Roberto Benedito de Oliveira Pereira

Alta Disponibilidade em Sistemas GNU/LINUX utilizando as ferramentas Drbd, Heartbeat e Mon

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Pós-Graduação "*Lato Sensu*" Administração em Redes Linux para obtenção do título de Especialista em "Administracao em Redes Linux".

Orientador
Prof. DSc. Ricardo Martins De Abreu Silva

Lavras
Minas Gerais - Brasil
2005

Roberto Benedito de Oliveira Pereira

Alta Disponibilidade em Sistemas GNU/LINUX utilizando as ferramentas Drbd, Heartbeat e Mon

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Pós-Graduação "*Lato Sensu*" Administração em Redes Linux para obtenção do título de Especialista em "Administracao em Redes Linux".

Aprovada em *12 de setembro de 2005*

Prof. Esp. Rêmulo Maia Alves

Prof. Esp. Samuel Pereira Dias

Prof. Esp. Anderson B. dos Santos

Prof. Esp. Andre Zambalde

Prof. DSc. Ricardo Martins De Abreu Silva
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Agradeço a Deus, a São Benedito, à Santa Rita e aos meus pais, que estão sempre presentes nos momentos mais difíceis da minha vida, apoiando e respeitando as minhas decisões, agradeço também aos meus amigos, ao meu orientador e a todos aqueles que me ajudam e que me ajudaram.

Resumo

Sistemas de alta disponibilidade são sistemas que tem como principal finalidade tornar os serviços prestados por um servidor disponível o maior tempo possível ao usuário de forma que, após a falha de um servidor primário, um servidor secundário supra a falha do mesmo, proporcionando com isso maior qualidade nos serviços prestado pelo mesmo. Para isso, são utilizadas tanto técnicas de *hardware* como de *software*, onde nesta monografia é focada mais a parte de *software*, que dentre eles, destacam-se o Drbd, Heartbeat e o Mon. Nesta monografia será apresentado a teórica da alta disponibilidade, bem como os resultados obtidos na implementação de um estudo de caso de um sistema de alta disponibilidade.

Dedico está monografia a todos os usuários de LINUX, à Universidade Federal de Lavras e em especial a todos os membros da ARL.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O objetivo principal desta monografia é apresentar a solução de Alta Disponibilidade em um Sistema Operacional GNU/Linux elucidando as principais características do funcionamento de um Cluster de Alta Disponibilidade, buscando com isso divulgar, por meios de um relatório escrito, um conjunto de poderosas ferramentas de baixo custo que unidas visam alcançar esse objetivo, objetivo esse que ainda não é muito explorado pelos administradores de sistemas.

Um sistema de Alta Disponibilidade tem como principal característica à de suprir a falha de um servidor em um sistema *on-line* de forma automática para que os serviços prestado pelo mesmo fique o maior tempo possível disponível ao usuário e o mais importante é que nesta monografia, isso foi feito de uma forma que não foi necessário gastar muito dinheiro para isso.

Desde o surgimento do sistema operacional Linux, lembrando que ele foi um dos precursores do movimento do software livre, tendo também como principal característica ser um dos grandes personagens da concretização dessa ideologia, onde que até pouco tempo atrás, a participação do Linux em servidores de missão critica se restringia bastante, pois ainda não se tinha uma estrutura muito bem estabelecida e ele ainda não era robusto o suficiente para isso.

Porem essa situação se alterou, quando foram desenvolvidas soluções de baixo custo que suprirão a necessidade do Linux preparando-o para tal. Grupos de desenvolvedores deram inicio a vários projetos *open-source* para pesquisarem tais soluções, como resultado disso, surgiram vários projetos, dentre eles o Drbd, o Heartbeat e o Mon, que tiveram a capacidade de solucionar problemas de disponibilidade de serviço, replicação e monitoramento de serviços.

Basicamente o termo alta disponibilidade consiste em tentar manter um serviço provido por um ou mais servidores o máximo de tempo possível disponível, onde que na falta de um dos servidores, seja ela por queima de um diso, pela queima

de uma placa de rede ou qualquer coisa do tipo, seja suprido automaticamente por outro servidor, de tal forma que não interrompa a disponibilidade do serviço.

Várias técnicas são utilizadas para isso, primeiramente é necessário ter em mente, que a alta disponibilidade não consiste somente a *software* instalados, mas também de uma estrutura de *hardware*. Quando é dito alta disponibilidade deve ser levado em consideração também a redundância tanto no que diz respeito a *hardware*, como por exemplo: *hubs*, *switchs*, *roteadores*, fonte de alimentação, quanto no que diz respeito a *software*, como por exemplo: programa de replicação, programas de decisões e programas de monitoramento.

No que diz respeito a parte de *software* está monografia aborda os seguintes programas: o Drbd, o Heartbeat e o Mon. O Drbd tem como principal finalidade a de estar replicando as informações de um servidor em outro, utilizando como meio de transmissão a rede. O Heartbeat é responsável pela verificação e tomada de decisões. O Mon é responsável pelo monitoramento dos serviços e enviar, caso configurado, informações ao Heartbeat.

Assim esta monografia foi estruturada em 6 partes, de tal forma a elucidar primeiramente os conceitos básicos para um melhor entendimento do leitor e logo, partindo para a parte teorica de Cluster de alta disponibilidade.

No Capítulo 2 foi apresentado uma pequena historia de um grande sistema operacional que surgiu na Universidade de Helsinky na Finlândia em 1991, sistema esse, que atualmente é conhecido com Linux. Nele foi abordado também tipos de licenças bem como uma visão geral desse formidável sistema operacional.

No Capítulo 3 foi apresentado a teoria geral de Cluster visando elucidar seu principio de funcionamento e principalmente destacar os principais tipos de Cluster que se tem atualmente que são Cluster de Processamente, Cluster de Balanceamento de Carga e Cluster de Alta Disponibilidade.

No Capítulo 4 foi apresentado a teoria de um Cluster de Alta disponibilidade tendo como principal característica a de estar abortando os conceitos de Alta Disponibilidade , bem como estar criando um jargão em comum para o meio de forma a padronizar os termos utilizados, além disso demonstrar a diferença entre avaria, erros e falhas e finalmente elucidar as fases na tolerância a falha, que consiste basicamente na detecção de erros, confinamento de estragos, recuperação de erros e tratamento de falhas.

E finalmente no Capítulo 6 será apresentado um estudo de caso que visa comprovar a funcionalidade das ferramentas bem como da teoria apresentada.

Capítulo 2

Linux

2.1 Introdução

O Linux é um sistema operacional criado por Linus Torvalds na Universidade de Helsinky na Finlândia em 1991.

O Linux tem em suas inúmeras características importantes, a de ser um sistema operacional de código fonte aberto e de ser distribuído gratuitamente pela Internet, conforme disse (BONAN, 2003).

Segundo (NORTON; GRIFFITH, 2000) o Linux representa um novo modelo de computação, bem como de distribuição.

A primeira versão oficial do Linux foi liberada em 5 de outubro de 1991, que teve como principais características,alguns programas GNU[1], como por exemplo, o Bash[2] e o GCC[3].

Dentre inúmeras características que o Linux tem é possível destacar:

1. **Poder:**
   O sistema operacional Linux tem a capacidade de transformar microcomputadores considerados obsoletos e ultrapassados em uma máquina poderosa para determinados tipos de tarefas. Isso é possível por diversos fatores, entre eles, por ter sido projetado desde o inicio para permitir o acesso a *hardware* e periféricos a um nível bastante básico.

[1]GNU é uma sigla recursiva que significa Gnu is Not Unix e é uma organização dedicada à criação de software livre. O Projeto GNU foi iniciado em 1984 para desenvolver um sistema operacional completo, compatível com o UNIX, que fosse software livre, para mais informações acesse http://www.gnu.org/home.pt.html

[2]Interpretador de linha de comando

[3]Compilador da linguagem C do projeto GNU, ele é gratuito e de código fonte aberto

2. **Preço:**
   Por ser gratuito, o Linux é desenvolvido e mantido por vários programadores voluntários.A maioria das distribuições Linux vem com vários pacotes de *software* que valem muito dinheiro se fossem adquiridos de empresas.

3. **Desenvolvimento:**
   O sistema operacional Linux oferece várias linguagens de programação maduras e respeitadas ferramentas de desenvolvimento. O Linux possui disponíveis diversos conjunto de ferramentas, IDEs e principalmente bibliotecas que tornam esse sistema operacional um sonho para os desenvolvedores.

4. **Portabilidade:**
   Uma das suas principais características é justamente a de ele ser executado em uma diversa gama de arquiteturas de processadores diferentes, como por exemplo: i386, Alpha, Arm, M68k, Mips, Ppc e Sparc.

5. **Distribuições:**
   Conforme disse (NORTON; GRIFFITH, 2000) ao contrário da crença popular, o Linux na verdade é apenas a porção Kernel do sistema operacional. Já as distribuições são um conjunto de pacotes de programas, normalmente regido pela GNU, mais o Kernel Linux, onde este conjunto se pode dar o nome de GNU/Linux ou somente Distribuição Linux.

Normalmente a maioria das distribuições estão disponíveis gratuitamente e para compra no varejo. A única diferença entre se baixar uma distribuição pela internet ou a de comprar é que quando se compra, recebe-se os manuais para auxiliar na configuração mais a assistência técnica.

Existem inúmeras distribuições Linux disponíveis na Internet, dentre elas, podem ser citadas a Suse Linux, Turbo Linux, Debian Linux e Slackware Linux[4].

Segundo (SZTOLTZ; TEIXEIRA; RIBEIRO., 2003) de uma forma geral, pode-se dizer que o Linux é um sistema operacional multiusuário, multitarefa e multiprocessado, de livre distribuição, baseado no sistema operacional UNIX . A origem do nome Linux vem justamente do nome de seu criador Linus Torvalds.

Ser multiusuário significa ter a possibilidade de que várias pessoas utilizem o mesmo computador e ao mesmo tempo, utilizando para isso conexões remotas ou terminais, essa característica, a de ser multiusuário é muito importante, principalmente no quesito de permissões, pos através disso é possível controlar o acesso a programas, que podem ser chamado de processos e também controlar o acesso

[4]Esta será a distribuição usada nesta monografia, a mesma pode ser baixado em www.slackware.org

aos arquivos , de forma que cada usuário tenha uma permissão diferente do outro, permissões esta que podem ser de execução, leitura e gravação.

Quando se diz que o Linux é multitarefa isso significa que ele é capaz de executar diversos programas, tarefas ou serviços ao mesmo tempo, de forma que possibilite rodar simultaneamente vários programas, como por exemplo: um servidor web, um servidor de e-mail, um servidor de arquivo e ate mesmo um banco de dados. Vale ressaltar que tudo isso é feito de forma transparente para o usuário.

Já no que diz respeito a multiprocessado, é a característica que o Linux tem de suportar mais de um processador e é capaz de utilizar de maneira inteligente esses processadores, de uma forma a obter o melhor desempenho possível.E finalmente, a livre distribuição representa que ele pode ser distribuído de forma que não se tenha que pagar por licenças de uso.

2.2 Tipos de licenças de *software*

O Linux tem o seu código fonte liberado na forma de *software* livre, aonde é importante elucidar que *software* livre, comummente chamado de livre distribuição é usado para caracterizar o *software* que pode ser copiado livremente e que possui junto o seu código fonte para que caso ocorra a necessidade, o usuário o consulte e ate mesmo o altere, porem é importante salientar também que existem vários tipos de licença, sendo a mais utilizada a GPL[5].

Já quando é dito *freeware* é o programa que é gratuito e não contem seu código fonte junto. Já o *shareware* é quando um programa pode ser copiado mas ele funcionara somente em modo de demonstração aonde expirado o tempo o usuário deverá comprar a sua licença.

E finalmente o *software* comercial que é aquela que é necessário comprar uma licença de uso da empresa que o desenvolveu e na maioria das vezes tem o código fonte não disponível.

2.3 Visão geral do sistema operacional Linux

Basicamente, conforme disse (SZTOLTZ; TEIXEIRA; RIBEIRO., 2003) o sistema operacional Linux é compostos dos seguintes componentes: kernel, aplicações de sistema e aplicações de usuário.

Embora o kernel do Linux seja uma das partes mais importantes do sistema operacional, ele por si só não constitui o sistema como um todo, aonde ele é

[5]General Public License - Licença Pública Geral.

somente o núcleo do sistema e tem como principal finalidade a de ser responsável pelas funções de mais baixo nível, dentre elas é possível citar: gerenciamento de memória, gerenciamento de processos e ate mesmo gerenciamento das CPUs.Além disso ele ainda é responsável pela parte do suporte ao sistema de arquivos, periféricos, dispositivos e das placas de forma geral.

O utilizados do Linux deve ter sempre em mente que o kernel do Linux pode ser compilado [6] de forma que ele se adeque melhor ao tipo de máquina e ao tipo de serviços e tarefas que essa máquina vai desempenhar. Um bom exemplo é justamente quando se necessite de algum recurso, como por exemplo um protocolo de comunicação que não veja por padrão compilado ou até mesmo se não houver a necessidade de se usar um determinado tipo de protocolo ou até mesmo um determinado tipo de modulo. Esse tipo de adequação resulta em um kernel menor, com isso liberando mais espaços em memória para os programas que serão executados.

No que diz respeito a aplicações de sistema, o kernel faz um pouco sozinho, uma vez que é provido por ele os recursos que são necessários para que outro programas sejam executados, logo é necessário a utilização de programas para implementar os vários serviços necessário ao sistema operacional, conforme disse (SZTOLTZ; TEIXEIRA; RIBEIRO., 2003)

A grande diferença entre a aplicação do sistema e a aplicação do usuário é no que dis respeito ao propósito de cada aplicação, normalmente as aplicações de sistema são necessárias para fazer o sistema funcionar, como por exemplo a função init[7] aonde ele é o primeiro programa lançado após a execução do kernel na memória e ele é responsável por continuar o processo de boot executando os outros programas, enquanto que as aplicações do usuários são os programas utilizados pelo cliente para realizar suas tarefas, como por exemplo, um editor de texto.

Logo é possível concluir que as aplicações do usuário são todos os programas utilizados pelo usuário para executar determinada tarefa, dentre elas é possível citar: Editor de texto, editor de imagens, navegador e leitor de email.

[6] Compilação é o processo de transformação de um código-fonte em um programa executável.
[7] para mais informações consulte (SZTOLTZ; TEIXEIRA; RIBEIRO., 2003)

Capítulo 3

Cluster

3.1 Introdução

Segundo (PITANGA, 2003) na sua forma mais básica, um cluster é um sistema que compreende dois ou mais computadores ou sistemas (denominados nodos) que trabalham em conjunto para executar determinada aplicações ou realizar tarefas, de tal forma que os usuários do agrupamento de máquinas tenham a impressão de que somente um único sistema responde para eles, criando assim uma ilusão de um recurso único.

A tecnologia de Cluster teve seu inicio nas máquinas de alto desempenho, também conhecidas como supercomputadores, porém o seu desenvolvimento em PC‘s começou em 1994 pela NASA[1], que teve como principal fator de motivação o alto preço dos supercomputadores.

Conforme disse (PITANGA, 2003) esse projeto foi iniciado pela NASA, pelo principal motivo de que a mesma precisava de alto processamento na ordem de alguns gigaflop. E nessa época, uma máquina, a esse nível de desempenho de processamento custava alguns milhões de dólares. Como resultado disso, Thomas Sterling e Donald J. Backer resolverão interligar 16 PC‘s, cada um era dotado de um processador 486 DX4-100, utilizava o sistema operacional Linux e os conectaram através de um rede.

Inicialmente esse grupo de computadores interligado pela rede atingiu a marca de 70 megaflops tendo um custo de aproximadamente quarenta mil dólares, o que equivale a pouco mais que dez por cento do preço de uma supercomputador equivalente na época.Esse grupo de computadores recebeu o nome de Cluster podendo possuir vários tipos de configurações diferentes, tanto no quesito *hardware* como *software*.

[1]National Aeronautics and Space Administration - www.nasa.org

Segundo (PITANGA, 2003) um Cluster é composto por um conjunto de nós processadores (PCs ou estações) autônomos e que interligados por uma rede comportam-se como um sistema de imagem única.

O que o autor (PITANGA, 2003) quis dizer com imagem única é que, pode ser um sistema paralelo ou distribuído, que pode ser multiprocessado ou não e podem estar geograficamente localizado em lugares diferentes, porém comportam-se como um sistema centralizado, o que é considerado como Cluster.

3.2 Princípio de um cluster

Para um Cluster ser realmente útil, segundo (FILHO, 2002), ele deve seguir alguns conceitos básicos como:

1. **Comodidade:**
   Todos os computadores, ao qual são chamados de nós, devem estar interconectados por uma rede genérica. Já com relação ao sistema operacional, ele deve ser padrão, pois o *software* de gerenciamento deve ir acima deles como uma aplicação comum.

2. **Escalabilidade:**
   Deve haver a possibilidade de adicionar aplicações, periféricos, nós e até mesmos mais interconexões de rede sem interromper a disponibilidade dos serviços do Cluster.

3. **Transparência:**
   O Cluster, que é construído por vários nós independentes e fracamente acoplados, deve parecer como se fosse apenas um computador para o cliente.

4. **Confiabilidade:**
   O Cluster tem que ser dotado da capacidade de detectar falhas internas, e assim, tomar a decisão cabível para solucionar o problema, para que o mesmo, não venha a prejudicar os serviços oferecidos.

5. **Gerenciamento e Manutenção:**
   Umas das grandes dificuldade de um Cluster é a sua manutenção e principalmente a sua configuração, pois na maioria das vezes são tarefas morosas e que tem grande propensão a erros.

Então um Cluster deve ter um ambiente que facilite a sua configuração e principalmente a sua administração, a fim de que o Cluster não seja um grande sistema complexo, com um moroso trabalho de configuração e administração.

3.3 Tipos de cluster

- **Cluster de Processamento Paralelo:**
  Um Cluster de Processamento Paralelo tem como principal característica a de que a cada novo processo inicializado, o Cluster pega o processo e divide entre os computadores. Utilizando essa tecnologia, o tempo de término de processamento desse processo torna-se consideravelmente menor do que se fosse realizado em um único computador.

- **Cluster de Balanceamento de Carga:**
  Um Cluster de Balanceamento de Carga tem como principal finalidade estar distribuindo a carga de informações entre servidores para que, no grupo de servidores, um único servidor não fique sobrecarregado e outros sem carga.

- **Cluster de Alta Disponibilidade:**
  Um Cluster de Alta Disponibilidade tem como principal finalidade a de manter os serviços de um servidor disponível o máximo de tempo possível, onde para isso, utiliza-se técnicas de replicação de arquivos e serviços, e redundância de hardware.

Capítulo 4

Alta Disponibilidade

4.1 Introdução

Um sistema de Alta Disponibilidade tem como principal finalidade a de estar o maior tempo possível disponível para que seus serviços, de alguma forma, não sejam interrompidos. Atualmente, as redes e *hardware* vêm se tornando cada vez mais rápidos e de uma certa forma, mais baratos, o que os torna muito mais atrativos e a principal conseqüência disso é que cada dia uma empresa acaba dependendo mais de um sistema computacional para realizar as tarefas críticas, onde algum tipo de falha, poderia causar danos materiais e até mesmo, em algumas ocasiões, a perda de vidas humanas.

Todo sistema está, de forma direta ou indireta, sucessível a falhas, que podem ser contornadas, usando algumas técnicas para garantir a continuidade desse serviço.

Segundo (FILHO, 2002) estas técnicas podem ser tanto no nível de *hardware* como no de *software*.

Conforme disse (PITANGA, 2003) não são raras as situações em que disco, fontes de alimentação, interfaces de rede e outras partes do sistema começa a apresentar falhas entre 30.000 e 80.000 horas de uso. Quando isso acontece e seu trabalho começa a ser penalizar, já é hora de pensar em uma solução que seja tolerante a falhas e que apresente uma boa relação custo/benefício.

Devido ao caso de algum sistema estar suscetível a falhas, Clusters são implementados de tal modo que quando um servidor primário falhe, outro servidor secundário tome o lugar do primário de forma transparente ao usuário.

Segundo (PITANGA, 2003) um Cluster de Alta Disponibilidade visa manter a disponibilidade dos serviços prestado por um sistema computacional replicando serviço e servidores através da redundância de *hardware* e reconfiguração de *soft-*

ware

4.2 Conceitos

Segundo (FILHO, 2002) primeiramente, é necessário compreender que a redundância de recursos é um pré-requisito para se atingir um alto grau de disponibilidade.

Lembrando que as falhas não são restritas somente aos *hardwares*, mas também à redes de computadores, às redes elétricas e à localização dos recursos.

No que diz respeito a redes de computadores, deve ser levar em consideração os *hubs*, *switchs*, cabos, roteadores e todos equipamento físico de uma rede.

Já com relação as redes elétricas deve se entender que qualquer tipo de oscilação ou indisponibilidade do serviço deve ser suprima por outra forma de geração de energia, como *no-break* e até mesmo geradores.

E finalmente com relação a localização física do recursos deve-se sempre lembrar que um local está propício a furacões, enchentes, terremotos, roubos e até mesmo ataques terroristas, então é importante sempre ter equipamentos em locais distinto, onde na falta de um o outro o supra, de forma que não pare o serviço.

Então, quando se diz que um sistema é de alta disponibilidade, este sistema deve envolver não somente uma simples redundância de recursos, mas sim toda uma estrutura que o suporte. Pode ser visto que quanto maior o grau de disponibilidade, maior será o custo para tal.

Um fator que deve ser levando em consideração em um sistema de alta disponibilidade é como medir a disponibilidade dos serviços. Para tal deve ser levado em consideração o tempo em que o serviços ficam ativos e quanto tempo o serviços ficam foram do ar.

Para se calcular a disponibilidade de um sistemas, basta utilizar a seguinte fórmula, conforme disse (GUNDA; TECHNOLOGIES, 2001):

$$AD = \frac{TA - TD}{TA} X 100$$

Onde AD $=$ Alta disponibilidade, TA $=$ Total de horas por ano Ativo e TD $=$ Total de horas por ano Desativado.

A tabela 4.1 apresenta um exemplo do resultado da formula conforme disse (FILHO, 2002) e (MARCUS; STERN, 2003), e na tabela 4.2 é apresentado uma outra visão de classificação da disponibilidade segundo (RUSSEL et al., 2001).

4.3 Confiabilidade e Disponibilidade

A confiabilidade é a habilidade de se confiar em alguém ou alguma coisa.

Tabela 4.1: Medindo a disponibilidade de um servidor

Tempo-Ativo	Tempo-Desativado	Tempo-Desativador-por-Ano	Tempo-Desativado-por-Semana
98%	2%	7.3 dias	3 horas e 22 minutos
99%	1%	3.65 dias	1 hora e 41 minutos
99.8%	0.2%	17 horas e 30 minutos	20 minutos e 10 segundos
99.9%	0.1%	8 horas e 45 minutos	10 minutos e 5 segundos
99.99%	0.01%	52.5 minutos	1 minutos
99.999%	0.001%	5.25 minutos	6 segundos
99.9999%	0.0001%	31.5 segundos	0.6 segundos

Tabela 4.2: Classificação da disponibilidade

Porcentagem	Tempo-Desativado	Classificação
99.5%	3.7 dias	Convencional
99.9%	8,8 horas	Disponivel
99.99%	1 minutos	Alta Disponibilidade
99.999%	6 segundos	Resistente a Falhas
99.9999%	0.6 segundos	Tolerante a Falhas

A disponibilidade é qualidade ou estado do que é disponível.

A disponibilidade e a confiabilidade são palavras ou termos muito parecidos.

Conforme pode ser visto em (FILHO, 2002) o objetivo básico da tolerância a falhas é aumentar a confiabilidade de um certo sistema, utilizando-se tolerância a falhas, muitas falhas que podem ocorrer são evitadas, aumentando-se assim a confiabilidade.

O que é possível concluir é que utilizando as técnicas de tolerância a falhas, aumenta-se a confiabilidade e a disponibilidade de um serviço.

4.4 Taxonomia de Alta-Disponibilidade

Esta sessão tem como principal finalidade elucidar os principais termos da alta-disponibilidade, bem com o seu sentido ou melhor dizendo, o seu significado, a fim de formar um vocabulário padrão da área, conforme disse (FILHO, 2002).

4.4.1 Disponibilidade básica

A Disponibilidade Básica, também conhecida como *Basic Availability - BA* é quando um sistema é desenhado, projetado, implementado e implantado com um

número suficiente de equipamentos para satisfazer sua funcionalidade, onde qualquer sistema de rede que funcione corretamente já pode ser considerado como disponibilidade básica.

4.4.2 Alta disponibilidade

A Alta Disponibilidade consiste nas mesmas teorias da Disponibilidade Básica, e além de tudo a Disponibilidade Básica possui, a Alta Disponibilidade que possui redundância nos equipamentos, para que na falha de um, essa falha seja mascarada por outro equipamento, de forma transparente ao usuário.

Quando é dito mascara, isso significa impedir a sua visualização por um observador externo, conforme disse (FILHO, 2002). Porém o nível de transparência, ou melhor dizendo, de mascaramento nesse processo pode levar a algumas modificações no sistema de "Alta Disponibilidade".

Com relação aos tipos de mascaramento pode-se citar:

- **Manual Masking - MM:**
  onde após uma falha é necessário uma intervenção manual para o sistema voltar.

- **Cold Standby - CS:**
  onde após uma falha, os usuário são desconectados e perdem todo o trabalho e o sistema redundante está desativado precisando ser inicializado para poder começar a operar.

- **Warm Standby - WS:**
  da mesma forma que o Cold Stanby, porém o sistema redundante já esta funcionando e operando, porém os usuário são desconectados e perdem o trabalho em progresso.

- **Hot Standby - HS ou Active Replication - AR:**
  neste tipo de mascaramento, todos os componentes estão fortemente agrupados e são transparentes ao usuário, o estado do processamento é ativa e completamente compartilhado.

Além disso deve ser levado em consideração a necessidade do sistema por Alta Disponibilidade, como por exemplo, em alguns casos não são necessários mascarar as falhas de hardware, só as de software e vice-versa.

Nem sempre é possível mascarar todas as falhas. Assim, quando desenvolver um projeto de Alta Disponibilidade é importante especificar qual é o conjunto de falhas que serão compensadas pelo sistema de Alta Disponibilidade.

4.4.3 Disponibilidade contínua

Como pode ser visto, a Alta Disponibilidade somente mascara as falhas, ou seja, mascara as paradas não planejadas dos serviços providos em uma rede, já quando dizemos Disponibilidade Contínua, ela estende a definição de Alta Disponibilidade a um ponto que até se mascara as paradas planejadas do serviços.

Este modelo de disponibilidade é exclusivo para o *Hot Standby* ou *Active Replication*.

A Disponibilidade Contínua é possível somente na teoria, onde a taxa da disponibilidade seria igual a 1, ou seja, o sistema nunca pararia por nada, conforme disse (GARCIA, 2003).

4.5 Avarias, erros e falhas

Como a principal característica da alta disponibilidade é a de mascarar falhas tornando-a imperceptível ao usuário, o termo "falha"acada sendo usado de uma forma vaga.

Alguns termos que devem ser levados em consideração são:

- **Avaria:**
  é quando ocorre um avaria no comportamento do sistema e ele é desviado do que foi designado a fazer.

- **Erro:**
  é a parte do estado do sistema que está suscetível a levar a avarias.

Conforme pode ser visto, quando há um erro no estado do sistema, logo se tem uma seqüencia de ação que pode ser executadas e que levarão a avarias no sistema, conforme disse (FILHO, 2002).

A causa de um erro é uma falha. Como um erro é a propriedade de um sistema, o erro pode ser observado e avaliado, já uma avaria não, pois ela não é uma propriedade do sistema. A avaria é deduzida detectando-se a ocorrência de algum erro no sistema. Já as falhas são associadas com a noção de defeitos e falhas é definido como sendo os defeitos que possuem o potencial de gerar erros.

Conforme disse (FILHO, 2002) a duração da falha e o seu momento de ocorrência, podem ser classificadas como transientes ou permanentes.

Onde as transientes são aquelas que tem uma duração limitada, que podem ser causadas por algum mal funcionamento temporário ou até mesmo por interferências.

Já as permanentes são aquelas que após ter falhado, nunca mais voltam a funcionar normalmente. Deve-se levar em consideração, que quando utilizado um sistema distribuído, é possível classificar as falhas de acordo com o comportamento do defeito, conforme disse (FILHO, 2002):

- **Falhas de Travamento:**
  são aquelas que causam a perda do estado interno do componente defeituoso ou pode até mesmo causar o travamento. A falha de travamento é conhecida como *Crash Faults*.

- **Falhas de Omissão:**
  é quando uma falha faz com que o componente não responda a algumas requisições. A falha de omissão é conhecida como *Omission Faults*.

- **Falhas de Timing:**
  é quando um componente responde muito rápido ou muito devagar, essa falha é conhecida também pelos nomes de *falhas de performance* ou *Timing Faults*.

- **Falhas Bizantinas:**
  é quando um componente se comporta de forma completamente arbitrária. A falha bizantina é conhecida como *Byzantine Faults*.

4.6 Fases na tolerância a falhas

Anteriormente foi verificado que um sistema tolerante a falhas tenta evitar a ocorrência de avarias no sistema mesmo quando ocorre uma falha. Um sistema tolerante a falhas dependerá totalmente da arquitetura empregada e ao projeto dos recursos e serviços que serão providos.

Abaixo será descrito algumas técnicas gerais de como adicionar tolerância a falhas em um sistema de Alta Disponibilidade.

4.6.1 Detecção de erros

O primeiro passo para a tolerância a falhas é detectar o erro. As falhas de avaria não podem ser detectadas diretamente, mas podem ser detectadas a partir de um erro no sistema.

Dessa forma é adequado executar teste para verificar a sua possível ocorrência. Através disso é possível saber a eficiência de um esquema de tolerância a falhas na capacidade de se detectar erros. Por causa da importância da detecção dos erros é preciso determinar como deve se proceder os teste ideal para a detecção de erros.

Existem algumas formas importantes que devem ser levadas em consideração durante esse processo que são:

- O teste deve se basear nas especificações do sistema e não pela constituição interna, de tal forma que a sua implementação seja ignorada, tratando como se fosse uma caixa preta.

- O teste deve verificar todos os possíveis erros existentes e nenhum erro deve ser declarado quando não existente.

- O teste deve ser independente do sistema no que se diz respeito à suscetibilidade de falhas.

Quando realizado o teste de detecção de erro, o teste pode ocorrer de várias formas, dependendo do sistema e dos erros de interesse, conforme disse (FILHO, 2002):

1. **Teste de Replicação:**
   Este teste tem como principal característica a de replicar um componente do sistema, compará-lo e retornar resultados de diferentes sistemas com a finalidade de detectar algum erro de replicação.

2. **Teste de Timing:**
   Este teste realiza uma solicitação a algum componente e verifica se o tempo de resposta excede ou não a restrição imposta. Eles podem ser usados tanto ao nível de hardware com ao de software.

3. **Teste Estruturais e Codificação:**
   Neste tipo de teste são realizados teste de semântica que visa garantir se o valor é consistente com o resto do sistema e teste estruturais que considera a informação e garantem que internamente a sua estrutura é como deveria ser, onde a forma mais comum é a codificação.

4. **Teste de Coerência:**
   Este teste tem como finalidade a de determinar se o estado de algum objeto no sistema está coerente.

5. **Teste de Diagnósticos:**
   Neste teste um sistema usa algumas técnicas de verificação em seus componente para verificar se o mesmo está funcionando de forma correta.

4.6.2 Confinamento de estragos e avaliação

Um sistema deve ser monitorado constantemente para que não haja um intervalo entre a falha e a detecção de erro. Caso isso não seja feito, o erro pode se propagar para outras partes do sistemas.

Conforme diz (FILHO, 2002) por isso, antes de executar medidas corretivas, é necessário determinar exatamente os limites da corrupção, ou seja, as partes do estado que estão corrompidas.

Para se avaliar os estragos que ocorreram após a detecção de um erro, deve ser examinado o fluxo das informações entre os diferentes componente, então deve ser feito algumas suposições referentes a origem e ao momento da geração do erro onde o objetivo disso é encontrar as barreiras no estado onde nenhuma das trocas de informações tenha sido feita.

Segundo (FILHO, 2002) as barreiras podem ser encontradas dinamicamente, gravando-se e examinando-se o fluxo das informações.

Por serem complexos, a melhor maneira é implementar um sistema tal que um *Firewall* sejam estaticamente incorporado no sistema, com a finalidade de garantir que nenhuma informação se espalhe além deles.

4.6.3 Recuperação de erros

Após a detecção do erro e sua identificação, o próximo passo a ser feito é retirá-lo do sistema. As técnicas para se recuperar de erros segundo (FILHO, 2002) são:

- **Recuperação para Trás:**
  Nesta técnica, o estado do sistema é retornado a um estado anterior a sua falha, na esperança de que neste estado esteja livre de erros. Para isso *check-points* periódicos são estabelecidos. Este é uma técnica muito utilizada, pois não depende da natureza da falha.

- **Recuperação para Frente:**
  Nesta técnica, nenhum dos estados do sistema está disponível, onde é feita a tentativa de se ”ir para frente” e tentar tornar o estado livre de erros aplicando medidas corretivas. Quando utilizado essa técnica pode ocorrer overhead.

4.6.4 Tratamento de falhas e serviços continuados

Como pode ser visto, o foco sempre foi o "erro"propriamente dito, onde caso ele tenha sido gerado por uma falha transiente não precisa ser tratada pois provavelmente o erro já desapareceu, mas caso tenha sido gerado por um erro permanente,

então teoricamente, o erro tende a estar presente após a sua recuperação, onde que no novo sistema o erro tende a ocorrer novamente.

Algumas técnicas são utilizadas para identificar o componente defeituoso e evitar que ocorra o erro novamente. Segundo (FILHO, 2002), esta fase possui duas sub-fases:

- **Localização da Falha:**
  Nesta fase tem que se identificar o erro. Caso não for possível sua localização, não será possível reparar o sistema.

- **Reparo do Sistema:**
  Nesta fase o sistema é reparado de forma que o componente defeituoso não seja novamente utilizado. A manutenção tem que ser feita on-line sem a intervenção manual. A restauração ou melhor dizendo o reparo é feito por um sistema que possui um método de reconfiguração dinâmica, de tal forma que o sistema distribuído seja usado para substituir o componente que apresenta falhas ou defeito.

4.7 Software utilizado

Os principais software utilizados para se ter uma alta-disponibilidade são o Drbd[1] que tem como finalidade fazer a replicação dos dados, o Heartbeat que tem como finalidade verificar o sistema e tomar decisões para contornar os erros apresentados e o Mon[2] que é responsável pelos alertas.

Cada um será abordado no próximo capítulo.

[1] Data Replication Block Device
[2] Service Monitoring Daemon

Capítulo 5

Ferramentas

Este capítulo visa elucidar as 3 ferramentas utilizadas nesta monografia para implementar um Cluster de Alta Disponibilidade e ferramentas estas que podem ser baixada gratuitamente pela internet ou copiadas do CD que está no Anexo.

Para verificar as opções de configuração dos programas veja o apêndice A

5.1 Drbd

5.1.1 Introdução

O Drbd[1] é um sistema de replicação desenvolvido para Cluster de Alta Disponibilidade. O que ele faz é exatamente criar um espelho de um dispositivo a outro utilizando a rede como meio de transmissão.

5.1.2 Como funciona

O Drbd funciona criando uma imagem, que na verdade é uma copia exata, de um dispositivo em outro, utilizando para isso, a rede como seu meio de transmissão.

Os dispositivos Drbd funcionam em dois estado: primário e secundário.

Quando um dos dispositivos é setado como primário e outro como secundário, todo o conteúdo do primário é transferido ao dispositivos secundário, esse processo é chamado de sincronização. Após feito a sincronização, somente será transferido informações para o dispositivo secundário, se houver um processo de escrita no dispositivo primário.

É possível criar vários espelhamentos, sendo limitado inicialmente somente pela largura de banda da rede.

[1] Data Replication Block Device

5.1.3 Como instalar

O primeiro passo a ser feito é baixar o pacote do Drbd[2]

Utilizando o usuário root, descompacte o Drbd com o comando `tar -zxvf drbd-0.7.10.tar.gz`

Entre no diretório do drbd, `cd drbd-0.7.10`

e compile o drbd com os comando `make`

e instale-o com o comando `make install`

Pronto se não der nenhum erro o Drbd foi instalado com sucesso.

O segundo passo é criar os device Drbd que são bem simples, a sintaxe para se criar é a seguinte:

mknod -m 0660 /dev/drbd*X* b 147 *X*

Onde o X é o número do dispositivo, abaixo segue dois exemplos de como criar

```
mknod -m 0660 /dev/drbd0 b 147 0
mknod -m 0660 /dev/drbd1 b 147 1
```

5.1.4 Comandos

Está sessão tem como principal finalidade elucidar os principais comandos que podem ser utilizado para configurar o Drbd e até mesmo estar interagindo com o mesmo.

drbd

Este comando é um script para gerenciar o drbd, que tem como principal finalidade a de inicializar ou parar os dispositivos Drbd.

Sintaxe

drbd *{start|stop|status|reload|restart|force-reload}*

Comandos

start: Inicia

stop: Para

status: Mostra o status dos dispositivos

[2] http://oss.linbit.com/drbd/

reload: Recarrega a configuração

restart: Seria a mesma coisa que um Stop e depois um Start.

force-reload: Serve para recarregar a configuração de uma modo forçado.

drbdsetup

O drbdsetup é um aplicativo para configurar o Drbd direto pela linha de comando.

Outra características principal do drbdsetup é estar verificando o status de cada dispositivos Drbd e caso seja necessário, estar modificando o estado do dispositivo em questão.

Sintaxe

```
drbdsetup device  disk  lower_dev  meta_data_dev  meta_data_index
[ -d size ] [ -e err_handler ]

drbdsetup device net local_addr [ :port ] remote_addr [ :port ] protocol
[ -c time ] [ -i time ] [ -t val ] [ -S size ] [ -k count ] [ -d
discon_handler ]

drbdsetup device syncer [ -k ] [ -g group ] [ -r rate ] [ -e extents ]

drbdsetup device disconnect

drbdsetup device detach

drbdsetup device down

drbdsetup device primary [ -h ] [ -t ] [ -d ]

drbdsetup device secondary

drbdsetup device on_primary [ -h ] [ -t ]

drbdsetup device invalidate

drbdsetup device invalidate_remote

drbdsetup device wait_connect [ -t wfc_timeout ] [ -d degr_wfc_timeout ]
```

```
drbdsetup device wait_sync [ -t wfc_timeout ] [ -d degr_wfc_timeout ]

drbdsetup device state

drbdsetup device cstate

drbdsetup device resize [ -d size ]

drbdsetup device show
```

Comandos

DISK

Serve para associar dispositivos de armazenamento sobre um bloco de dados. O -d (ou –disk-size) deve ser usado somente se for necessário. Se não for utilizado o -d o dispositivo somente será operacional assim que conectado ao seu par.

-d, –disk-size *size*

É possível cancelar o método de detecção do tamanho do Drbd. Se for necessário usar o dispositivo antes de se conectar com o outro par, então essa opção deve ser especificada. A unidade de medida padrão é em KB, lembrando que 1KB é = a 1024 Bytes.

-e, –on-io-error *err_handler*

Se o driver de um dispositivo relatar algum erro ao Drbd, ele irá passar o erro a camada superiores do sistema operacional. As opções de err_handler são: pass_on, panic and detach.

NET

Configura os dispositivos para escutar em um endereço_local:porta para conexão entrantes e para tentar conectar ao endereço_remoto:porta. Se o valor da porta for omitido, o valor padrão a ser usado será 7788.
Em um link TCP/IP a especificação do protocolo deve ser usado. As especificações válidas dos protocolos são A, B e C.

O Protocolo A:

Uma operação de escrita é considerada completa, se está é escrita no disk local e enviado pela rede.

O Protocolo B:

Uma operação de escrita é considerada completa, se está é escrita no disk local e confirmado a recepção no disco remoto.

O Protocolo C:
Uma operação de escrita é considerada completa, se chegar a confirmação do disco local e do remoto.

A descrição das opções abaixo já foram elucidadas na sessão de parâmetro do arquivo de configuração.

-c, –connect-int *time*

-i, –ping-int *time*

-t, –timeout *val*

-S, –sndbuf-size *size*

-k, –ko-count *count*

-e, –max-epoch-size *val*

-b, –max-buffers *val*

-d, –on-disconnect *discon_handler*

SYNCER
A descrição das opções abaixo já foram elucidadas na sessão de parâmetro do arquivo de configuração.

-r, –rate *rate*

-k, –skip-sync

-g, –sync-group *group*

-e, –al-extents *extents*

PRIMARY
Configura o dispositivo no estado primário, isto significa que as aplicações podem abrir o dispositivo em modo de leitura e escrita. Os dados escritos no dispositivos primário são espelhados para o dispositivo secundário.
Não é possível ajustar ambos os dispositivos de um par conectado do dispositivo de Drbd ao estado preliminar.
Não é possível configurar ambos os dispositivos Drbd no estado primário.

-h, –human
Indica que o estado está sendo mudado pelo administrado e o Cluster reinicia o tempo precedente sobre a decisão para os outros nos.

-t, –timeout-expired
Indica que a mudança do estado foi por causa de um nó que não mostrou que inicio o Cluster, então o Cluster irá iniciar em modo degraded.

-d, –do-what-I-say
Torna-se primário, mais a replicação local pode se tornar inconsistente. Usando esta opção é possível forçá-lo a entrar no estado primário. Só utilize esta opção se souber o que está fazendo.

SECONDARY
Configura um dispositivo Drbd no estado secundário.
É possível que ambos os pares de um dispositivos Drbd estejam no estado secundário.

ON_PRIMARY
Isto configura o flags adicionais para a transação para o estado primário. As possibilidades das flags são *–inc-human* e *–inc-timeout-expired*.

INVALIDATE
Este comando força o dispositivo local que estão conectado ao Drbd ao estado de *SyncTarget*, que significa que todos os blocos de dados do dispositivos são copiados para o par.
Este comando falhará se o dispositivo não for parte de um par conectado ao dispositivo.

INVALIDATE_REMOTE
Este comando força o disco local de um par do dispositivo Drbd ao estado *SyncSoure*, que significa que todos os blocos dos dados do dispositivos serão copiados ao par.

WAIT_CONNECT
A descrição das opções abaixo já foram elucidadas na sessão de parâmetro do arquivo de configuração.

-t, –wfc-timeout *wfc_timeout*

-d, –degr-wfc-timeout *degr_wfc_timeout*

WAIT_SYNC
Retorna assim que o dispositivo sair do estado de sincronização e retornar ao estado de conectado. As opções são as mesmas do comando *wait_connect*.

DISCONNECT
Remove as informações configuradas pelo comando net para um dispositivo. Isso significa que os dispositivos em estados de não conectado não mais aguardarão um conexão entrante.

DETACH
Remove as informações configuradas pelo comando *disk* para um dispositivo. Isto significa que o dispositivo estará desconectado do seu dispositivo Drbd de armazenamento para votar ao modo normal.

DOWN
Remove todas as informações de configuração de um dispositivo forçando-o a voltar ao estado de não configurado.

STATE
Mostra os estado do dispositivo e a do seu par (local / remoto).

CSTATE
Mostra os estados atuais da conexão de um dispositivo.

RESIZE
Isto faz com que o Drbd reexamine o tamanho do dispositivo de armazenamento.

SHOW
Mostra todas as informações de configuração disponíveis de um dispositivo.

drbdadmin

O drbdadmin é um aplicativo para administração do Drbd

Sintaxe
drbdadm [*-d*] [*-c file*] [*-s cmd*] *command* [*all* | *resource ...*]
Opções

-d, –dry-run
Somente imprime as chamadas do drbdsetup ao stdout, mas não roda como um comando.

-c, –config-file *file*
Especifica o local do arquivo de configuração que será usado.

-s, –drbdsetup *file*
Especifica o caminho completo para o programa drbdsetup. Esta opção é omitida, pois o drbdadmin ira olhar para /sbin/drbd-setup e ./drbdsetup

Comandos

attach
Anexa um dispositivo de bloco local ao dispositivo Drbd. Na linguagem do sistema será uma operação mount.

detach
Remove o dispositivo de armazenamento do dispositivo Drbd.

connect
Ajusta a configuração da rede do dispositivo Drbd. Se os dois dispositivos já estiverem configurados, os dois se conectarão.

disconnect
Remove a configuração da rede do dispositivo Drbd. O dispositivo entrará no estado de StandAllone.

syncer
Lê os parâmetros de ressincronização do dispositivo.

up
É um atalho para o attach e connect.

down
É um atalho para a disconnect e detach.

primary
Altera o dispositivo Drbd para o estado primário. É necessário fazer isto antes de se montar um sistema de arquivo no dispositivo.

secondary
Altera o dispositivo Drbd para o estado secundário. Isto é necessário desde que um dos pares conectados ao dispositivo Drbd tem que ser secundário.

invalidate
Isto força o Drbd a considerar os dados no dispositivo local de armazenamento como *out-of-sync*. Para depois o Drbd copiar cada bloco do seu par, para trazer de volta a sincronização entre os dispositivos.

invalidate_remote
Este comando é similar ao comando de invalidate, mas o armazenamento suportando pelo par invalidado e reescrito com os dados do nó local.

resize
O Drbd ira reexaminar todas as constantes de um tamanho e redimensionará aos dispositivos Drbd de acordo.

adjust
Sincroniza a configuração dos dispositivos com o arquivo de configuração local.

wait_connect
Espera até que o dispositivos esteja conectado ao outro dispositivo.

state
Mostra os estados atuais dos dispositivos (local/peer). Por exemplo *primary*/*secondary*.

cstate
Mostra o status atual das conexões dos dispositivos.

dump
Apenas analisa gramaticalmente o arquivo de configuração e o depura. Pode ser usado para verificar o arquivo de configuração no que diz respeito as sintaxe.

5.2 Heartbeat

5.2.1 Introdução

A principal finalidade do Heartbeat é a de ser um monitor do estado do sistema e dependendo do resultado encontrado, tomar uma decisão.

Heartbeat representa batimentos cardíacos, esse termo foi escolhido porque o Heartbeat tem como principal características estar enviando pacotes a outra máquina para verificar se essa está ainda operante.

O Heartbeat é um dos principais programas para a criação de um sistemas de Cluster de Alta-Disponibilidade.

5.2.2 Como funciona

Após configurado o arquivo ha.conf, o Heartbeat vai ficar verificando de tempos em tempos as máquinas configuradas em seu arquivo de configuração. Caso alguma delas não responda ele poderá assumir os recursos desse computador que não esta respondendo.

Com apenas 2 nodos IP o Heartbeat pode assumir os recursos e serviços de um número ilimitado de interfaces IPs. O Heartbeat tem várias formas de estar verificando se um computador está ativo ou não, a primeira forma é utilizando a própria interface de rede e a outra é através de uma porta serial.

Cada pacote enviado pelo Heartbeat com a finalidade de checar se o outro computador esta ativo ou não é chamado de heartbeats.

5.2.3 Como instalar

Inicialmente é necessário baixar o pacote chamado libnet[3] versão 1.1.2.1, net-smnp[4] versão 5.1.3. e swig [5] versão 1.3.24.

Lembre-se que antes de começar qualquer compilação é importante já se ter criado o usuário hacluster e o grupo hacluster utilizando os comandos `useradd hacluster` e `groupadd hacluster`.

Descompacte o libnet com o comando `tar -zxvf libnet.tar.gz`

Entre no diretório libnet, `cd libnet` e inicie o script de configuração `./configure`

Compile o libnet com o comando `make` e depois instale-o com o comando `make install`

Agora descompacte o net-smnp com o comando `tar -zxvf net-snmp-5.1.3.tar.gz`

Entre no diretório net-smnp, `cd net-snmp-5.1.3/` e inicie o script de configuração `./configure`

Durante esse processo serão feitos algumas perguntas, caso tenha duvida, tecle <ENTER> para adotadar o valor padrão.

Compile-o com o comando `make` e depois instale-o com o comando `make install`

Descompacte o swig com o comando `tar -zxvf swig-1.3.24.tar.gz`

Entre no diretório swig, `cd SWIG-1.3.24/` e inicie o script de configuração `./configure`

Compile o swig com o comando `make` e depois instale-o com o comando `make install`

O último passo é baixar a última versão do heartbeat[6], que até o atual momento é a 1.99.4.

Descompacte com o comando `tar -zxvf heartbeat-1.99.4.tar.gz`

Entre no diretório do heartbeat, `cd heartbeat-1.99.4` e inicie o script configuração `./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var`

Compile o hearbeat com o comando `make` e depois instale-o com o comando `make install`

Feito isso, ainda no diretório do hearbeat copie o arquivo modelo de configuração ha.conf para o diretório /etc/ha.d, `cp doc/ha.cf /etc/ha.d/`

Finalmente copie os arquivos haresources e authkeys para o diretório /usr/local/etc/ha.d com os comandos `cp doc/haresources /etc/ha.d/` e `cp doc/authkeys /etc/ha.d/`

[3] http://unix.freshmeat.net/redir/second-libnet/34005/url_tgz/libnet-1.1.2.1.tar.gz

[4] http://www.net-snmp.org/download/

[5] http://www.swig.org/download.html

[6] http://handhelds.freshmeat.net/projects/linux-ha/

5.2.4 Comando

Está sessão tem como principal finalidade elucidar o principal comando que é utilizado para gerenciar o Heartbeat que é o heartbeat. Este comando é um script para gerencia-lo, que tem como principal finalidade a de inicializa-lo ou para-lo.

Sintaxe

Usage: /etc/rc.d/heartbeat {start|stop|status|restart|reload|force-reload}

Opções

start: Inicia.

stop: Para.

status: Mostra o status dos dispositivos.

restart: Seria a mesma coisa que um Stop e depois um Start.

reload: Recarrega a configuração.

force-reload: Serve para recarregar a configuração de uma modo forçado.

5.3 Mon

5.3.1 Introdução

O Mon tem como principal finalidade a de ser um monitor do sistema. Quando ele detecta uma falha é possível enviar um e-mail e até mesmo fazer com que ele acione o Heartbeat para que ele tome alguma decisão para a solução do problema.

5.3.2 Como instalar

O primeiro passo para se instalar o Mon é ter o perl instalado.

Após isto entre na shell do module MCPAN, para isso digite no console o comando:

```
perl -MCPAN -e shell
```

Caso for, a primeira vez que você esteja executando, responda o questionário, na duvida pode ir pressionando <Enter>, exceto na hora de configurar o local de onde se deve baixar o modulo[7].

[7] É necessário ter uma conexão com a Internet para poder baixar os arquivos

Feito isso, aparecera o prompt do CPAN, agora é só digitar os seguintes comandos para instalar os módulos adicionais:

```
install Time::Period
install Time::HiRes
install Convert::BER
install Mon::Client
exit
```

Os módulos para rodar o Mon já estão prontos e ainda se faz necessário baixar o programa fping [8] versão 2.4b2 para utilizar todas as funcionalidade do Mon.

Após baixado apenas descompacte-o com `tar -xvf fping.tar` , entre no diretório com `cd fping-2.4b2_to` e configure-o com `./configure` .

Feito isso compile o fping com `make` e instale-o com `make install` .

Agora baixe os arquivos do Mon[9] versão 0.99.2.

Feito isso descompacte-o com o comando `tar -jxvf mon-0.99.2.tar.bz2`

Agora a estrutura dos arquivos do Mon será, arquivos de configuração ficaram em /etc/mon/, o arquivo executável ficara em /usr/local/bin e os scripts em /usr/lib/mon/.

Entre no diretório do Mon, `cd mon-0.99.2`

Feito isso vamos criar a estrutura de diretório com os seguintes comandos:

```
mkdir /etc/mon/
mkdir /usr/lib/mon/
mkdir /usr/lib/mon/state.d/
mkdir /usr/lib/mon/log.d/
```

E por fim, colocar cada arquivo no seu lugar, primeiro copiando os arquivos de configuração:

```
cp etc/example.cf /etc/mon/
cp etc/auth.cf /etc/mon/
```

Agora devem ser copiados os arquivos executáveis, com o seguinte comando:

```
cp mon /usr/local/bin/
cp clients/moncmd /usr/local/bin/
cp clients/monshow /usr/local/bin/
```

E por fim, deve ser copiados os scripts, com os comandos:

```
cp alert.d/ /usr/lib/mon/ -R
cp mon.d/ /usr/lib/mon/ -R
```

[8] http://www.fping.com/download/

[9] http://freshmeat.net/projects/mon/

5.3.3 Comando

Está sessão tem como principal finalidade elucidar o principal comando que é utilizado para gerenciar o Mon que é o mon. O software Mon tem como principal finalidade a de ser o monitor dos serviços para a disponibilidade, emitindo alarmes para as falhas.

sintaxe: *mon* [*-dfhlMSv*] [*-a dir*] [*-A authfile*] [*-b dir*] [*-B dir*] [*-c config*] [*-D dir*] [*-i secs*] [*-k num*] [*-l dir*] [*-m num*] [*-p num*] [*-P pidfile*] [*-r delay*] [*-s dir*]

Descrição

O Mon é um monitor com finalidade geral de monitorar a disponibilidade do serviço e gerar alertas no caso de detectar falhas.

O Mon foi projetado para ser de fácil monitoração e também utilizar métodos de alertas através de uma relação comum, que são executados facilmente com os programas (em C, em Perl, em Shell, etc.), traps SNMP e traps MON (pacote do UDP).

Opções

-a *dir*
Indica o local dos scripts de alerta. O padrão é /usr/local/bin/mon/alert.d

-b *dir*
Indica o diretório base do mon. O padrão é /usr/lib/mon.

-B *dir*
Indica o diretório base do arquivo de configuração. Todos os arquivos de configuração devem ficar nesse diretório, incluindo o mon.cf, monusers.cf e auth.cf.

-A *authfile*
Indica o local do arquivo de autenticação. O padrão é /etc/mon/auth.cf, isso se o diretório /etc/mon exister, caso não, o padrão é /usr/lib/mon/auth.cf.

-c *file*
Lê o arquivo de configuração especificado em *file*.

-d
Ativa o mode de depuração.

-D *dir*
Indica o caminho do diretório de estado. O valor padrão é /var/state/mon, /var/lib/mon e /usr/lib/mon/state.d caso existão.

-f
força a rodar em modo daemon. Está é a maneira preferida de executar o mon.

-h
Mostra a tela de ajuda.

-i *secs*
Indica o intervalo de descanso, em segundos. O valor padrão é 1.

-k *num*
Configura o número máximo de linhas do log histórico. O valor padrão é 100.

-l
Lê o estado de um arquivo.

-L *dir*
Configura o local do diretório de log. O valo padrão é /var/log/mon.

-M
Pre-processa o arquivo de configuração utilizando a pacote m4.

-m *num*
Configura o número de trottle para o número máximo de processos.

-p *num*
Cria um porta de comunicação. O valor padrão é 2583.

-S
Inicia quando scheduler estiver parado.

-P *pidfile*
Grava o pid do servidor no arquivo *pidfile*. O valor padrão é /var/run/mon/mon.pid, /var/run/mon.pid, e /etc/mon.pid

-r *delay*
Configura o número de segundos a ser usado no atraso da inicialização após cada serviço ter sido programado. Mais informações verifique o comando randstart.

-s *dir*
Indica o caminho dos scripts de monitoração. O valor padrão é /usr/local/bin/mon/mon.d/

-v
Imprime as informações da versão

Capítulo 6

Estudo de Caso

Neste estudo de caso será abordado a implementação de um simples sistema de alta disponibilidade com a demonstração de como esse sistema funciona. O interessante dessa demonstração é que a partir dela pode-se modificar sua estrutura adaptando a diversas necessidade que possa vir a existir.

6.1 Estrutura

Esta implementação consiste na seguinte estrutura, um servidor primário, que terá sua base principal de dados replicada em um servidor secundário, que na sua queda o servidor secundário tomará seu lugar.

Está replicação foi feita da seguinte forma, foi centralizado em uma partição todos os arquivos de configuração, arquivos de dados e até mesmo o banco de dados. Foram criados links simbólicos para os arquivos de configurações, para que com isso não fosse mudado a estrutura padrão da localização desses arquivos.

Com relação a configuração dos equipamentos utilizado, bem como suas especificações para este estudo de caso, seguem especificados na figura x:

A configuração do servidor secundário é apresentada na figura x

foi utilizado um hub-switch para interligar os servidores onde sua configuração é apresentada na figura x

foi utilizado um roteador para fazer a conexão com a Internet onde sua configuração é apresentada na figura x

abaixo segue um diagrama ilustrando o mapeamento da localização dos equipamentos.

A lógica de funcionamento deste esquema é o seguinte, tem o servidor primário que está no IP 10.0.0.200 e o servidor secundário que esta no IP 10.0.0.4, o IP que será utilizado para acessar os serviços provido será o 10.0.0.201.

Servidor Primário
Processador: AMD XP 2600+
Memória: 512MB DDR 400 Mhz
Disco Local: 40 GB
Partições/Ponto de montagem: hda1 / , hda2 /home e hda3 swap
Partição replicada: hda2
Coolers: 1 lateral, 2 traseiros, 1 frontal e 2 para HD.
Fonte: 400 W
Placa mãe: Soyo Dragon
Placa de rede: 2 placas 3com corporation 3c905 100BaseTX
Nome: nd
IP-1: 10.0.0.200
IP-2: 192.168.200.1
Serviços: Replicação de disco, servidor de páginas e proxy
Distro/Kernel: Slackware 10.1 / 2.4.29

Figura 6.1: Servidor primário

Servidor Secundário
Processador: AMD XP 2000+
Memória: 512MB DDR 400 Mhz
Disco Local: 40 GB
Partições/Ponto de montagem: hda1 / , hda2 /home e hda3 swap
Partição replicada: hda2
Coolers: 1 exautor, 1 frontal e 2 para HD.
Fonte: 400 W
Placa mãe: PCChips
Placa de rede: 2 placas 3com corporation 3c905 100BaseTX
Nome: Servidor3
IP-1: 10.0.0.4
IP-2: 192.168.200.2
Serviços: Replicação de disco, servidor de páginas, proxy e monitoramento do roteador.
Distro/Kernel: Slackware 10.1 / 2.4.29

Figura 6.2: Servidor secundário

Os serviços providos nesta estrutura são: Serviço de Página e Proxy. Então quando inicializado o serviço, o servidor primário adicionará uma interface virtual com o IP 10.0.0.201.

Switch
Fabricante: 3com
Modelo: Switch - 3C16471
Portas: 24

Figura 6.3: Switch

Roteador
Fabricante: 3Com
Modelo: OfficeConnect 812
Portas: 4

Figura 6.4: Roteador

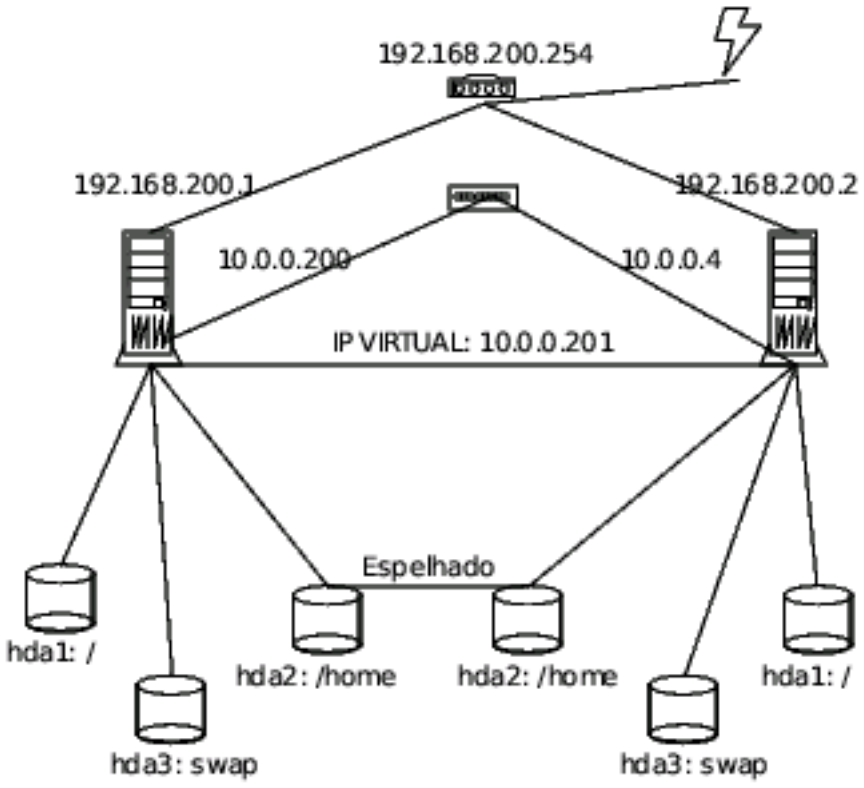

Figura 6.5: Mapeamento geral da rede de alta disponibilidade

Toda a comunicação entre os servidores é feito pelos IPs 10.0.0.200 e 10.0.0.4, isso inclui, a replicação das informações bem como a verificação se o servidor esta ativo ou não.

No caso da queda do primeiro servidor, isso será verificado pelo IP 10.0.0.200, o servidor secundário irá adicionar uma interface virtual e nela atribuindo o IP 10.0.0.201 logo tornando-se responsável pela resposta a esse IP.

pq A replicação da partição é feita pelos IPs 10.0.0.200 e 10.0.0.4.

No que diz respeito ao proxy, o que o servidor faz é somente redirecionar os pacotes vindo até ele para o IP 192.168.200.254.

O servidor secundário além de ficar monitorando o servidor primário, também tem a finalidade de ficar monitorando o roteador, cujo IP é 192.168.200.254, utilizando para isso a segunda interface de rede que tem o IP 192.168.200.2.

6.2 Drbd

Nesta sessão será demonstrado como é feito a replicação dos dados entre o servidor primário e o servidor secundário e para isso, será necessário elucidar melhor como foi feito a estruturação dos diretórios.

O disco local foi particionado em 3 partições: hda1, hda2 e hda3, onde na hda1 é onde ficou o sistema operacional como um todo, na hda2 ficou a pasta /home e na hda3 fica a memória swap.

O diretório de configuração do Apache, que fica em /etc/apache, foi movido para dentro do diretório /home, onde em seu lugar padrão foi criado um link simbólico, com isso mantendo o padrão de localização dos arquivos de configuração do sistema.

Ex: ln -s /home/apache /etc/apache

6.2.1 Configuração

A configuração nos dois servidores é monstrada na figura x.

6.2.2 Inicialização

Para inicializar a replicação basta executar o comando `/etc/rc.d/drbd start` nos dois servidores.

```
resource drbd0 {
  protocol C;
  incon-degr-cmd "halt -f";
  startup {
    degr-wfc-timeout 20;
  }
  disk {
   on-io-error detach;
  }
  net {
   on-disconnect reconnect;
  }
  syncer {
   rate 15M;
   group 1;
   al-extents 257;
  }
  on nd {
     device   /dev/drbd0;
     disk     /dev/hda2;
     address  10.0.0.200:7789;
     meta-disk internal;
  }
  on servidor3 {
     device   /dev/drbd0;
     disk     /dev/hda2;
     address  10.0.0.4:7789;
     meta-disk internal;
  }
}
```

Figura 6.6: Configuração do drbd.conf

6.3 Heartbeat

6.3.1 Configuração

Esta sessão tem como finalidade mostrar um exemplo de configuração do Heartbeat, lembrando que os arquivos devem ser idênticos nos dois servidores e que os serviços a serem utilizados serão a réplica de disco e o servidor de páginas.

ha.cf

Arquivo localizado em /etc/ha.d/ha.cf

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility    local0
keepalive 2
deadtime 5
warntime 10
initdead 120

auto_failback on

bcast eth0

node nd
node servidor3
```

Figura 6.7: Configuração do ha.cf

haresources

Arquivo localizado em /etc/ha.d/haresources

```
nd IPaddr::10.0.0.201 drbddisk httpd
```

Figura 6.8: Configuração do haresource

authkeys

Arquivo localizado em /etc/ha.d/authkeys

```
#
auth 1
1 crc
#2 sha1 HI!
#3 md5 Hello!
```

Figura 6.9: Configuração do authkeys

Após a configuração deste arquivo não esquecer de mudar as permissões com `chmod 600 authkeys`

6.3.2 Inicialização

Para inicializar o Hearbeat digite `/etc/rc.d/heartbeat start` , deve ser levado em conta que é interessante sempre levantar o servidor principal e depois o de alta-disponibilidade.

6.4 Mon

6.4.1 Configuração

A configuração do Mon neste estudo de caso foi feito somente no servidor de alta-disponibilidade, dependendo da situação e da necessidade pode ser implementando em ambos os servidores.

mon.cf

Arquivo localizado em /etc/mon/

auth.cf

Arquivo localizado em /etc/mon/

6.4.2 Inicialização

Para executar o Mon digite `mon` na linha de comando e caso queira que o mesmo fique em modo background digite `mon &` .

6.5 Teste no sistema

Para validar a configuração feita neste estudo de caso, foi realizados testes de simulação de falha no servidor nd onde o servidor 03 deve tomar todas as funções servidas pelo servidor nd.

A primeira fase deste teste foi levantar a replicação no servidor primário, que pode ser visto no apêndice A.1. e também no servidor secundário, que pode ser visto no apêndice A.2.

Quando executado o Drbd nos dois servidores, os dois até o presente momento são estabelecidos como secundário.

A segunda fase foi levantar o Heartbeat no servidor primário, que pode ser visto no apêndice A.3. e no servidor secundário que pode ser visto no apêndice A.4.

Feito isso pode ser visto que a interface virtual já e adicionada no servidor primário, bem como o serviço de página e ele é automaticamente configurado como servidor primário no Drbd.

A terceira fase foi executar o programa de monitoramento Mon para verificar o estado do roteador, pode ser visto no apêndice A.5.

Na Quarta fase é simulado a queda do servidor primário arrancando o cabo de rede, conforme pode ser visto no apêndice A.6.

Após retirado o cabo, o servidor secundário toma o lugar do servidor primário como pode ser visto no apêndice A.7

Já na Quinta fase o servidor primário retorna a sua funcionalidade, conforme pode ser visto no apêndice A.8. E o servidor secundário volta ao estado de monitoramento liberando os serviços para que voltem ao servidor primário, conforme visto no apêndice A.9.

E a Sexta fase foi a simulação da perda de comunicação do servidor secundário com o roteador, aonde o mesmo gerou um email de alerta, conforme pode ser visto em A.10.

Com isso pode ser confirmado a validade desse estudo de caso.

```
#
# Configurações Geral
#

cfbasedir   = /etc/mon/
alertdir    = /usr/lib/mon/alert.d/
mondir      = /usr/lib/mon/mon.d/
maxprocs    = 20
histlength = 100
randstart = 60s

#
# Tipo de autenticação
#

authtype = getpwnam

#
# Definidos os grupos
#

hostgroup roteador 192.168.200.254

hostgroup www 10.0.0.200

#
# Regras dos grupos
#

# Servidor WWW

watch www
   service ping
      interval 2m
      monitor fping.monitor
      allow_empty_group
      period wd {Sun-Sat}
         alert mail.alert nd@nide.com.br
         alertevery 45m
   service http
      interval 4m
      monitor http.monitor
      allow_empty_group
      period wd {Sun-Sat}
         alert mail.alert nd@nide.com.br
         upalert mail.alert -S "Servidor WEB esta desligado" mis
         alertevery 45m
```



```
# Roteador

watch roteador
   service ping
      interval 1m
      monitor fping.monitor
      period wd {Sun-Sat}
         alert mail.alert nd@nide.com.br
```

```
#
# command section
#
command section

ack:          AUTH_ANY
checkauth:      all
clear:        AUTH_ANY
disable:       AUTH_ANY
dump:          AUTH_ANY
enable:        AUTH_ANY
get:          AUTH_ANY
list:         all
loadstate:      AUTH_ANY
protid:        all
quit:         all
reload:        AUTH_ANY
reset:        AUTH_ANY
savestate:      AUTH_ANY
servertime:     all
set:          AUTH_ANY
start:        AUTH_ANY
stop:         AUTH_ANY
term:         AUTH_ANY
test:         AUTH_ANY
version:       all

#
# trap section
#

trap section
```

Figura 6.11: Configuração do auth.cf

Capítulo 7

Conclusão

Como pôde ser visto nesta monografia é possível ter um sistema de alta disponibilidade baseado em ferramentas *open-source*, sem gastar muito dinheiro no quesito software.

Um fator que chama muito a atenção é que com a implementação de um servidor de alta disponibilidade em uma rede se tem um aumento na qualidade de serviço prestado pelo mesmo, onde o serviços prestado por um servidor fica o maior tempo possível disponível ao usuário, valendo lembrar que cada minuto que um servidor fica fora é dinheiro perdido.

O mais satisfeito foi o potencial e a maturidade que o sistema operacionais Linux possui hoje. Ele pode ser colocado como sendo um servidor de missões criticas tanto de uma pequena empresa quanto de uma grande empresa, sem deixar a desejar nada das outras soluções concorrentes.

Um fator muito importante que se deve levar em consideração durante a implantação de um servidor de alta disponibilidade é justamente o custo x beneficio, como pode ser visto, quando maior o grau de disponibilidade maior será o recurso financeiro necessário para implementação de tal, porem tudo isso depende do quanto importante é a sua informação estar disponível para os usuários.

Como pode ser visto, os resultados obtidos com o teste dos três *software* foram muito satisfatórios, pos de uma forma barata, foi possível resolver o problema de replicação, bem como a de monitoramento dos servidores, fator este que tornou essas ferramentas muito atrativas, justamente pelo recurso que as mesmas proporciona.

Um dos fatores que chamou a atenção durante a implantação da alta disponibilidade foi justamente na parte das paradas obrigatórias para manutenção, valendo lembrar que quando isso ocorre se trabalha com a manutenção preventiva onde se trabalha com a prevenção e não com a manutenção corretiva que se trabalha

com a correção de uma falha após ela ter surgido, pos dessa forma é possível estar parando um servidor sem interromper os serviços prestado e muito menos ter que fazer horas extras para não estar atrapalhando a produtividade da empresa.

Inicialmente a alta disponibilidade parece ser muito difícil de ser alcançada, pois para atingir tal meta, tem que se levar em consideração vários fatores, como por exemplo, a redundância de fonte de energia, de equipamentos e até a própria redundância das informações e programas se analisado no ponto financeiro, isso seria um grande desperdício de dinheiro, mas no decorrer desta monografia, percebe-se completamente o contrário, de tal forma que o desperdício de recurso se convertia em investimento, em qualidade de serviço, coisa ao qual é essencial a uma empresa.

Outro problema interessante foi como que um problema de disponibilidade pode ser resolvida de uma forma tão simples, sem a necessidade de nenhum equipamento caro e de arquitetura proprietária.

E por fim, nada melhor que se ter um servidor de alta disponibilidade, principalmente quando o disco rígido do servidor queira e não ter mais que escutar as famosas frases do chefe, como por exemplo: *"Quanto tempo o servidor demora a voltar, 5 minutos?"*

Referências Bibliográficas

BONAN, A. R. *Configurando e usando o Sistema Operaciona Linux*. 2. ed. São Paulo: Futura, 2003.

FILHO, N. A. P. *Linux, Clusters e Alta Disponibilidade*. Dissertação (Mestrado) — Universidade de São Paulo, 2002.

GARCIA, S. *Alta Disponibilidade (High Availability) em sistemas GNU/LINUX*. 2003. Http://ha.underlinux.com.br/.

GUNDA, B.; TECHNOLOGIES, O. S. *High Availability and Disaster Recovery Strategies White Paper*. 2001. Revison A.

MARCUS, E.; STERN, H. *Blueprints for High Availability*. 2. ed. Rio de Janeiro: Wiley Publishing, 2003.

NORTON, P.; GRIFFITH, A. *Guia completo do Linux*. 1. ed. São Paulo: Berkeley Brasil, 2000.

PITANGA, M. *Computação em Cluster*. 1. ed. Rio de Janeiro: Brasport, 2003.

RUSSEL, S. et al. *High Availability Without Clustering*. 1. ed. março: IBM, 2001.

SZTOLTZ, L.; TEIXEIRA, R. S.; RIBEIRO., E. de O. F. *Entendendo o Conectiva Linux*. 2003. Http://www.conectiva.com.br.

Apêndice A

Ferramentas - Configuração

A.1 Drbd

O arquivo de configuração do Drbd é o drbd.conf que se localiza em /etc/drbd.conf.

A estrutura de configuração desenvolvida pela equipe do Drbd foi feita de uma forma onde se tenha que editar somente um vez o arquivo de configuração e copiá-lo a todos os nodes que farão parte da replicação.

Exemplo de configuração do drbd.conf:

```
resource drbd0 {

         protocol B;

         incon-degr-cmd "halt -f";

         on servidor1 {
            device    /dev/drbd0;
            disk      /dev/hda2;
            address   10.0.0.2:7789;
       }

         on servidor3 {
            device    /dev/drbd0;
            disk      /dev/hda2;
            address   10.0.0.4:7789;
         }
}
```

Esse foi um exemplo de uma configuração de uma replicação chamada drbd0 que utiliza o protocolo B[1] como método de confirmação entre os dispositivos.

O primeiro dispositivo está no servidor1 em /dev/drbd0, que na verdade corresponde a partição /dev/hda2.

Já o IP é usado para especificar qual interface de rede ele usará.

Poderá haver vários resources no arquivo de configuração caso deseja-se ter mais de uma replicação de disco.

A.1.1 Formato do arquivo

O arquivo consiste em sessões e parâmetros. As sessões iniciam com uma palavra-chave, várias vezes associada a um nome e um abre chaves '{' e um fecha chaves '}' que tem como finalidade fechar o parâmetros.

sintaxe: sessão [nome] parâmetro valor; ...

O parâmetro inicia com o identificador de parâmetros seguido por um espaço em branco e o valor subseqüente é considerado como um valor que faz parte do parâmetro.

Em alguns casos especiais os parâmetros booleanos consistem somente de um identificador.

Para finalizar o parâmetro utilize o ';'

Sessão

skip
Comentários fora do texto, sempre tendo mais de uma linha.

global
Configuração do parâmetro global. Usa somente *minor-count*,*dialog-refresh*, e *disable-io-hits* são permitidos nesta sessão. Pode-se ter somente um sessão global e preferencialmente ela sendo a primeira sessão do arquivo de configuração.

resource *name*
Configura os resource Drbd. A sessão de resource necessita de duas sessões *"on host"* e pode ter uma sessão *"startup"*, uma *"syncer"*, uma *"net"* e uma *"disk"*.
Os seguintes parâmetro é requerido: *Protocol*
Parâmetros opcionais: *incon-degr-cmd*.

[1] Este protocolo especifica que uma operação de escrita é considerada completa, se está é escrita no disco local e enviado pela rede

on *host-name*
Chaves são parâmetros de configuração necessários para fazer o fechamento de um dispositivos Drbd, o host-name é o nome da máquina, que pode ser conseguida com o comando *uname -n*.
Parâmetros requeridos: *device, disk,address,meta-disk*.

disk
Esta sessão é usada para uma configuração precisa das propriedades do Drbd no que diz respeito a localização do disco a baixo nível.
Parâmetros opcionais: *on-io-error*

net
Esta sessão é utilizada para a configuração das propriedades da rede.
Parâmetros opcionais: *sendbuf-size,timeout, connect-int, ping-int,max-buffers,max-epoch-size,ko-count,on-disconnect*.

startup
Esta sessão é utilizada para configurar a inicialização do Drbd. Para mais referências sobre os parâmetros, veja em drbdsetup.
Parâmetros opcionais:*wfc-timeout,degr-wfc-timeout*.

syncer
Esta sessão é utilizada para configurar o daemon de sincronismo para um dispositivo. Para mais referencias sobre os parâmetros, veja em drbdsetup.
Parâmetros opcionais:*rate,group,al-extents*.

Parâmetro

minor-count *count*
count pode ter um número entre 1 e 255.
Use minor-count se desejar definir o número de resoucers apos ter recarregado o módulo Drbd. Por padrão o módulo lê exatamente o número de device configurado no arquivo de configuração.

dialog-refresh *time*
O time pode ser o valor 0 ou um número positivo.
O uso do dialog redesenha o contador de segundo a cada segundo especificado (ou não se o valor for 0). O valor padrão para ele é 1.

disable-io-hints
Use o disable-io-hints se quiser configurar o drbd utilizando o endereço de rede loopback ou entre duas máquinas virtuais.

protocol *prot-id*
Em um link TCP/IP a especificação do protocolo deve ser usado. As especificações válidas dos protocolos são A, B e C.

> **O Protocolo A:**
> Uma operação de escrita é considerada completa, se está é escrita no disco local e enviado pela rede.
>
> **O Protocolo B:**
> Uma operação de escrita é considerada completa, se está é escrita no disco local e confirmado a recepção no disco remoto.
>
> **O Protocolo C:**
> Uma operação de escrita é considerada completa, se chegar a confirmação do disco local e do remoto.

incon-degr-cmd *command*
Caso um dos nós, durante a sua inicialização, inicialize na modalidade degradado e a replicação dos dados local é inconsistente, este comando é executado. Se o comando não retornar erro, o drbddisk espera o device Drbd ir para o estado primário.

device *name*
O nome do nó de um dispositivo de bloco do recurso é especificado nesta opção. Poderá ser usado o device como sendo uma aplicação (sistema de arquivo) ou não poderá usar o dispositivo de block de baixo nível que é especificado no parâmetro de disco.

disk *name*
O Drbd usa realmente este dispositivo de bloco para armazenar e recuperar as informações. Nunca utilize este dispositivo quando Drbd estiver rodando pos pode acabar danificando os arquivos caso algum opção seja passada de forma errada.

address *IP:port*
O recurso precisa de um endereço IP para o dispositivo, que é usado para esperar as conexões com o outro dispositivo.
Cada recurso do Drbd necessita de uma porta TCP para ser usada durante a conexão com o outro dispositivo. Dois dispositivos Drbd diferentes não podem usar a mesma combinação de IP e Porta.

meta-disk *internal*
Está opção já fala por si so.

meta-disk *device* [*index*]
Internamente, os últimos 128 MB de um dispositivo são usados para gravar os meta-data. Está opção não poderá ser usado [index] quando utilizado a opção interna descrita acima.
Com essa opção é possível gravar a informação de meta-data de dois dispositivos em apenas um.

on-io-error *handler*
O handler é verificado, e se o dispositivo de baixo nível reportar um erro de I/O[2] a ele a opção será executado.
O handler poderá ser pass_on, panic ou detach.

> **pass_on:**
> Reporta um erro de io para o programa. Quando primário o relatório é sobre o sistema de arquivo montado. Já no secundário isso é ignorado.
>
> **panic:**
> Este opção leva o nó Cluster a receber um kernel panic.
>
> **detach:**
> O nó desativa o dispositivo de baixo nível e continua com um disco a menos.

sndbuf-size *size*
O size é o tamanho do buffer de envio do pacote TCP. O valor padrão é 128k.

timeout *time*
Se o no falhar ao enviar a resposta esperada no tempo de décimos de segundo, o nó sócio é considerado morto e conseqüentemente a conexão TCP/IP é abandonada. Isso deve ser mais baixo que as opções connect-int e ping-int. O valor padrão é 60 = a 6 segundos, a unidade é medida em 0.1 segundos.

connect-int *time*
Caso não for possível conectar ao device Drbd remoto imediatamente, o DRBD tenta conectar novamente. Com essa opção é possível configurar o tempo entra as tentativas. O valor padrão é de 10 segundos e a unidade de medida é em 1 segundo.

ping-int *time*
Se a conexão que liga o par de dispositivos Drbd for inativa mais do que o temo expresso em segundos, o Drbd irá gerar um pacote keep-aliva para

[2] Isto corresponde a um processo de escrita/leitura.

verificar se o seu sócio ainda esta vivo. O valor padrão é 10 segundos, e a unidade de medida é 1 segundo.

max-buffers *number*
Número máximo de requisições alocada pelo Drbd. A unidade do tamanho da pagine é de 4 KB na maioria dos sistemas.

max-epoch-size *number*
O maior número de blocos de dados entre duas barreiras de escritas. Se ajustada para um valor menor que 10, poderá diminuir o desempenho.

ko-count *count*
No caso do segundo nó falhar completamente um único pedido de escrita por um tempo contado de intervalos de parada, ele está expulso do conjunto (Caso for o no primário ele irá para o modo StandAlone). O valor padrão é 0, o que não ativa esta opção.

on-disconnect *handler*
Quando a conexão entre o par é perdida, o Drbd pode entrar na modalidade Stand_Alone, tentando reconectar com o outro par ou então parando todos os pedidos de IO. As opções válidas para essa opção é stand_alone, reconnect e freeze_io. O valor padrão do handler é reconnect.

stand_alone:
Não reconecta e vai para o estado StandAlone.

reconnect:
Tenta reconectar

freeze_io:
Tenta reconectar, mas para todo IO até que a conexão sejá refeita.

wfc-timeout *time*
Espera que a conexão caia. O script de blocos Drbd é obstruído para não carregar o processo até que os recursos do Drbd estejam conectados. Isso acontece quando o gerente do conjunto de Cluster inicia depois.
Caso seja necessário limitar o tempo de espera, isso será feito aqui. O valor padrão é 0, ou seja, ilimitado. A unidade é medidas em segundos.

degr-wfc-timeout *time*
Espera que a conexão caia, se este node for um degraded Cluster , caso o degraded Cluster sejá reinicializado, o valor do time-out será o valor especificado aqui em vez do wfc-timeout. O valor padrão é de 60 e a unidade é tempo é medida em segundos. O valor 0 corresponde a ilimitado.

rate *rate*
Serve para controlar as operações feitas pelo Drbd, nesta opção é possível limitar a largura da banda que é usada para uma sincronização em modo background. O valor padrão é de 250 KB/sec, e a unidade de medida é KB/sec. Os sufixos opcionais permitidos são: K, M, G.

group *number*
Ressincronização de todos os dispositivos de um grupo paralelo. Os grupos são colocados em séria e na ordem crescente. O valor padrão do grupo é 0. Isso é padrão para todos os dispositivos sincronizados em paralelo. Valor positivos e negativos são permitidos.

al-extents *extents*
O Drbd automaticamente faz a detecção da hot área. Com este parâmetro é possível controlar o inicio do tamanho da hot área (= active set). Cada extensão de armazenamento marca 4M. Caso o no primário saia do Clustes inesperadamente as áreas cobertas pela active ser vão ressincronizar tornando a reunir o no que ocorreu a falha.
A estrutura de dados é armazenada na área do meta-data, conseqüentemente cada mudança da active set é necessário reescrever a operação no dispositivo meta-data. Um alto número da extents dá um tempo mais elevado para a ressincronização, porém menor para a atualização do meta-data. O número padrão é 127 onde o mínimo é 7 e o maior 3843.

A.2 Heartbeat

O Heartbeat possui basicamente três arquivos de configurações que são: ha.conf, haresources e authkeys que serão abordados mais abaixo.

ha.conf

Será abordado uma lista de opções para o arquivo de configuração . O que terá que ser feito é configurar as opções deste arquivo, como por exemplo, os nós, no que diz respeito ao método de verificação, poderá ser feito através da serial, broadcast, multicast e unicast, além disso deve ser especificado o valor do auto_failback.

debugfile */var/log/ha-debug*
Está opção indica onde será escrito as mensagens de debugação.

logfile */var/log/ha-log*
Está opção especifica o local do arquivo de log.

logfacility ***local0***
Facilidade de uso para syslog/logger.

keepalive ***2***
Tempo de vida entre os heartbeats, a unidade é medida em 10 mini segundos, porém pode ser especificados em unidade de milesegundos: 1500ms

deadtime ***30***
Está opção indica quanto tempo o host é declarado como morto.

warntime ***10***
Quanto tempo após deve ser usado "o hearbeat atrasado"de perigo.

initdead ***120***
Verifica a primeira vez que morreu. Deve ser pelo menos duas vezes o tempo de deadtime.

udpport ***694***
Especifica a porta udp que será usada para broadcast e unicast

baud ***19200***
Velocidade da porta serial.

serial ***/dev/ttyS0***
Localização da porta serial.

bcast ***eth0*** **ou**
bcast ***eth1 eth2***
Esta opção indica qual interface deve ser enviada os hearbeats.

mcast ***eth0 225.0.0.1 694 1 0***
configura o multicast da seguinte forma, mcast [device] [mascara] [porta] [ttl] [loop]

ucast ***eth0 192.168.1.2***
Configura um unicast da seguinte forma, ucast [device] [ip]

auto_failback ***on***
Está opção determina se o resource retornar ela ira automaticamente a node primário . Opções: *on* = ativa, *off* = desativa e *legacy* = Ativa automaticamente o *failback* no sistema quando todos os nos não suportarem está opção. O valor padrão é legacy. Está opção é equivalente a antiga *nice_failback*.

stonith ***baytech /etc/ha.d/conf/stonith.baytech***
Está diretriz assume que há um conjunto de dispositivos stonith e sua forma de usar é stonith [tipo] [arquivo]

stonith_host **** baytech 10.0.0.3 mylogin mysecretpassword*** **ou**
stonith_host ***ken3 rps10 /dev/ttyS1 kathy 0*** **ou**
stonith_host ***kathy rps10 /dev/ttyS1 ken3 0***
com está opção é possível configurar multiplos stonith, a forma de se usar é stonith_host [hostfrom] [tipo_stonith] [parametro]. Para pegar a lista de stonith suportados digite stonith -L e para mais informações digite stonith -h.

watchdog ***/dev/watchdog***
watchdog é o temporizador do ”Cão de guarda”. Ele funciona assim, se o próprio coração não bater por alguns minutos, a máquina irá reinicializar. Se for usar o software watchdog, terá que ser carregado o modulo com o parâmetro ”nowayout=0” ou compila-lo com a opção *CONFIG_WATCHDOG_NOWAYOUT*.

node ***ken3*** **ou**
node ***kathy***
está opção especifica quais nós fazem parte do Cluster, o nome pode ser encontrado com o comando uname -n.

ping ***10.10.10.254***
Configuração de membro de um pseudo-Cluster.

ping_group ***group1 10.10.10.254 10.10.10.253***
São a configuração de grupo de membros de um pseudo-Cluster.

hbaping ***fc-card-name***
São pings destinado a canais de fibra.

respawn ***userid /path/name/to/run*** **ou**
respawn ***hacluster /usr/lib/heartbeat/ipfail***
Processo de inicio e parada do heartbeat. Reinicia quando sai com valor rc=100.

apiauth ***client-name gid=gidlist uid=uidlist*** **ou**
apiauth ***ipfail gid=haclient uid=hacluster***
são controle a api por usuários.

hopfudge ***1***
configura o número máximo de pulos entre os nos.

deadping *30*
tempo de vida dos ping dos nos.

hbgenmethod *time*
Método de criação de generation heartbeat. Normalmente estes são armazenados no disco e incrementados quando necessário.

realtime *off*
Ativa ou desativa a execução em tempo real, o valor padrão é *on*.

debug *1*
Configura o nível de depuração, valor padrão é 0.

apiauth *ipfail uid=hacluster* ou
apiauth *ccm uid=hacluster* ou
apiauth *cms uid=hacluster* ou
apiauth *ping gid=haclient uid=alanr,root* ou
apiauth *default gid=haclient*
configura a autenticação API. o padrão para ipfail e ccm é uid=HA_COMUSER e para ping e cl_status é gid=HA_APIGROUP

msgfmt *classic/netstring*
formato da mensagem . Valor padrão: *classic*.

use_logd *yes/no*
Deve ser logado o daemon, valor padrão *no*.

conn_logd_time *60*
O tempo de log nos intervalo de reconexão no caso de uma conexão falhar. Valor padrão 60 segundos.

haresources

Após configurado o arquivo ha.conf, este arquivo deve ser configurado com os nodes do arquivo anterior. Porém a principal finalidade deste arquivo é estar especificando quais serviços devem ser verificados.

Este arquivo deve ser igual em todos os nós.

Está é uma lista dos recursos que se movem de máquina para máquina enquanto os conjuntos de nós caem ou levantam. Não inclua "endereços administrativos"

A forma de se utilizar é *node-name resource1 resource2 ... resourceN serviços1 serviços2..... serviçosN*

Exemplos:

```
just.linux-ha.org 135.9.216.110

just.linux-ha.org 135.9.216.110 http

just.linux-ha.org 135.9.216.110 135.9.215.111 135.9.216.112 httpd

just.linux-ha.org 135.9.216.3/28/eth0/135.9.216.12 httpd

node1 10.0.0.170 Filesystem::/dev/sda1::/data1::ext2
```

authkeys

Este é o arquivo de autenticação. A permissão dele deve ser 600

A forma dele é a seguinte *auth* <*metodo*> onde auth indica qual a forma de autenticação.

Os métodos válidos são: crc, sha1 e md5.

crc
método que utiliza menos recurso da rede.

sha1
melhor autenticação sem se preocupar com os recursos da CPU.

md5
No caso da rede ser insegura e gostar de muita segurança.

exemplo:

```
auth 2

1 crc

2 sha1 HI!

3 md5 Hello!
```

A.3 Mon

mon.cf

O arquivo de configuração consiste em nenhuma ou mais definições de *hostgroup*, e uma ou mais definições de *watch*. Cada definição de *watch* pode ter uma ou mais

definições de serviços. Uma linha começa com um espaço em branco e um ”#” que é reservado a comentários e são ignorados.

As linhas são analisadas gramaticalmente enquanto são lidas. as linhas longas podem conter terminadores com os backslash ” \ ”. Se uma linha continuar e o espaço em branco principal da próxima linha for removido, no final será considerado como somente uma única linha.

Abaixo segue um exemplo de configuração do arquivo mon.cf

Opções Globais

cfbasedir = /usr/lib/mon/etc
Localização dos arquivos de configurações.

alertdir = /usr/lib/mon/alert.d
Localização dos arquivos de alerta.

mondir = /usr/lib/mon/mon.d
Localização dos arquivos do monitoramento do Mon.

maxprocs = 20
Número máximo de processos.

histlength = 100
Tamanho da lista do histórico.

randstart = 60s
Tempo de inicialização.

Tipos de Autenticação

authtype = getpwnam

getpwnam = Senhas padrão do UNIX, não para senhas no formato "shadow"[3].

shadow = Senhas padrão shadow do UNIX, não implementado.

userfile = "mon"user file.

Definição de Grupo

hostgroup *hostname ou endereços IP*
exemplos:

[3] xxx

hostgroup serversbd1 dns-yp1 foo1 bar1

hostgroup serversbd2 dns-yp2 foo2 bar2 ola3

hostgroup routers cisco7000 linuxrouter agsplus

hostgroup hubs cisco316t hp800t ssii10

hostgroup workstations blue yellow red green cornflower violet

hostgroup netapps f330 f540

hostgroup wwwservers www

hostgroup printers hp5si hp5c hp750c

hostgroup new nntp

hostgroup ftp ftp

Seguem alguns casos de configuração das watch

```
watch serversbd1

 service ping
 description ping servers in bd1
 interval 5m
 monitor fping.monitor
 period wd {Mon-Fri} hr {7am-10pm}
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com
     alertevery 1h
 period NOALERTEVERY: wd {Mon-Fri} hr {7am-10pm}
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com
 period wd {Sat-Sun}
     alert mail.alert bofh@domain.com
     alert page.alert bofh@domain.com
    service telnet
 description telnet to servers in bd1
 interval 10m
 monitor telnet.monitor
```

```
 depend serversbd1:ping
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com

begingroup
watch serversbd2
    service ping
 description ping servers in bd2
 interval 5m
 monitor fping.monitor
 depend routers:ping
 period wd {Mon-Fri} hr {7am-10pm}
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com
     alertevery 1h
 period wd {Sat-Sun}
     alert mail.alert bofh@domain.com
     alert page.alert bofh@domain.com
    service telnet
 description telnet to servers in bd2
 interval 10m
 monitor telnet.monitor
 depend routers:ping serversbd2:ping
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com

watch mailhost
    service fping
 period wd {Mon-Fri} hr {7am-10pm}
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com
     alertevery 1h
    service telnet
```

```
 interval 10m
 monitor telnet.monitor
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com
    service smtp
 interval 10m
 monitor smtp.monitor
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert page.alert mis-pagers@domain.com
    service imap
 interval 10m
 monitor imap.monitor
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert page.alert mis-pagers@domain.com
    service pop
 interval 10m
 monitor pop3.monitor
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert page.alert mis-pagers@domain.com

watch wwwservers
    service ping
 interval 2m
 monitor fping.monitor
 allow_empty_group
 period wd {Sun-Sat}
     alert qpage.alert mis-pagers
     alertevery 45m
    service http
 interval 4m
```

```
 monitor http.monitor
 allow_empty_group
 period wd {Sun-Sat}
     alert qpage.alert mis-pagers
     upalert mail.alert -S "web server is back up" mis
     alertevery 45m
    service telnet
 monitor telnet.monitor
 allow_empty_group
 period wd {Mon-Fri} hr {7am-10pm}
     alertevery 1h
     alertafter 2 30m
     alert mail.alert mis@domain.com
     alert page.alert mis-pagers@domain.com
```

Se os roteadores não estiverem comunicável, será enviado uma mensagem utilizando a linha telefônica ou o Protocolo IXO. Falhas no roteadores da rede são coisas sérias, então eles têm que ser checado a cada 2 minutos.

```
watch routers
    service ping
 description routers which connect bd1 and bd2
 interval 1m
 monitor fping.monitor
 period wd {Sun-Sat}
     alert qpage.alert mis-pagers
     alertevery 45m
 period LOGFILE: wd {Sun-Sat}
     alert file.alert -d /usr/lib/mon/log.d routers.log
```

Se o mon não conseguir verificar um dos hubs, o usuário será chamado através de uma mensagem no pager.

```
watch hubs
    service ping
 interval 1m
 monitor fping.monitor
 period wd {Sun-Sat}
     alert qpage.alert mis-pagers
     alertevery 45m
```

Monitor de espaço livre em disco em um servidor NFS. Quando o espaço estiver entre 5 MB, será enviado um email. Monitores terminados com ”;;”não serão executados em um grupo de host adicionado á linha de comando.

```
watch netapps
    service freespace
     interval 15m
 monitor freespace.monitor /f330:5000 /f540:5000 ;;
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
\#  alert delete.snapshot
     alertevery 1h
```

Estações de rede

```
watch workstations
    service ping
 interval 5m
 monitor fping.monitor
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
     alertevery 1h
```

Servidor de NEWS

```
watch news
    service ping
 interval 5m
 monitor fping.monitor
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
     alertevery 1h
    service nntp
 interval 5m
 monitor nntp.monitor
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
     alertevery 1h
```

Impressora HP

```
watch printers
```

```
   service ping
 interval 5m
 monitor fping.monitor
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
     alertevery 1h
   service hpnp
 interval 5m
 monitor hpnp.monitor
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
     alertevery 1h
```

Servidor FTP

```
watch ftp
    service ftp
 interval 5m
 monitor ftp.monitor
 period wd {Sun-Sat}
     alert mail.alert mis@domain.com
     alertevery 1h
```

Servidor de terminal dial-in.

```
watch dialin
    service 555-1212
        interval 60m
        monitor dialin.monitor.wrap -n 555-1212 -t 80 ;;
        period wd {Sun-Sat}
            alert mail.alert mis@domain.com
            upalert mail.alert mis@domain.com
            alertevery 8h
    service 555-1213
        interval 33m
        monitor dialin.monitor.wrap -n 555-1213 -t 80 ;;
        period wd {Sun-Sat}
            alert mail.alert mis@domain.com
            upalert mail.alert mis@domain.com
            alertevery 8h
```

Definições

monitor
É um programa que testará uma determinada condição, retornando um valor verdadeiro ou falso, produzindo opcionalmente uma saída a ser passada ao programa scheduler. Os monitores normalmente detectam um host inalcansável através de um pacote ”echo ICMP” ou através de uma conexão a um serviço TCP.

period
Indica um período de tempo onde é interpretado o módulo Time::Period.

alert
É um programa que envia mensagem quando invocoda pelo scheduler. O scheduler chama um alerta quando detecta que uma falha de um monitor ocorreu. O programa de alerta aceita uma argumentos de linha de comando do scheduler, além de dados através de entrada padrão.

hostgroup
É um único host ou uma lista de hosts, especificado com nomes ou endereços IP

service
É uma coleção de parâmetros usado para tratar um monitoramento em particular que fosse fornecido por um grupo. Os serviços são modelados geralmente após verificar o servidor SMTP, o ECHO ICMP, a disponibilidade do espaço em disco ou um evento SNMP.

watch
É uma coleção de serviços que se aplica a um grupo em particular.

Operações

Quando o scheduler do Mon é inicializado, a primeira coisa feita é ler o arquivo de configurações, assim determinado os serviços que necessitam ser monitorados.

O Arquivo de configuração normalmente fica em /etc/mon.cf e pode ser especificado com a opções -c *parâmetro*. Se a opção -M for especificada, o arquivo de configura será pre-processado com o m4. Se o arquivo de configuração já estiver no formato m4, ele ira processá-lo automaticamente sem a necessidade de especificar o parâmetro.

O sheculer entra no laço de conexões do cliente, invocando os monitores e alertando sobre falhas. Cada serviço tem um temporizador que pode ser especificado no arquivo de configuração como variável de intervalo, o que diz ao scheduler com qual freqüência invocar um processo de monitoramento. O temporizador do scheduler pode ser parado temporariamente.

Programas monitores

Os processos do monitor são invocados com os argumento especificados no arquivo de configuração, adicionando o hosts do grupo da aplicação. Por exemplo, se o grupo watch for "server", que contenha os *hostnames* "smtp", "nntp"e "ns", o monitor lê as linhas da seguinte forma, monitor fping.monitor -t 4000 -r 2 e adiciona os *hostnames*.

O ”fping.monitor” será executado conforme o parâmetro abaixo:

MONITOR_DIR/fping.monitor *-t 4000 -r 2 smtp nntp ns*

O MONITOR_DIR é o caminho atual a ser pesquisado, por padrão /usr/local/lib/mon/mon.d e /usr/lib/mon/mon.d, mas pode ser modificado com a opção -s ou no arquivo de configuração.

Se todos os *hosts* de um *hostgroup* forem desativados, um aviso será enviado ao *syslog* e o monitor não funcionara mais. Este comportamento pode ser modificado com a opção *allow_empty_group* na definição do serviço.

Se ao final da lina do argumento do "monitor"tiver ";;", a linha do host não será adicionada à lista de parâmetro.

MON_LAST_SUMMARY

A primeira linha da saída após a última vez que o monitor saiu.

MON_LAST_OUTPUT

A saída inteira do monitor da última vez que saiu.

MON_LAST_FAILURE

As duas últimas vezes da falha deste serviço.

MON_FIRST_FAILURE

As duas primeiras vezes que o serviço falhou.

MON_DESCRIPTION

A descrição do serviço

MON_DEPEND_STATUS

O status de dependência, ”0” se a dependência falhar.

MON_LOGDIR

O diretório do arquivo de log, como indicado na variável global de configuração *logdir*.

MON_STATEDIR

O diretório onde devem ser mantida os arquivos de estados.

"fping.monitor"deve retornar um status de saída 0 se terminar com sucesso, ou um valor diferente de zero se algum problema for encontrado. A primeira linha da saída do script do monitor tem um especial meaning: onde isto é usado para um breve sumário da falha detectada e logo passada ao programa de alerta. Toda saída restante é passada também ao programa de alerta, porem não tendo nenhuma interpretação requerida

Programas de alerta

Os programas de alerta são encontrado no diretório fornecido com o parâmetro -a, ou no diretório /usr/local/lib/mon/alert.d. Eles podem ser invocados com os seguintes parâmetro de comando de linha.

-s *service*
Tag de serviço para o arquivo de configuração.

-g *group*
Nome do grupo de host do arquivo de configuração.

-h *hosts*
Uma versão estendida do grupo de host, espaço limitado, mas contendo uma única palavra *shell*.

-l *alertevery*
Número de segundos para o próximo alerta ser enviado.

-O
Esta opção é fornecida a um alerta somente se o alerta está sendo gerado em conseqüência de um esperado *timing out*

-t time
o time de quando esta condição da falha foi detectada

-T
Esta opção é fornecida a um alerta somente se o alerta foi provocado por uma *trap*.

-u
Esta opção é fornecida a um alerta somente se ele esta sendo chamado por um *upalert*.

Os argumentes restantes são fornecido por parâmetros adquiridos do arquivo de configuração, após o serviço de parâmetro *alert*.

Como os programas de monitoramento, os programas de alerta são invocados com as variáveis ambiente definidas pelo usuário durante a definição do serviço, as opções abaixo são ajustadas explicitamente pelo servidor.

MON_LAST_SUMMARY
A primeira linha da saída após a última vez que o monitor saiu.

MON_LAST_OUTPUT
A saída inteira do monitor da última vez que saiu.

MON_LAST_FAILURE
As duas últimas vezes da falha deste serviço.

MON_FIRST_FAILURE
As duas primeiras vezes que o serviço falhou.

MON_LAST_SUCCESS
As duas últimas vezes que o serviço passou.

MON_DESCRIPTION
A descrição do serviço

MON_GROUP
O grupo watch que provocou este alarme.

MON_SERVICE
O headin do serviço que gerou este alerta.

MON_RETVAL
O valor da saída do programa monitor que falhou ou retorna um valor como aceito para o *trap*.

MON_OPSTATUS
Status operacional do serviço.

MON_ALERTTYPE
Esta opção tem os seguintes valores: "failure", "up", "startup", "trap"ou "traptimeout"que significam o tipo de alerta que foi provocado.

MON_LOGDIR
O diretório do arquivo de log, como indicado na variável global de configuração *logdir*.

MON_STATEDIR
O diretório aonde devem ser mantida os arquivos de estados.

A primeira linha da entrada padrão deve ser usada como um breve sumário do problema, normalmente fornecido como uma linha de um subjetc de um email ou

texto, sendo enviado a um pager. Os parâmetros usuais são uma lista de receptores para entrega de notificação. A interpretação dos receptores não é especificada e é até o programa de alerta.

Variáveis Globais

As seguintes variáveis podem ser ajustadas como valor padrão. A opção de linha de comando terá uma prioridade mais elevada nas definições.

alertdir = ***dir***
Indica o diretório onde estão os scripts de alerta. O valos padrão pode ser passado pela opção -a na linha de comando.

mondir = ***dir***
Indica o diretório onde estão os scripts monitor. O valor padrão pode ser passado pela opção -s na linha de comando.

statedir = dir
Indica o diretório de estado. Este diretório é utilizado para salvar as informações referentes ao estado.

logdir = ***dir***
Indica o diretório onde serão gravados os log.

basedir = ***dir***
Indica o caminho da base de diretório de estado, scripts e diretório de alerta.

cfbasedir = ***dir***
Indica o diretório onde ficam todos os arquivos de configuração.

authfile = ***file***
Indica o caminho do arquivo de autenticação.

authtype = ***type [type...]***
Tipo de autenticação utilizada.

userfile = ***file***
Este arquivo é usado quando o *authtype* for ajustado para userfile.

pamservice = ***service***
No caso do serviço PAM for usado para autenticar. Esta opção aceita somento o parâmetro ”pam”.

snmpport = ***portnum***
Configura qual porta SNMP a qual o usuário conectara.

serverbind = ***addr***
trapbind = ***addr***
Configura qual endereço que o *serverbind* e o *trapbind* irão utilizar.

snmp = **{yes|no}**
Ativa ou desativa o suporte a SNMP.

dtlogfile = ***file***
Indica o nome do arquivo que contem os registro do *downtime*.

dtlogging = ***yes/no***
Ativa ou desativa o log de *downtime*.

histlength = ***num***
Configura o número máximo dos eventos na lista do histórico. O valor padrão é 100. Esta opção pode ser ajustada com o parâmetro -k na linha de comando.

historicfile = ***file***
Se esta variável for configurada, os alertas serão logados em arquivos, e quando inicializado o serviço, algum ou todos os historicos serão carregados na memória.

historictime = ***timeval***
é a quantidade de arquivos de históricos carregados na memória durante a inicialização.

serverport = ***port***
Especifica o número da porta TCP que o servidor vai usar. O valor padrão é 2583

trapport = ***port***
Especifica a porta UDP que vai ser usada para transferir os *trap*. O valor padrão é 2583.

pidfile = ***path***
Indica o caminho do arquivo que vai ser gravado o pid. Esse valor pode ser alterado com o parâmetro -P na linha de comando.

maxprocs = ***num***
Indica o número máximo de processos

cltimeout = ***secs***
Configura o tempo de inatividade do usuário para dar *timeout*, valor em segundos.

syslog_facility = **facility**
Especifica a facilidade do syslog usada no login. o valor padrão é daemon.

startupalerts_on_reset = **{yes|no}**
Se configurado com ”yes” a inicialização dos alertar serão invocados pela linha de comando quando o cliente executar. O valor padrão é ”no”.

Sintaxe hostgroup

As entradas do *hosgroup* começam com a palavra chave ”*hostgroup*” e são seguidas por um *tag* do *hostgroup* seguido de um ou mais *hostname* ou endereço IP, separados por um espaço em branco.

A Tag do *hostgroup* deve ser composta de caracteres alfanuméricos, após a primeira que não estejam em branco são consideradas *hosts* pertencentes ao grupo. A definição do *hostgroup* termina com uma linha em branco.

Conforme o exemplo abaixo:

```
hostgroup servers nameserver smtpserver nntpserver nfsserver httpserver
smbserver

hostgroup router_group cisco7000 agsplus
```

Sintaxe de grupo watch

A entradas watch começam com a linha com a palavra chave *watch* seguido por um espaço em branco e por uma única palavra que normalmente referencia um *hostgroup* predefinido. Se a segunda palavra não for reconhecida como uma tag do *hostgroup*, um *hostgroup* novo será criado com esse nome.

As entradas watch consistem em uma ou mais definições de serviços.

Definições de serviço

service *servicename*
Uma definição de serviço começa com está palavra chave seguida por uma palavra que seja uma *Tag* para este serviço

interval *timeval*
A palavra chave *interval* seguida de uma valor especifica com que freqüência o scrip monitor irá disparar. Os valores são definidos como ”30s”. ”5m”, ”1h”, ”1d” , ”1.5h” este último representa uma hora e meia.

traptimeout *timeval*
Esta palavra chave faz exame do mesmo argumento de especificação *internal*, e faz o serviço esperar ao menos um trap de uma fonte externa. Isso é usado para o serviço *heartbeat-style*.

trapduration *timeval*
Se o trap for recebido dentro do tempo, o status do serviço do trap continuar normal, permanecendo constante.

randskew *timeval*
Coloca o *scheduler* do monitor de script para funcionar no início de cada intervalo, ele terá um ajuste aleatório durante o intervalo de tempo especificado pelo parâmetro.

monitor *monitor-name [arg...]*
A palavra chave monitor seguida por um nome de script especifica para o monitor roda-lo quando o tempo espirar.

allow_empty_group
A opção allow_empty_group permitira que um monitor seja invocado mesmo quando o *hostgroup* para essa *watch* esteja vazio por causa de um *host* desativado.

description descriptiontext
Um texto seguido de uma descrição é perguntada pelo programa cliente, passando aos alertas e aos monitores por uma variável de ambiente. Deve conter uma descrição breve do serviço, apropriada para a inclusão em e-mail ou em uma *web page*.

exclude_hosts *host* [***host...***]
Qualquer host listado após exclude_hosts será excluído da verificação do serviço

exclude_period *periodspec*
Não execute o monitor programado durante o tempo identificado pelo *periodspec*

depend *dependexpression*
A palavra chave *depend* é usada para especificar a expressão da dependência, seu valor pode ser verdadeiro ou falso, no senso booleanico. Dependendo a expressão Perl e de várias outras regras de sintaxe. Esta feature pode ser usada para controlar alerta de serviços que depende de outro serviço.

dep_behavior *{a|m}*
A avaliação da dependência gráfica pode ser controlada a suspensão de alertas ou monitoração invocadas.

Alert suppression
Se a opção for configurada como 'a', a expressão de dependência ira avaliar após o monitoramento do serviço ou após o *trap* ser recebido.

Monitor suppression
Se a opção for configurada como 'm', a expressão de dependência ira avaliar antes de um serviço ter sido executado.

Definições de período

Os períodos são usados para definir as circunstâncias que devem permitir que os alertas sejam enviados.

period [***label:***] ***periodspec***
Um *period* agrupa um ou mais alarmes e variáveis que controlam, com qual freqüencia um alerta acontece quando há uma falha. A palavra chave period tem duas formas. A primeira faz o exame do argumento que seja uma especificação do período do módulo do Perl 5 a Time::Period de Patrick Ryan. A segunda forma requer um nome para a especificação do period, como definido acima. Este nome é uma Tag que consistem em um caracter alfabético ou em um underscore seguido por zero ou mais indicadores alfanuméricos que terminados por dois pontos. Esta forma permite múltiplos periods com a mesma definição de period.

alertevery *timeval [observe_detail]*
A palavra chave alertevery faz um exame do mesmo tipo de argumento que a variável interval e limita o número de vezes que um alerta é emitido quando o serviço continua a falhar.

alertafter *num* ou
alertafter *num timeval*
alertafter *timeval*
A palavra chave alertafter tem três formas: a primeira onde tem somente o argumento "num", outra que tem os argumentos "num trimeval" e a terceira que tem "trimeval".
Na primeira forma, um alerta será invocado somente após a quantidade do número de falhas consecutivas especificadas em "num".
Na segunda forma, os argumentos é um valor inteiro seguido por um intervalo de tempo, funcionando da mesma forma que a anterior. Caso haja

falhas em um intervalo de tempo ela são; exemplo: 3 falhas em 30 minutos. E na terceira forma, o argumento é o interval, como descrito pela ítem anterior, os alertas para esse período serão chamados somente se o serviço estiver em um estado de falhas para mais que o tempo descrito pelo interval.

numalerts *num*
Esta variável diz ao servidor para chamar não mais que o ”num” de alertas durante uma falha.

no_comp_alerts
Se está opção for especificadas, os upalerts serão chamados sempre que o estado do serviço mudar de falho para sucesso.

alert *alert* [***arg...***]
Um period pode conter vários alertas, que são provocados por falhas de serviço. Um alerta é especificado com a palavra chave ”alert” seguido de um parâmetro de saída e pelos argumentos que são interpretador da mesma forma que na definição do Monitor.

failure_interval *timeval*
Ajusta o intervalo de verificação do timeval em que a verificação do serviço está falhando. Reinicia o intervalo original quando o serviço é bem sucedido.

upalert *alert* [***arg...***]
Um upalert é um complemento de um *alert*, um upalert é chamado quando os serviços fazem a transição do estado de falhar para sucesso, se corresponder a ”down” um alerta será previamente enviado.

startupalert *alert* [***arg...***]
um startualert é chamado somente quando o usuário do Mom inicia a execução.

upalertafter *timeval*
O parâmetro upalertafter é especificado com uma string de parâmetro de intervalo, como por exemplo ”30s”, ”1m” e etc e controles de disparo do upalert. Se um serviço voltar após estar caido por um momento maior do que o especificado nesta opção, um upalert será chamado.

MON TRAPPING

Mon tem a facilidade para receber um ”mon traps” especial de todas as máquinas ou de uma máquina remota. Atualmente, o único método disponível para enviar

um ”mon trap” e através da relação do classe Perl Mon::Client, embora o formato do pacote seja UDP ele é bem definido para permitir as escrita das traps em outras linguagens.

As traps são cabeçalhos semelhantes ao monitores, uma trap envia um status operacional, um texto de descrição e o Mon gera um alerta ou um upalert quando necessário. Elas podem ser travadas por qualquer grupo watch/service especificado no arquivo de configuração do Mon, porém é aconselhável configurar grupos watch/service especificamente para cada tripo de traps que se é esperado receber.

Quando definido um grupo especial de watch/service para traps, não inclua as diretrizes do ”monitor”, pois não é necessário invocar um monitor. Desde que um monitor não esteja sendo invocado, não é necessário que a definição watch tenha um hostgroup que contenha os nomes reais dos hosts. Somente faça o nome do usuário e o Mon automaticamente criará um grupo watch.

Abaixo segue um exemplo do arquivo de configuração:

```
watch trap-service
      service host1-disks
              description TRAP: for host1 disk status
              period wd {Sun-Sat}
                    alert mail.alert someone@your.org
                    upalert mail.alert -u someone@your.org
```

O Mon escuta as portas UDP para qualquer trap, é um padrão de facilidade disponível para handling traps aos grupos desconhecidos ou serviços.

Para ativar tal facilidade, deve ser incluído um grupo padrão ”watch” de serviços que deve conter as especificações dos alarms.

Se um grupo padrão ”watch” e seus serviços não forem configurados, traps desconhecidos serão logados via syslog e nenhum alarme sera enviado.

Caso queira mais facilidade, utilize o default/default para travar todas as traps. Seria mais indicado desabilitar os upalerts e usar as variável de ambiente MON_TRAP_INTENDED em scripts de alerta para fazer os alertas mais significativo.
Abaixo segue um exemplo padrão

```
watch default
      service default
              description Default trap service
              period wd {Sun-Sat}
                     alert mail.alert someone@your.org
                     upalert mail.alert -u someone@your.org
```

auth.cf

A especificação do arquivo para as variáveis de autenticação estão no arquivo de configuração, ou passado pelo parâmetro -A, que será lido na inicialização. Este arquivo define as restrições referentes ao comandos que os usuários podem executar.

O arquivo de autenticação é em modo texto que consiste em comentários, definições de comandos e parâmetros trap para autenticação.

As linhas em branco são ignoradas.

O arquivo é separado pela seção do comando e uma seção trap. As seções são especificadas por uma única linha que contem as seguintes formas:

command section

ou

trap section

A definição do comando consiste em um comando, seguido por dois pontos e por uma lista de usuários separado por vírgula.

O padrão é que nenhum usuário pode executar todos os comandos, a menos que for explicitamente permitido neste arquivo de configuração. Para maior detalhe, um usuário pode ser negado prefixando o seu nome com um ”!”. Se a palavra ”AUTH_ANY” for usada para um usuário, todos os usuário autenticado executarão o comando.

A seção trap permite a configuração de quais usuários podem enviar traps para um hosts. A sintaxe é um host fonte, que pode ser o nome ou o endereço IP, um espaço em branco, um usuário e uma senha em texto puro. Se um usuário for ”*”, então todos os traps de todos os hosts serão aceitos sem considerar o usuário e a senha. Se nenhum host ou senha for especificado, então nenhum trap será aceito.

Um exemplo do arquivo de configuração:

```
command section
list:           all
reset:          root,admin
loadstate:              root
savestate:              root

trap section
```

```
127.0.0.1       root    r@@tp4sswrd
```

Isto significa que todos os clientes podem executar a lista de comando, porém só o ”root” pode executar o ”reset”, ”loadstate” e ”savestate”. O ”admin” so pode executar o comando do ”reset”.

Arquivo de configuração de autenticação

Sintaxe: *comando: {user|all}*[*,user...*]

Abaixo segue um exemplo do arquivo de configuração

Sessão de comandos

```
ack:  AUTH_ANY
checkauth: all
clear:  AUTH_ANY
disable: AUTH_ANY
dump:  AUTH_ANY
enable:  AUTH_ANY
get:  AUTH_ANY
list:  all
loadstate: AUTH_ANY
protid:  all
quit:  all
reload:  AUTH_ANY
reset:  AUTH_ANY
savestate: AUTH_ANY
servertime: all
set:  AUTH_ANY
start:  AUTH_ANY
stop:  AUTH_ANY
term:  AUTH_ANY
test:  AUTH_ANY
version: all
```

Apêndice B

Log

Nesta sessão será abordado os log gerado por cada programa do estudo de casos

B.1 Drbd - Servidor ND

```
Jul 26 18:05:42 nd syslogd 1.4.1: restart.
Jul 26 18:05:43 nd kernel: klogd 1.4.1, log source =
/proc/kmsg started.
Jul 26 18:07:08 nd kernel: drbd: initialised. Version:
 0.7.10 (api:77/proto:74)
Jul 26 18:07:08 nd kernel: drbd: SVN Revision: 1743 build
by root@nd.nide.com.br, 2005-05-19 10:45:31
Jul 26 18:07:08 nd kernel: drbd: registered as block device
major 147
Jul 26 18:07:09 nd kernel: drbd0: resync bitmap: bits=366848
words=11464
Jul 26 18:07:09 nd kernel: drbd0: size = 1433 MB (1467392 KB)
Jul 26 18:07:09 nd kernel: drbd0: 0 KB marked
out-of-sync by on disk bit-map
Jul 26 18:07:09 nd kernel: drbd0: Found 3 transactions
 (3 active extents) in activity log.
Jul 26 18:07:09 nd kernel: drbd0: drbdsetup [5584]:
 cstate Unconfigured --> StandAlone
Jul 26 18:07:09 nd kernel: drbd0: drbdsetup [5597]:
cstate StandAlone --> Unconnected
Jul 26 18:07:09 nd kernel: drbd0: drbd0_receiver [5598]:
cstate Unconnected --> WFConnection
Jul 26 18:07:13 nd kernel: drbd0: drbd0_receiver [5598]:
cstate WFConnection --> WFReportParams
Jul 26 18:07:13 nd kernel: drbd0: Handshake successful:
```

```
DRBD Network Protocol version 74
Jul 26 18:07:13 nd kernel: drbd0: Connection established.
Jul 26 18:07:13 nd kernel: drbd0: I am(S):
1:00000004:00000001:0000001d:00000004:00
Jul 26 18:07:13 nd kernel: drbd0: Peer(S):
 1:00000004:00000001:0000001d:00000004:00
Jul 26 18:07:13 nd kernel: drbd0: drbd0_receiver [5598]:
cstate WFReportParams --> Connected
Jul 26 18:07:13 nd kernel: drbd0: Secondary/Unknown -->
Secondary/Secondary
```

B.2 Drbd - Servidor 03

```
Jul 26 18:07:26 servidor3 syslogd 1.4.1: restart.
Jul 26 18:07:27 servidor3 kernel: klogd 1.4.1, log
source = /proc/kmsg started.
Jul 26 18:09:32 servidor3 kernel: drbd: initialised. Version:
0.7.10 (api:77/proto:74)
Jul 26 18:09:32 servidor3 kernel: drbd: SVN Revision: 1743 build
 by root@servidor3.nide.com.br, 2005-07-20 14:29:52
Jul 26 18:09:32 servidor3 kernel: drbd: registered as block device
 major 147
Jul 26 18:09:32 servidor3 kernel: drbd0: resync bitmap: bits=366848
 words=11464
Jul 26 18:09:32 servidor3 kernel: drbd0: size = 1433 MB (1467392 KB)
Jul 26 18:09:33 servidor3 kernel: drbd0: 0 KB marked out-of-sync
 by on disk bit-map.
Jul 26 18:09:33 servidor3 kernel: drbd0: drbdsetup [3193]:
 cstate Unconfigured --> StandAlone
Jul 26 18:09:33 servidor3 kernel: drbd0: drbdsetup [3206]:
 cstate StandAlone --> Unconnected
Jul 26 18:09:33 servidor3 kernel: drbd0: drbd0_receiver [3207]:
 cstate Unconnected --> WFConnection
Jul 26 18:09:33 servidor3 kernel: drbd0: drbd0_receiver [3207]:
 cstate WFConnection --> WFReportParams
Jul 26 18:09:33 servidor3 kernel: drbd0: Handshake successful:
DRBD Network Protocol version 74
Jul 26 18:09:33 servidor3 kernel: drbd0: Connection established.
Jul 26 18:09:33 servidor3 kernel: drbd0: I am(S):
1:00000004:00000001:0000001d:00000004:00
Jul 26 18:09:33 servidor3 kernel: drbd0: Peer(S):
1:00000004:00000001:0000001d:00000004:00
Jul 26 18:09:33 servidor3 kernel: drbd0: drbd0_receiver [3207]:
 cstate WFReportParams --> Connected
```

```
Jul 26 18:09:33 servidor3 kernel: drbd0: Secondary/Unknown -->
Secondary/Secondary
```

B.3 Heartbeat - Servidor ND

```
Jul 26 18:08:07 nd heartbeat: [5654]: info: *************************
Jul 26 18:08:07 nd heartbeat: [5654]: info: Configuration validated.
Starting heartbeat 1.99.4
Jul 26 18:08:07 nd heartbeat: [5655]: info: heartbeat: version 1.99.4
Jul 26 18:08:08 nd heartbeat: [5655]: info: Heartbeat generation: 16
Jul 26 18:08:08 nd heartbeat: [5655]: info: glib: UDP Broadcast
heartbeat started on port 694 (694) interface eth0
Jul 26 18:08:08 nd heartbeat: [5655]: info: pid 5655 locked in memory.
Jul 26 18:08:08 nd heartbeat: [5655]: info: Local status now set to:
'up'
Jul 26 18:08:08 nd heartbeat: [5658]: info: pid 5658 locked in memory.
Jul 26 18:08:08 nd heartbeat: [5659]: info: pid 5659 locked in memory.
Jul 26 18:08:08 nd heartbeat: [5660]: info: pid 5660 locked in memory.
Jul 26 18:08:08 nd heartbeat: [5655]: info: Link nd:eth0 up.
Jul 26 18:08:10 nd heartbeat: [5655]: info: Link servidor3:eth0 up.
Jul 26 18:08:10 nd heartbeat: [5655]: info: Local status now set to:
 'active'
Jul 26 18:08:10 nd heartbeat: [5655]: info: Status update for node
 servidor3: status up
Jul 26 18:08:10 nd heartbeat: [5655]: info: Status update for node
 servidor3: status active
Jul 26 18:08:10 nd heartbeat: info: Running /etc/ha.d/rc.d/
status status
Jul 26 18:08:10 nd heartbeat: info: Running /etc/ha.d/rc.d/
status status
Jul 26 18:08:22 nd heartbeat: [5655]: info: local resource
 transition completed
Jul 26 18:08:22 nd heartbeat: [5655]: info: Initial resource
acquisition complete (T_RESOURCES(us))
Jul 26 18:08:22 nd heartbeat: [5655]: info: remote resource
 transition completed.
Jul 26 18:08:23 nd heartbeat: [5674]: info: Local Resource
acquisition completed.
Jul 26 18:08:23 nd heartbeat: info: Running /etc/ha.d/rc.d/
ip-request-resp ip-request-resp
Jul 26 18:08:23 nd heartbeat: received ip-request-resp IPaddr::10.0.0.201
 OK yes
Jul 26 18:08:23 nd heartbeat: info: Acquiring resource group: nd
IPaddr::10.0.0.201 drbddisk httpd
```

```
Jul 26 18:08:23 nd heartbeat: info: Running /etc/ha.d/resource.d/IPaddr
 10.0.0.201 start
Jul 26 18:08:23 nd heartbeat: info: /sbin/ifconfig eth0:0 10.0.0.201
netmask 255.0.0.0\^Ibroadcast 10.255.255.255
Jul 26 18:08:23 nd heartbeat: info: Sending Gratuitous Arp for 10.0.0.201
 on eth0:0 [eth0]
Jul 26 18:08:23 nd heartbeat: /usr/lib/heartbeat/send_arp -i 500 -r 10 -p
/var/lib/heartbeat/rsctmp/send_arp/send_arp-10.0.0.201 eth0 10.0.0.201
auto 10.0.0.201 ffffffffffff
Jul 26 18:08:23 nd heartbeat: info: Running /etc/ha.d/resource.d/drbddisk
 start
Jul 26 18:08:23 nd kernel: drbd0: Secondary/Secondary -->
Primary/Secondary
Jul 26 18:08:24 nd heartbeat: info: Running /etc/ha.d/resource.d/httpd
  start
```

B.4 Heartbeat - Servidor 03

```
Jul 26 18:10:30 servidor3 heartbeat: [3265]: info: *************************
Jul 26 18:10:30 servidor3 heartbeat: [3265]: info: Configuration validated.
Starting heartbeat 1.99.4
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: heartbeat:
 version 1.99.4
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: Heartbeat
generation: 16
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: glib: UDP Broadcast
heartbeat started on port 694 (694) interface eth0
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: pid 3266 locked
 in memory.
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: Local status now
set to: 'up'
Jul 26 18:10:30 servidor3 heartbeat: [3269]: info: pid 3269 locked
 in memory.
Jul 26 18:10:30 servidor3 heartbeat: [3270]: info: pid 3270 locked
 in memory.
Jul 26 18:10:30 servidor3 heartbeat: [3271]: info: pid 3271 locked
in memory.
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: Link servidor3:
eth0 up.
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: Link nd:eth0
up.
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: Status update
for node nd: status active
Jul 26 18:10:30 servidor3 heartbeat: [3266]: info: Local status
```

```
 now set to: 'active'
Jul 26 18:10:30 servidor3 heartbeat: info: Running
/etc/ha.d/rc.d/status status
Jul 26 18:10:42 servidor3 heartbeat: [3266]: info: local
 resource transition completed.
Jul 26 18:10:42 servidor3 heartbeat: [3266]: info: Initial
resource acquisition complete (T_RESOURCES(us))
Jul 26 18:10:42 servidor3 heartbeat: [3266]: info: remote
 resource transition completed.
Jul 26 18:10:42 servidor3 heartbeat: [3278]: info: No local
 resources [/usr/lib/heartbeat/ResourceManager listkeys servidor3] to acquire.
Jul 26 18:10:43 servidor3 kernel: drbd0: Secondary/Secondary
--> Secondary/Primary
```

B.5 Mon - Servidor 03

```
Jul 26 18:11:22 servidor3 mon[3287]: mon server started
```

B.6 Primeiro servidor cai

```
Jul 26 18:10:27 nd kernel: eth0: Setting half-duplex based on MII
 #1 link partner capability of 0000.
Jul 26 18:10:29 nd heartbeat: [5655]: info: Resources being
acquired from servidor3.
Jul 26 18:10:29 nd heartbeat: [5655]: info: Link servidor3:eth0
 dead.
Jul 26 18:10:29 nd heartbeat: info: Running /etc/ha.d/rc.d/status
status
Jul 26 18:10:29 nd heartbeat: info: /usr/lib/heartbeat/
mach_down: nice_failback: foreign resources acquired
Jul 26 18:10:29 nd heartbeat: [5655]: info: mach_down takeover
 complete.
Jul 26 18:10:29 nd heartbeat: info: mach_down takeover complete
 for node servidor3.
Jul 26 18:10:29 nd heartbeat: [5914]: info: Local Resource
acquisition completed.
Jul 26 18:10:36 nd kernel: drbd0: drbd0_asender [5608]:
cstate Connected --> NetworkFailure
Jul 26 18:10:36 nd kernel: drbd0: asender terminated
Jul 26 18:10:36 nd kernel: drbd0: drbd0_receiver [5598]:
cstate NetworkFailure --> BrokenPipe
Jul 26 18:10:36 nd kernel: drbd0: worker terminated
Jul 26 18:10:36 nd kernel: drbd0: drbd0_receiver [5598]:
```

```
cstate BrokenPipe --> Unconnected
Jul 26 18:10:36 nd kernel: drbd0: Connection lost.
Jul 26 18:10:36 nd kernel: drbd0: drbd0_receiver [5598]:
cstate Unconnected --> WFConnection
Jul 26 18:11:37 nd heartbeat: [5655]: info: See documentation for
information on tuning deadtime.
```

B.7 Segundo servidor assume

```
Jul 26 18:12:51 servidor3 heartbeat: [3266]: info: Resources being
acquired from nd.
Jul 26 18:12:51 servidor3 heartbeat: [3266]: info: Link nd:eth0 dead.
Jul 26 18:12:51 servidor3 heartbeat: info: Running /etc/ha.d/rc.d/status
status
Jul 26 18:12:51 servidor3 heartbeat: info: Taking over resource group
 IPaddr::10.0.0.201
Jul 26 18:12:51 servidor3 heartbeat: [3298]: info: No local resources
 [/usr/lib/heartbeat/ResourceManager listkeys servidor3] to acquire.
Jul 26 18:12:51 servidor3 heartbeat: info: Acquiring resource
group: nd IPaddr::10.0.0.201 drbddisk httpd
Jul 26 18:12:51 servidor3 heartbeat: info: Running /etc/ha.d/res
ource.d/IPaddr 10.0.0.201 start
Jul 26 18:12:51 servidor3 heartbeat: info: /sbin/ifconfig eth0:0
 10.0.0.201 netmask 255.0.0.0\^Ibroadcast 10.255.255.255
Jul 26 18:12:51 servidor3 heartbeat: info: Sending Gratuitous Arp
 for 10.0.0.201 on eth0:0 [eth0]
Jul 26 18:12:51 servidor3 heartbeat: /usr/lib/heartbeat/send_arp
-i 500 -r 10 -p /var/lib/heartbeat/rsctmp/send_arp/send_arp-
10.0.0.201 eth0
10.0.0.201 auto 10.0.0.201 ffffffffffff
Jul 26 18:12:51 servidor3 heartbeat: info: Running /etc/ha.d/
resource.d/drbddisk  start
Jul 26 18:12:56 servidor3 kernel: drbd0: drbd0_asender [3217]:
cstate Connected --> NetworkFailure
Jul 26 18:12:56 servidor3 kernel: drbd0: asender terminated
Jul 26 18:12:56 servidor3 kernel: drbd0: drbd0_receiver [3207]:
cstate NetworkFailure --> BrokenPipe
Jul 26 18:12:56 servidor3 kernel: drbd0: worker terminated
Jul 26 18:12:56 servidor3 kernel: drbd0: drbd0_receiver [3207]:
cstate BrokenPipe --> Unconnected
Jul 26 18:12:56 servidor3 kernel: drbd0: Connection lost.
Jul 26 18:12:56 servidor3 kernel: drbd0: drbd0_receiver [3207]:
cstate Unconnected --> WFConnection
Jul 26 18:12:56 servidor3 kernel: drbd0: Secondary/Unknown --> Pr
```

```
imary/Unknown
Jul 26 18:12:56 servidor3 heartbeat: info: Running /etc/ha.d/resou
rce.d/httpd  start
Jul 26 18:12:57 servidor3 heartbeat: info: /usr/lib/heartbeat/mach
_down: nice_failback: foreign resources acquired
Jul 26 18:12:57 servidor3 heartbeat: [3266]: info: mach_down takeover
 complete.
Jul 26 18:12:57 servidor3 heartbeat: info: mach_down takeover complete
 for node nd.
```

B.8 Primeiro servidor volta

```
Jul 26 18:11:37 nd heartbeat: [5655]: info: Link servidor3:eth0 up.
Jul 26 18:11:37 nd heartbeat: [5655]: info: Status update for node
servidor3: status active
Jul 26 18:11:37 nd heartbeat: [5655]: info: No pkts missing from
servidor3!
Jul 26 18:11:37 nd heartbeat: info: Running /etc/ha.d/rc.d/status status
Jul 26 18:11:37 nd kernel: eth0: Setting full-duplex based on MII
\#1 link partner capability of 4de1.
Jul 26 18:11:39 nd heartbeat: [5655]: info: Heartbeat shutdown in
progress. (5655)
Jul 26 18:11:39 nd heartbeat: [5655]: info: Received shutdown notice
from 'servidor3'.
Jul 26 18:11:39 nd heartbeat: [5655]: info: Resource takeover cancelled
 - shutdown in progress.
Jul 26 18:11:39 nd heartbeat: [5980]: info: Giving up all HA resources.
```

B.9 Segundo servidor libera os serviços

```
Jul 26 18:13:57 servidor3 heartbeat: [3266]: info: See documentation for
information on tuning deadtime.
Jul 26 18:13:57 servidor3 heartbeat: [3266]: info: Link nd:eth0 up.
Jul 26 18:13:57 servidor3 heartbeat: [3266]: info: Status update for node
nd: status active
Jul 26 18:13:57 servidor3 heartbeat: info: Running /etc/ha.d/rc.d/status status
Jul 26 18:13:57 servidor3 heartbeat: [3266]: info: No pkts missing from nd!
Jul 26 18:13:59 servidor3 heartbeat: [3266]: info: Heartbeat shutdown in
progress. (3266)
Jul 26 18:13:59 servidor3 heartbeat: [3595]: info: Giving up all HA resources.
Jul 26 18:13:59 servidor3 heartbeat: info: Releasing resource group: nd
IPaddr::10.0.0.201 drbddisk httpd
Jul 26 18:13:59 servidor3 heartbeat: info: Running /etc/ha.d/resource.d/httpd
```

```
stop
Jul 26 18:13:59 servidor3 heartbeat: info: Running /etc/ha.d/resource.d/drbddisk
stop
Jul 26 18:13:59 servidor3 kernel: drbd0: Primary/Unknown --> Secondary/Unknown
Jul 26 18:13:59 servidor3 heartbeat: info: Running /etc/ha.d/resource.d/IPaddr
10.0.0.201 stop
Jul 26 18:13:59 servidor3 heartbeat: info: /sbin/route -n del -host 10.0.0.201
Jul 26 18:13:59 servidor3 heartbeat: info: /sbin/ifconfig eth0:0 down
Jul 26 18:13:59 servidor3 heartbeat: info: IP Address 10.0.0.201 released
```

B.10 Servidor de Alta-Disponibilidade perde conectividade com o Roteador

```
Jul 26 18:15:13 servidor3 mon[3287]: failure for roteador ping 1122416113
unidentified output from fping: [ICMP Host Unreachable from 192.168.200.2
for ICMP Echo sent to 192.168.200.254]
Jul 26 18:15:13 servidor3 mon[3287]: calling alert mail.alert for
roteador/ping (/usr/lib/mon/alert.d/mail.alert,nd@nide.com.br) unidentified
output from fping: [ICMP Host Unreachable from 192.168.200.2 for ICMP Echo
sent to 192.168.200.254]
```

Apêndice C

CD

Esta sessão leva um CD, onde nele conterá uma cópia digital em formato PDF desta monografia, os programas para se implementar um servidor de Alta-Disponibilidade, os e-books utilizado como referência para a elaboração desta monografia e os logs de todo o estudo de caso.