/samba stop
```

Este procedimento simula a ocorrência de algum problema de inicialização ou interrupção do serviço, constatando que foi enviado um alerta (*heartbeat.alert*) parando assim o *Heartbeat* no *node1*, ocasionando a ativação dos serviços no *node2* em aproximadamente 8 segundos.

Para finalizar os testes e verificar se a solução estava tolerante a falhas de conectividade, o cabo de rede que ligava o *node1* a rede local foi desconectado, por onde os clientes acessavam os serviços disponibilizados. Após aproximadamente 8 segundos, ocorreu um *failover*, pois o *Heartbeat* detectou uma falha de conexão entre os servidores provocando assim que o *node2* assumisse o papel de servidor primário, comprovando assim um ambiente altamente disponível.

## 8 CONSIDERAÇÕES

Empresas, sejam elas de pequeno ou grande porte, dispõem de um grande número de informações, das quais são de vital importância para o crescimento e continuidade dos negócios. Devido a este fator, quanto menor o tempo elas se mantiverem inacessíveis melhor serão aproveitadas.

A alta disponibilidade tem como maior objetivo, se não eliminar por completo, diminuir consideravelmente grande parte dos contratempos provocados por falhas ou algum tipo de manutenção planejada. Entretanto ao se pensar em tal arquitetura num primeiro momento vem à mente altíssimos investimentos, fator que poderia limitar ou até mesmo inviabilizar o projeto.

Com base neste princípio, o presente trabalho comprova que é possível implementar um sistema altamente disponível somente com uso de ferramentas de código aberto e praticamente sem custos.

Para comprovar a eficiência do *cluster* de alta disponibilidade abordamos o uso do mesmo criando um ambiente onde o intuito era deixar as informações disponíveis pelo máximo tempo possível, simulando testes através de um servidor de arquivos Samba, provocando vários tipos de situações e falhas das quais em todas se obteve sucesso garantindo um serviço ininterrupto.

Através do uso dos três *softwares* abordados, bem como os resultados obtidos com os testes realizados, foi possível resolver o problema de disponibilidade através de replicação e monitoramento dos servidores. E o mais importante, de maneira barata, fator este que torna essas ferramentas muito atrativas, justamente pelos recursos que as mesmas proporcionam. Demonstrando que falhas são inevitáveis, porém podem ser minimizadas.

Demonstramos então que, se as falhas são inevitáveis, as suas consequências para a disponibilidade podem ser minimizadas, senão eliminadas.

## REFERÊNCIAS

AGUIAR, G. M. **O que é alta disponibilidade?**. 2012. Disponível em:

<<http://technet.microsoft.com/pt-br/sqlserver/hh226646.aspx>>. Acesso em: 18 ago. 2012.

AVIZIENIS, A. et al. **Basic Concepts and Taxonomy of Dependable and Secure Computing**.

2004. Disponível em: <<http://www.ge.tt/#!/7LaXQYE/v/0>>. Acesso em 19 ago. 2012.

BRANDÃO, R. **Conceitos sobre disponibilidade**. 2011. Disponível em:

<<http://technet.microsoft.com/pt-br/library/cc668492.aspx>>. Acesso em: 18 ago. 2012.

CHRISTIANINI, T. **Minha empresa necessita realmente de alta disponibilidade (HA)?**. 2011.

Disponível em: <<http://www.dualtec.com.br/blog/2011/05/24/minha-empresa-necessita-realmente-de-alta-disponibilidade-ha/>>. Acesso em 16 out. 2012.

DRBD. **What is DRBD**. 2008. Disponível em: <<http://www.drbd.org/>>. Acesso em 12 out. 2012.

DRBD. **Replication modes**. 2008a. Disponível em: <<http://www.drbd.org/users-guide-emb/s-replicationprotocols.html>>. Acesso em 12 out. 2012.

DRBD. **DRBD Fundamentals**. 2008b. Disponível em: <<http://www.drbd.org/users-guide-emb/chfundamentals.html>>. Acesso em 12 out. 2012.

DRBD. **User space administration tools**. 2008c. Disponível em: <<http://www.drbd.org/users-guide-emb/suserland.html>>. Acesso em 12 out. 2012.

KOPPER, K. **The Linux enterprise cluster**. San Francisco: No Starch Press, 2005. 468p.

LINUX-HA. **HA-HighAvailability**. 2011. Disponível em: <<http://linux-ha.org/>>. Acesso em 12 out. 2012.



MORIMOTO, C. E. **Hardware II, o Guia Definitivo**. Porto Alegre: Sul Editores , 2007. 1088p.

MORIMOTO, C. E. **Servidores em cluster e balanceamento de carga**. 2007. Disponível em: <<http://www.hardware.com.br/artigos/cluster-carga/>>. Acesso em 12 out. 2012.

MORIMOTO, C. E. **Servidores Linux: Guia Prático**. 2. ed. Porto Alegre: Sul Editores, 2010. 719p.

PEREIRA, N. A. **Serviços de pertinência para clusters de alta disponibilidade**. 2004. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04102004-104700/publico/dissertacao.pdf>>. Acesso em 24 ago. 2012.

PFISTER, G. F. **In Search of Clusters**. 2. ed. New Jersey: Prentice Hall, 1997. 608p.

PITANGA, M. **Computação em Cluster**. Rio de Janeiro: Brasport, 2003. 344p.

PITANGA, M. **Construindo Super Computadores com Linux**. 3. ed. Rio de Janeiro: Brasport, 2008. 400p.

RESNICK, R. I. **A Modern Taxonomy of High Availability**. 1996. Disponível em: <<http://www.generalconcepts.com/resources/reliability/resnick/HA.htm>>. Acesso em 25 ago. 2012.

RUIZ, André. **Alta disponibilidade em servidores Linux**. 2000. Disponível em: <<http://augustocampos.net/revista-do-linux/006/alta.html>>. Acesso em: 11 ago. 2012.

TANEMBAUM, A. S. **Organização Estruturada de Computadores** 5. ed. São Paulo: Pearson Education do Brasil, 2007. 457p.

TROCKI, J. **Mon**. 2008. Disponível em: <<http://linux.die.net/man/8/mon>>. Acesso em 24 ago. 2012.

WEBER, T. S. **Tolerância a falhas: conceitos e exemplos**. 2002. Disponível em: <<http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF>>. Acesso

em 24 ago. 2012.

WEBER, T. S. **Um roteiro para exploração dos conceitos básicos de tolerância a falhas.**

2002.

Disponível

em:

<<http://www.inf.ufrgs.br/~taisy/disciplinas/textos/Dependabilidade.pdf>>. Acesso em 24 ago.

2012.

## **ANEXOS**

### **A. Vídeo de demonstração do funcionamento da alta disponibilidade**