***Centro de Difusão de***
***Tecnologia e Conhecimento***

*EliphasS*

# CDTC

# [CLUSTER]

*Um cluster é um sistema formado por um conjunto de computadores (nós) que trabalham em conjunto como se fossem apenas uma grande máquina. Este tipo de implementação utiliza-se de um tipo especial de sistema operacional classificado como sistema distribuído. É construído muitas vezes a partir de computadores convencionais (desktops), sendo que estes vários computadores são ligados em rede e comunicam-se através do sistema de forma que trabalhem como se fosse uma única máquina de grande porte.*

# CONTEÚDO

# LIÇÃO 1 - INTRODUÇÃO

## INTRODUÇÃO

Um cluster é um sistema formado por um conjunto de computadores (nós) que trabalham em conjunto como se fossem apenas uma grande máquina. Este tipo de implementação utiliza-se de um tipo especial de sistema operacional classificado como sistema distribuido. É construído muitas vezes a partir de computadores convencionais (desktops), sendo que estes vários computadores são ligados em rede e comunicam-se através do sistema de forma que trabalhem como se fosse uma única máquina de grande porte.

## UM POUCO DE HISTÓRIA

A idéia inicial que conduz ao cluster foi desenvolvida na década de 1960 pela IBM como uma forma de interligar grandes main frames, visando obter uma solução comercialmente viável de paralelismo. Nesta época o sistema HASP (*Houston Automatic Spooling Priority*) da IBM e seu sucessor, JES (*Job Entry System*) proviam uma maneira de distribuir tarefas nos mainframes interligados. A IBM ainda hoje suporta o cluster de mainframes através do *Parallel Sysplex System* que permite ao hardware, sistema operacional, *middleware*, e o software de gerenciamento do sistema prover uma melhora dramática na performance e custo ao permitir que usuários da grandes mainframes continuem utilizando suas aplicações existentes.

Entretanto, o cluster ganhou força até que três tendências convergiram nos anos 1980: microprocessadores de alta performance, redes de alta velocidade, e ferramentas padronizadas para computação distribuida de alto desempenho. Uma quarta tendência possível é a crescente necessidade de poder de processamento para aplicações científicas e comerciais unida ao alto custo e a baixa acessibilidade dos tradicionais dos supercomputadores.

No final de 1993, Donald Becker e Thomas Sterling iniciaram um esboço de um sistema de processamento distribuido construído a partir de hardware convencional como uma medida de combate aos custos dos supercomputadores. No início de 1994, trabalhando no CESDIS, com o patrocínio do projeto HTPCC/ESS, nasce o primeiro cluster e, conseqüentemente, o projeto Beowulf.

O protótipo inicial era um cluster de 16 processador DX4 ligados por dois canais Ethernet acoplados (*Ethernet Bolding*). A máquina foi um sucesso instantâneo, e esta idéia rapidamente se espalhou pelos meios acadêmicos e pela NASA e outras comunidades de pesquisa.

## TIPOS DE CLUSTER

Existem basicamente três tipos de cluster: **Alta Disponibilidade**, **Balanceamento de Carga** e **Processamento Distribuído**.

### CLUSTER DE ALTA DISPONIBILIDADE

Um cluster de alta disponibilidade visa a manter os serviços prestados por um sistema a outro mesmo que esse venha a falhar internamente. Aí está implícito o conceito de mascaramento de falhas, através de redundância ou replicação. Um determinado serviço, que se quer altamente disponível, é colocado por trás de uma camada de abstração, que permita mudanças em seus mecanismos internos mantendo intacta a interação com elementos externos. Este tipo de implementação é usado principalmente em aplicações críticas onde o serviço não pode ficar indisponível em momento algum, mesmo em caso de falha de hardware.

Para temos uma idéia mais concreta pense numa grande loja virtual, imagine qual seria o prejuízo se o site ficasse indisponível por 10 minutos!

Clusters de alta disponibilidade visam a resolver este tipo de problema. A idéia básica por trás dessa implementação é que caso um nó venha a falhar outro assuma no mesmo momento sem que seja percebido pelos usuários do serviço.

## CLUSTER DE BALANCEAMENTO DE CARGA

Essa implementação permite a distribuição das requisições de recursos para todas as máquinas que fazem parte do cluster. Este tipo de implementação é muito comum em servidores web e funciona da seguinte maneira: uma máquina é responsável por distribuir as requisições entre os diversos nós, cada nó faz o seu trabalho e manda uma resposta, o sistema que fez a requisição vê tudo isso como apenas uma máquina.

## CLUSTER DE PROCESSAMENTO DISTRIBUÍDO

Esse tipo de cluster tem como objetivo dividir toda tarefa computacional em diversas tarefas pequenas que podem ser realizadas por diversos pcs ao mesmo tempo. A idéia é mais ou menos parecida com a do cluster de balanceamento de carga. Essa implementação é muito utilizada em cálculo de problemas matemáticos e renderização de imagens.

*CLUSTER DE ALTA DISPONIBILIDADE: CONSTRUIDO PARA PROVER SERVIÇOS E RECURSOS DE FORMA ININTERRUPTA.*

*CLUSTER DE BALANCEAMENTO DE CARGA: DISTRIBUI AS REQUISIÇÕES DE SERVIÇOS ENTRE TODOS OS NODOS DO CLUSTER.*

*CLUSTER DE PROCESSAMENTO DISTRIBUÍDO: DIVIDE O PROCESSAMENTO ENTRE AS DIVERSAS MÁQUINAS, RESULTANDO NUMA ALTA PERFORMACE COMPUTACIONAL.*

# LIÇÃO 2 - ALTA DISPONIBILIDADE

## ALTA DISPONIBILIDADE

Cada vez mais o homem depende de máquinas e de diversos sistemas computacionais. Alguns desses sistemas não podem ficar insdisponíveis por menor que seja o intervalo de tempo.

Podemos pensar, por exemplo, num banco. É totalmente inaceitável que o sistema de caixas eletrônicos de um banco fique indisponível por um tempo menor que seja, isso pode causar grande prejuízo a instituição.

Uma loja virtual (*Submarino*, *Mercado Livre*, *Best Buy*) é um sistema crítico e que tem que ficar o tempo todo disponível, a simples queda de um servidor de banco de dados pode causar prejuízo devido a interrupção nas vendas.

Para garantir que o sistema esteja sempre, ou a maior parte do tempo, disponível são necessárias diversas atitudes quando da implementação desse sistema. Várias podem ser as causas das falhas, tanto pode ser um problema de hardware como pode ser um problema de software, por isso deve haver a redundância de hardware e de software.

A alta disponibilidade tem por objetivo manter um sistema que provém algum tipo de serviço através de uma rede, mesmo que este passe por problemas de falhas de hardware, de manunteção, de fornecimento de energia, atualização de softwares. Para manter um sistema seguindo essa política são necessárias ações de manutenção e administração, não basta apenas instalar e configurar para que tudo funcione corretamente, deve-se trabalhar ao máximo para fazer com que o tempo que um serviço que fica no ar seja o maior possível.

## DISPONIBILIDADE

Disponibilidade é a probabilidade de um sistema estar funcional e pronto para uso em um determindado instante. Quanto mais disponível for um sistema, maior será a probabilidade de ele estar disponível.

A disponibilidade pode ser dividida em três tipos: **Disponibilidade Básica**, **Alta Disponibilidade** e **Disponibilidade Contínua**.

### DISPONIBILIDADE BÁSICA

É a disponibilidade de computadores comuns, máquina desta classe estão disponíveis 99% do tempo;

### ALTA DISPONIBILIDADE

É a disponibilidade de sistemas montados com redundância em todos os pontos prováveis de falha, esse tipo de implementação deve ser feita de maneira a torná-lo capaz de detectar, corrigir e esconder falhas.

A tabela a seguir mostra o nível de disponibilidade e o tempo que o sistema permanece indisponível por unidade de tempo.

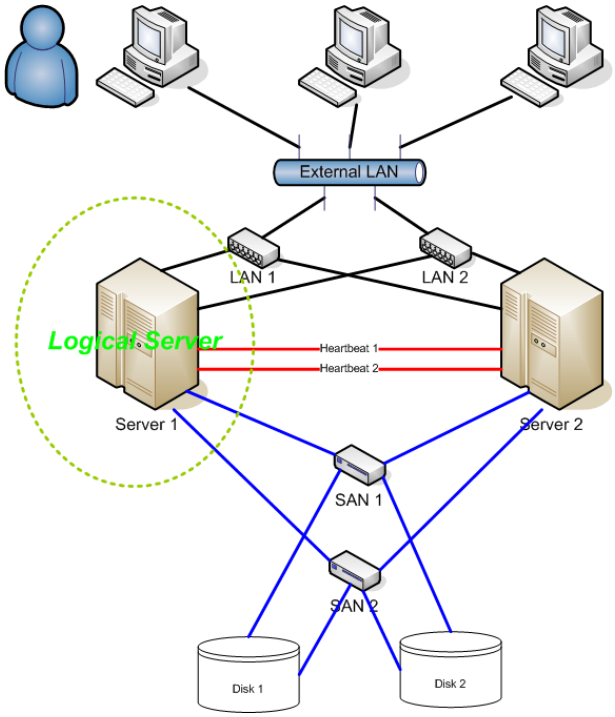
| **Disponibilidade (%)** | **Indisponibilidade/ano** | | | | **Indisponibilidade/mês** | | | |
|---|---|---|---|---|---|---|---|---|
| | D | H | M | S | D | H | M | S |
| 95 | 18 | 06 | 00 | 00 | 01 | 12 | 00 | 00 |
| 96 | 14 | 14 | 24 | 00 | 01 | 04 | 48 | 00 |
| 97 | 10 | 22 | 48 | 00 | 00 | 21 | 36 | 00 |
| 98 | 07 | 07 | 12 | 00 | 00 | 14 | 24 | 00 |
| 99 | 03 | 15 | 36 | 00 | 00 | 07 | 12 | 00 |
| 99,9 | 00 | 08 | 45 | 35,99 | 00 | 00 | 43 | 11,99 |
| 99,99 | 00 | 00 | 52 | 33,60 | 00 | 00 | 04 | 19,20 |
| 99,999 | 00 | 00 | 05 | 15,36 | 00 | 00 | 00 | 25,92 |

## DISPONIBILIDADE CONTÍNUA

Nesse implementação o sistema ficaria disponível 100% do tempo, pode-se chegar muito perto dessa situação com clusters de alta disponibilidade, onde todas as falhas são mascaradas.

## REDUNDÂNCIA

Para manter a disponibilidade de um serviço é essencial que haja redundância em todos as partes para que sejam diminuídos os pontos de falha, conhecidos como SPOF (*Single Point of Failure*). Quanto maior a redundância implementada menor a probabilidade de interrupções. A figura abaixo ilustra um cluster de alta disponibilidade projetado para prover um serviço de forma ininterrupta.




Note que na figura há redundância em todos os pontos: os servidores estão replicados, todos os dados então replicados (*RAID/Shared Storage*), a energia, os switches, tudo é construido de forma a manter o sistema "em pé" sob qualquer condição.

## FALHA

Um dos objetivos de implementar um sistema de alta disponibilidade é tornar esse sistema imune a falhas. Falha é um problema ocorrido devido ao mau funcionamento do hardware. Para todo tipo de falha existe uma solução, a queda de energia pode ser tratada, por exemplo, com o uso de no-breaks; um switch queimado pode acabar com a comunicação do sistema com os clientes, para resolver esse problema pode-se usar dois switches.

Resumindo, todos os SPOFs podem ser tratados de modo a tornar o sistema imune a falhas (problemas com respeito ao hardware), esse tipo de implementação geralmente encarece a construção do sistema, mas torna-o mais seguro e livre de problemas.

## ERRO E DEFEITO

Um problema com o hardware que compõe um sistema (falha), pode causar uma distorção durante a transferência dos dados, como por exemplo, a troca de um bit de 0 para 1. Este tipo de problema é denomindado erro.

Esse erro pode ser propagado durante a execução de alguma tarefa causando um defeito que pode acarretar no travamento de parte do sistema ou na geração de mensagens de erro, situações que podem ser percebidas pelos usuários e que devem ser evitadas.

## FAILOVER E FAILBACK

A ação de uma máquina assumir o serviço de outra é chamada de failover. É desejado que esse procedimento seja feito de forma automática, se tiver que ser feito de forma manual pode ter alguns problemas quanto à disponibilidade do serviço. Um failover manual exige que alguém monitore o serviço e em caso de problema execute o failover, esse procedimento pode levar algum tempo, o que não é aceitável em sistemas críticos.

O procedimento inverso ao *failover* é o *failback*. Logo após a realização de um failover, cabe ao administrador do sistema reparar todos os problemas e tornar a máquina novamente disponível. Assim que a máquina está novamente disponível pode-se fazer com que esta volte a fornecer os serviços, isso pode ser feito de maneira manual ou automática.

*UMA FALHA PODE CAUSAR UM ERRO, QUE PODE CAUSAR UM DEFEITO E TORNAR O SISTEMA INDISPONÍVEL.*

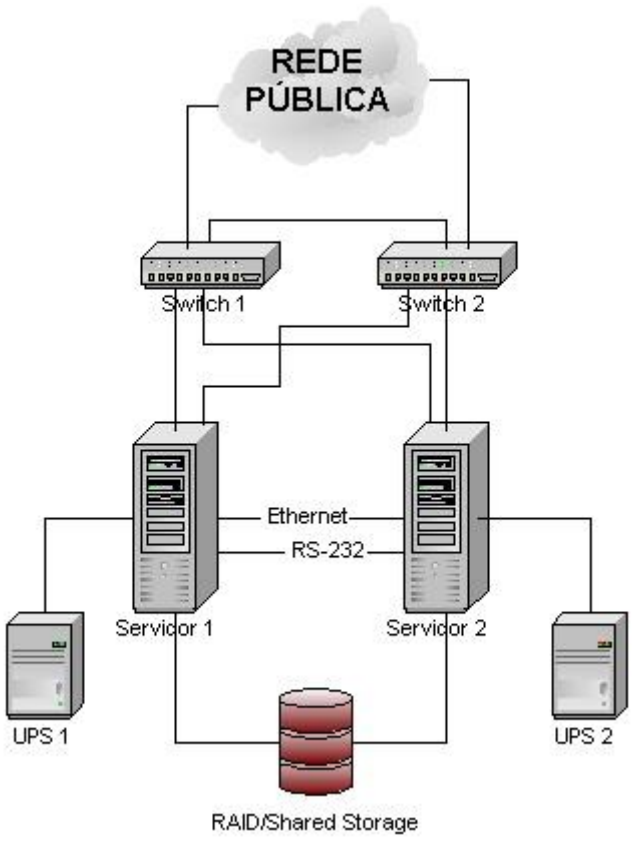
# LIÇÃO 3 - UMA SOLUÇÃO PARA A ALTA DISPONIBILIDADE

## UMA SOLUÇÃO PARA ALTA DISPONIBILIDADE

A visão deste curso é mais em cima dos procedimentos referentes aos softwares que compõem um cluster, não serão abordados profundamente aspectos referentes ao hardware.

O ambiente idealizado durante o decorrer do curso será o de um servidor web que deve permacer disponível o maior tempo possível, sem que haja queda nos serviçios prestados. Os clientes se conectarão ao servidor web por um ip que será o ip do cluster, no cluster cada nó (para finalidades práticas chamaremos os computadores de nós ou nodos) terá o seu próprio ip interno que não poderá ser visto pelos clientes. Um nó será o primário e ficará com o ip do cluster disponibilizando os serviços, caso esse nó venha a falhar, os serviços serão disponibilizados pelo outro nó e este passará à atender pelo ip do cluster. Pode parecer um pouco confuso agora mas ficará mais claro adiante.

A configuração do servidor será mais ou menos como na imagem abaixo:




- Os dois computadores são ligados cada um a um no-break para evitar que a queda de energia acarrete no desligamento da máquina. Note que estão ligados em no-breaks diferentes pois se estivessem ligados no mesmo equipamento uma simples queima de circuito deste causaria indisponibilidade total do serviço.

- Entre eles há uma conexão dedicada via cabo cross-over para a comunicação, mais tarde veremos qual a finalidade dessa conexão.

- O uso de dois switches é uma boa prática, assim como no caso do no-break, se os dois servidores estivessem ligados no mesmo equipamento, a falha deste acarretaria na indisponibilidade do sistema;

- Há uma redundância dos dados através de um sistema de RAID, para que os dados de um nó sejam os mesmo do outro, essa RAID será feita através da rede interna e serão utilizados os próprios HDs das máquinas para armazenar os dados;

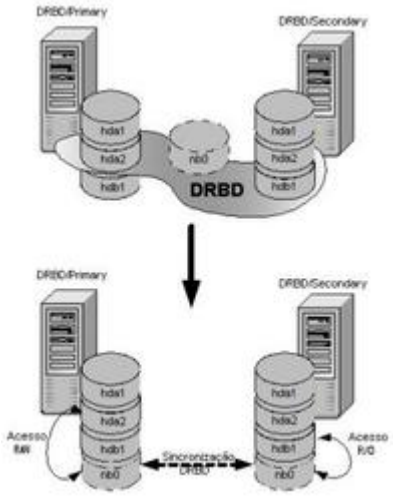
## RAID

RAID (*Redundant Array of Independent Disks*) ou Arranjo Redundante de Discos Independentes) é um arranjo de dois ou mais discos que trabalham realizando uma tarefa com o mesmo fim. Existem vários níveis de RAID, a usada no nosso sistema será a RAID 1.

***RAID 1*** consiste no espelhamento de dados, tudo que for gravado em um disco será gravado igualmente no outro, essa é uma opção de alta disponibilidade para o armazenamento de dados. No nosso caso será implementado RAID por software, o DRBD (Distributed Replicated Block Device ou Dispositivo de Bloco Replicado Distribuido) será o responsável por tal procedimento.

## DRBD

O DRBD além de ser um módulo do kernel do Linux, é também alguns scripts responsáveis por espelhar dados gravados em disco. O DRBD trabalha com dispositivos de bloco , cada um destes tem um estatuto que pode ser primário ou secundário. Tudo que for gravado no dispositivo primário será enviado para ser gravado também no dispositivo secundário. O DRBD cria um link entre um dispositivo virtual criado (**/dev/drbd***, de alto nível) e uma partição local do disco **(/dev/hda*** de baixo nível). Tudo é "gravado" no dispositivo de alto nível do dispositivo com estado primário, os dados são passados para o dispositivo de nível mais baixo e para os dispositivos secundários que realizam a mesma tarefa.



Resumindo, um programa escreve alguma coisa no **/dev/drbd*** do nó primário do cluster, isso será gravado então no **/dev/hda*** e passado para o nó secundário para ser gravado da mesma maneira. No caso de falha do nodo primário, o dispositivo que era secundário passa a ser primário. Assim que o nó que caiu voltar, pode-se reestabelecer ou não o antigo estado das máquinas (Primário/Secundário), mas é necessária a sincronização dos dados, pois podem ter sido gravadas informações no disco e para manter a consistência dos dados é necessário essa sincronização.

## CONSISTÊNCIA DE DADOS

A consistência dos dados é um fator muito importante e por isso é aconselhável utilizar um sistema de arquivos com suporte a **journaling**, no nosso caso utilizaremos **ext3**. A técnica de journaling consiste na criação de um log antes que alguma coisa seja gravada no disco, no caso de algum problema como o desligamento inesperado da máquina não é necessária uma verificação de todo o sistema de arquivos, com o log pode-se saber quais arquivos podem ter sido afetados.

A não utilização um sistema de arquivos com journaling, implicará num trabalho extra, toda vez que ocorrer um transição de estados entre os nós será necessária uma checagem da consistência do sistema de arquivos utilizando o fcsk, isso pode não ser muito agradável e prático.

## HEARTBEAT

Utilizaremos o programa **Heartbeat** para verificar a comunicação entre os nodos do cluster. A função do Heartbeat é trocar sinais entre as máquinas para verificar se determinada máquina está sozinha ou não. Caso o Heartbeat do nó secundário envie um sinal para o nó primário e não obtenha resposta, ele vai considerar que está sozinho no cluster e passará a prover os serviços que eram prestados por aquela.

## MON

O último utilitário para a montagem do cluster será o **MON**. Este software é responsável por monitorar a rede interna do cluster. Caso o MON verifique que um certo computador está sozinho na rede ele dá um shutdown no Heartbeat e dessa maneira o outro nó assume o serviço.

Para a montagem do nosso cluster utilizaremos então a solução: **DRBD + Heartbeat + MON + Apache**.

*DENTRE OS SOFTWARES QUE COMPÕEM O CLUSTER, O RESPONSÁVEL PELA REPLICAÇÃO DOS DADOS É O: DRBD*

# LIÇÃO 4 - UM POUCO SOBRE O SSH

## NOÇÕES DE SSH

A ferramenta SSH pode ajudar muito na hora de realizar alguns procedimentos de instalação e configuração das máquinas.

Para instalar o SSH basta dar o comando a seguir no terminal e como o super-usuário:

```
debian:~# apt-get install ssh
```

Com o SSH você pode utilizar vários computadores sem ter que ficar mudando de computador, ele faz com que não seja necessário sentar em cada computador e configurar tudo, de uma máquina você pode acessar todas as outras.

Vamos a um exemplo:

Suponhamos que você queira instalar uma dada ferramenta em dois computadores, um com nome (*hostname*) **debian1** e IP 10.0.0.1, outro com nome **debian2** e IP 10.0.0.2. Agora considerando que o usuário se encontra na máquina debian1, para acessar a máquina debian2 ele deve dar o seguinte comando:

```
debian1:~# ssh root@10.0.0.2
```

Com esse comando ele acessará a máquina com o IP 10.0.0.2 e como o usuário root, o programa logicamente pedirá a senha do usuário determinado:

```
root@10.0.0.2's password:
```

É só digitar a senha e pronto você estará na máquina **debian2**. Agora é so utilizar o terminal como se estivesse na máquina debian2, para instalar a ferramenta pelo APT bastaria:

```
debian2:~# apt-get install ferramenta_qualquer
```

Para sair de debian2 e voltar para **debian1** basta:

```
debian2:~# exit
```

Uma outra ferramenta muito interessante é o SCP, ele é muito útil para copiar arquivos de um computador para o outro.

Se, por exemplo, um programa foi instalado e esse tem um arquivo de configuração, para não ter que ficar configurando esse mesmo arquivo em todas as máquinas podemos simplesmente configurar em um e depois copiar para os demais.

Levando em conta as mesmas máquinas do exemplo do SSH, para copiar o arquivo **/etc/conf/exemplo.conf** do **debian1** para a pasta **/etc/conf** do **debian2** damos o comando:

```
debian1:~# scp /etc/conf/exemplo.conf root@10.0.0.2:/etc/conf
```

Para copiar uma pasta, todos os arquivos e subpastas devemos usar o parâmetro "**–r**":

```
debian1:~# scp -r /etc/conf/ root@10.0.0.2:/etc/
```

Essas duas ferramentas podem ser muito úteis para configurar diversas máquinas de uma mesma rede, para saber mais sobre elas basta acessar os manuais com os comandos:

```
debian:~# man ssh
debian:~# man scp
```

---

*O COMANDO SSH PARA QUE UM USUÁRIO ACESSE A MÁQUINA DE IP 192.168.0.105 COM O USUÁRIO CDTC SERÁ:*

*LOCALHOST:~# SSH CDTC@192.168.0.105*

---

# LIÇÃO 5 - CONFIGURANDO A REDE

## PROCURANDO PELAS FERRAMENTAS

Tudo que for feito nessa lição deve ser repetido nos dois nós do cluster.

O primeiro passo para configurar a rede no cluster é verificar se possui a ferramenta para configuração da rede que será muito útil no decorrer do curso, para isso abrimos o terminal e damos o seguinte comando:

```
user@debian:~$ dpkg -l iproute
```

Se esse pacote não estiver instalado no sistema instale-o de preferência com a ferramenta APT. Logado como root:

```
debian:/home/user# apt-get update
debian:/home/user# apt-get install iproute
```

É só aguardar o sistema baixar e instalar o pacote.

(ISSO VALE NO DEBIAN, QUE É A DISTRIBUIÇÃO BASE PARA ESTE CURSO. FICA A CARGO DO ALUNO SABER COMO SE EFETUA TAL PROCEDIMENTO NA SUA DISTRIBUIÇÃO)

Geralmente as distribuições já vem com essa ferramenta. Agora podemos dar o comando abaixo para verificar o estado atual das interfaces de rede.

```
user@debian:~$/sbin/ifconfig
```

Aparecerá algo como:

```
eth0 Link encap:Ethernet HWaddr 08:00:46:7A:02:B0
inet addr:192.168.0.3 Bcast:192.168.0.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:23363 errors:0 dropped:0 overruns:0 frame:0
TX packets:21798 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:13479541 (12.8 MiB) TX bytes:20262643 (19.3 MiB)
Interrupt:9

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:230172 errors:0 dropped:0 overruns:0 frame:0
TX packets:230172 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:22685256 (21.6 MiB) TX bytes:22685256 (21.6 MiB)
```

As interfaces **eth0** e **lo** estão ativas, a eth0 é utilizada para se comunicar com a rede, a lo (*loopback*) é utilizada para que o computador responda a si mesmo. No nosso caso utilizaremos a interface Ethernet (eth) para a comunicação entre os nodos.

*PARA A MELHOR COMPREENSÃO VAMOS DEFINIR COMO **USER@DEBIAN** UM USUÁRIO QUALQUER DO SISTEMA E **DEBIAN** O USUÁRIO LOGADO COMO ROOT.*

*NOTE SEMPRE O DIRETÓRIO ONDE ESTÃO SENDO EXECUTADOS OS COMANDOS.*

## CONFIGURANDO A REDE

Para executar os passos a seguir é necessário estar logado como super-usuário:

```
user@debian:~$ su
Password:
```

Cada nodo do cluster terá um IP fixo, esse IP não será visto pelos clientes que requisitarão os serviços será apenas para a comunicação entre os nodos, existirá um IP que será o IP do cluster mas isso será definido num outro momento. Devemos editar então o arquivo **/etc/network/interfaces**, utilize o editor de sua preferência.

```
debian:/home/user# nano /etc/network/interfaces
```

Adicione as seguintes linhas ao final do arquivo:

```
auto eth0:0
iface eth0:0 inet static
address 10.0.0.1
netmask 255.255.255.0
network 10.0.0.0
broacast 10.0.0.255
```

Para levantar essa rede dê o comando:

```
debian:/home/user# ifup eth0:0
```

Salve e feche o arquivo. Para certificarmos que tudo ocorreu de maneira correta podemos verificar as interfaces através do comando **ifconfig**, devem aparecer então as informações relativas a interface **eth0:0**.

```
eth0:0 Encapsulamento do Link: Ethernet Endereço de HWaddr
 08:00:46:7A:02:B0
inet end.: 10.0.0.1 Bcast:10.0.0.255 Masc:255.255.255.0
UP BROADCASTRUNNING MULTICAST MTU:1500 Métrica:1
IRQ:209 Endereço de E/S:0x8800
```

Esse será então o nodo primário do cluster, o mesmo procedimento deve ser feito no nodo secundário, a única diferença é o IP, o primário responderá pelo ip 10.0.0.1 e o secundário por 10.0.0.2 (apenas mude a linha address do arquivo interfaces).

Para terminar basta editar o arquivo hosts:

```
debian:/home/user# nano /etc/hosts
```

Adicione as seguintes linhas no arquivo, elas resolvem os nomes entre os nodos sem a necessidade de um DNS. Na nossa implementação o nodo principal será chamdo de debian1 e o secundário de debian2. (Faça isso nos dois nodos do cluster).

```
10.0.0.1 debian1
10.0.0.2 debian2
```

Feito isso, a rede interna do cluster estará configurada agora pode-se usar ferramentas como o SSH para realizar as tarefas no dois nodos ao invés de sentar em cada máquina e realizar as mesmas tarefas.

***IMPORTANTE!***

*ESTAS CONFIGURAÇÕES SÃO PARA CADA NODO. O CLUSTER NÃO RESPONDERÁ PELO IP 10.0.0.1 NEM PELO IP 10.0.0.2. POSTERIORMENTE SERÁ ESCOLHIDO UM IP PARA QUE O CLUSTER RESONDA POR ELE.*

# LIÇÃO 6 - COMPILANDO O KERNEL

## O KERNEL

O kernel é o núcleo do sistema, ele representa a camada mais baixa de interface com o Hardware, sendo responsável por gerenciar os recursos do Sistema operacional como um todo. É no kernel que estão definidas funções para operação com periféricos (mouse, discos, impressoras, interface serial/interface paralela), gerenciamento de memória, entre outros. Resumidamente, o kernel é um conjunto de programas que fornece para os programas de usuário (aplicativos) uma interface para utilizar os recursos do sistema. O kernel do Linux é um kernel monolítico carregador de módulos, ao contrário dos núcleos monolíticos padrão, os drivers de dispositivos são facilmente configurados como módulos, e são carregados/descarregados enquanto o sistema corre.

Para o funcionamento do aplicativo DRBD é altamente recomendável uma compilação do kernel, como já foi dito anteriormente o DRBD é formado por um módulo do kernel e alguns scripts. Apesar de demorado a compilação é um processo razoavelmente fácil de ser realizado.

Atenção, o kernel não é o sistema operacional em si, por isso um novo kernel não fará com que você perca os seus programas, arquivos e etc.

## COMPILANDO O KERNEL

Para compilar o kernel precisamos antes instalar alguns pacotes, utilizaremos novamente a ferramenta APT.

```
debian:/home/user# apt-get update
```

```
debian:/home/user# apt-get install libdb3-dev libncurses5-dev
 docbook-utils fakeroot dpatch kernel-package
```

Depois de instalados todos esses pacotes baixamos os fontes do kernel, utilizaremos o 2.6.15.

```
debian:/home/user# apt-get install linux-source-2.6.15
```

Feito isso, haverá agora um pacote chamado linux-source-2.6.15.tar.bz2 no diretório /usr/src. Vá até esse diretório e descompacte o pacote.

```
debian:/home/user# cd /usr/src
debian:/usr/src# tar xjf /usr/src/linux-source-2.6.15.tar.bz2
```

Feito isso será então criado automáticamente o diretório /usr/src/linux-source-2.6.15. Entre nesse diretório e copie o seu config, isso é altamente recomendado se você já possui um kernel debian funcionando perfeitamente.

```
debian:/usr/src# cd linux-source-2.6.15
debian:/usr/src/linux-source-2.6.15# cp
 /boot/config_kernel_atual.config
```

Você também pode configurar o kernel manualmente com o comando:

```
debian:/usr/src/linux-source-2.6.15# make menuconfig
```

*O KERNEL DEVE TER A OPÇÃO DE CARREGAR MÓDULOS CONFIGURADA.*

Vamos agora a compilação em si.

```
debian:/usr/src/linux-source-2.6.15# make-kpkg --rootcmd
 fakeroot --revision custom01 --append-to-version -cluster --
 initrd binary-arch
```

*NOTE QUE ANTES DA CADA COMANDO EXISTEM DOIS "-" E NÃO APENAS UM!*

Cada uma dos opções do comando make-kpkg tem um significado:

- --rootcmd - fornece meios de ganhar o acesso de super usuário necessário para contruir um pacote .deb com o kernel;
- --revision - troca o número de revisão do Debian;
- --append-to-version - adiciona a palavra ao nome do kernel que você está a compilar, no nosso caso o kernel será o 2.6.15-cluster;
- --initrd - indica que junto com o kernel deve ser criado um arquivo initrd que contém todos os módulos para carregar o sistema;

- binary-arch - produz pacotes separados kernel_headers e kernel_image;

Para saber mais sobre o comando make-kpkg leia o manual deste.

```
debian:/usr/src/linux-source-2.6.15# man make-kpkg
```

## INSTALANDO O NOVO KERNEL

Depois de um bom tempo compilando o kernel serão criados então dois pacotes .deb para serem instalados (linux-image-2.6.15-cluster_custom01_i386.deb e linux-headers-2.6.15-cluster_custom01_i386.deb), esses pacotes estarão no diretório /usr/src. Se tudo ocorreu bem durante a compilação basta instalar o pacote image.

```
debian:/usr/src/linux-source-2.6.15# cd ..
debian:/usr/src# dpkg -i linux-image-2.6.15-
 cluster_custom01_i386.deb
```

Agora você tem disponível um novo kernel, o 2.6.15-cluster, para acessá-lo temos primeiro que dizer ao gerenciador de boot para carregá-lo na inicialização. No GRUB:

```
debian:/usr/src#update-grub
```

Na próxima vez que o sistema for inicializado haverá na tela de opções de sistemas algo como "Debian GNU/Linux, kernel 2.6.15-cluster", basta selecionar para inicializar o sistema com o novo kernel instalado.

No LILO:

```
debian:/usr/src# nano /etc/lilo.conf
```

Adicione o novo kernel ao arquivo lilo.conf

```
image=/boot/vmlinuz-2.6.15-cluster
label=Linux-2.6.15-cluster
root=/dev/hd_partição_do_sistema
read-only
```

Salve, feche o arquivo e reinstale o LILO com o comando:

```
debian:/usr/src# lilo
```

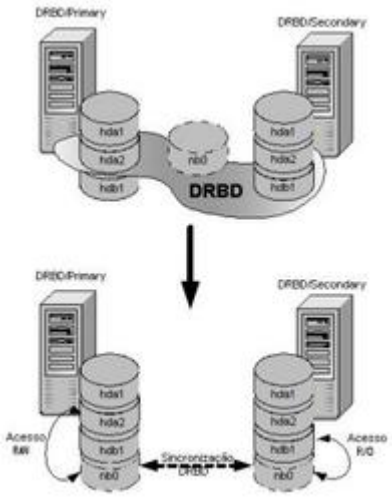
Na próxima vez que o sistema for inicializado escolha a opção referente ao novo kernel para inicializá-lo.

*O COMANDO **APT-GET INSTALL LINUX-SOURCE-2.6.15** IRÁ REALIZAR APENAS O DOWNLOAD DO KERNEL, SEM EFETUAR SUA COMPILAÇÃO OU INSTALAÇÃO.*

# LIÇÃO 7 - INSTALAÇÃO E CONFIGURAÇÃO DO DRBD

## DRBD

Como já foi dito anteriomente o DRBD (Distributed Replicated Block Device) é um módulo do kernel do Linux e mais alguns scripts responsáveis por espelhar dados gravados em disco. No nosso cluster a única função deste programa é replicar os dados, tudo que for escrito em uma determinada pasta do nodo principal será replicada no nodo secundário de forma automática. Sem esse programa se um nodo ficasse fora do ar, na hora que voltasse poderia haver a incompatibilidade de dados pois poderiam ter sidos gravadas informações no outro nodo durante esse intervalo de tempo. A figura abaixo ilustra o funcionamento do DRBD.




## INSTALANDO OS PRÉ-REQUISITOS

Depois de um longo tempo compilando o kernel, vamos à instalação do DRBD.

O DRBD poderia ser instalado através da ferramenta APT, mas para ajudar também quem não utiliza a distribuição Debian ou alguma outra derivada desta, vamos compilar e instalar através do pacote baixado da intenet. Baixe o pacote do endereço **http://oss.linbit.com/drbd/** , aparecerão várias opções mas a abordada aqui será a 0.7.18 por isso escolha o pacote drbd-0.7.18.tag.gz .

Para a instalação do DRBD existem alguns programas que são pré-requisitos, esses serão instalados pelo APT pois a instalação de cada pacote na mão tornaria o curso muito extenso, como eu disse anteriormente, nada impede que perguntas sejam feitas no fórum. Os pacotes são:

- libc6.1
- module-assistant

Para instalar essas dependências damos então o comando:

```
debian:~# apt-get install module-assistant libc6.1
```

## INSTALANDO O DRBD

Finalmente vamos instalar o DRBD, no diretório onde foi baixado o programa dê o comando para descompactar:

```
debian:/usr/src# tar zxvf drbd-0.7.18.tar.gz
```

Será criada então uma pasta chamada drbd-0.7.18, entre nela para fazer a compilação do programa.

```
debian:/usr/src# cd drbd-0.7.18
debian:/usr/src/drbd-0.7.18# cd drbd
debian:/usr/src/drbd-0.7.18/drbd# make clean ; make
 KDIR=/diretorio/onde/estão/as/fontes CC=gcc-3.x
```

Algumas coisas tem que ser notadas nesse último comando, a diretiva KDIR serve para indicar onde estão os fontes do seu kernel, no nosso caso eles devem estar em /lib/modules/2.6.15-cluster/build; outro ponto importante é a diretiva CC, esta serve para indicar qual compilador deve compilar o programa, este deve ser o mesmo usado na compilação do kernel, no meu caso foi o gcc-3.3. Levando em conta essas considerações o comando seria make clean ; make KDIR=/lib/modules/2.6.15-cluster/build CC=gcc-3.3 .

Se tudo ocorrer corretamente durante a compilação basta instalar.

```
debian:/usr/src/drbd-0.7.18/drbd# cd ..
debian:/usr/src/drbd-0.7.18/# make install
```

## UM OUTRO MÉTODO DE INSTALAÇÃO

Um outro método de instalação do drbd é utilizando o module-assistant. Com ele já instalado dê o comando:

```
debian:~#m-a update
```

Isso fará com que ele atualize os módulos. Agora é só instalar o módulo do drbd com o comando:

```
debian:~# m-a a-i drbd0.7-module-source
```

Talvez ele mande você instalar algum programa como bulid-essencial ou qualquer outra coisa pelo apt, instale. Depois de tudo instalado é só prosseguir para a configuração.

## CONFIGURANDO O DRBD

Se tudo ocorreu conforme o esperado durante a instalação então vamos à configuração. Como já foi dito anteriormente o funcionamento do DRBD é feito em cima de um dispositivo virtual de alto nível e um dispositivo de baixo nível, devemos criar o dispositivo de alto nível enquanto o dispositivo de baixo nível será uma partição onde os dados serão gravados. Para criar um dispositivo de alto nível:

```
debian:~# mknod /dev/drbd0 b 147 0
```

O comando mknod serve para criar arquivos especiais de bloco ou caracteres, este tipo de arquivo é utilizado para comunicação com o sistema operacional e não para a gravação de dados, esta será realizada na partição do disco rígido. O parâmetro /dev/drbd0 indica o nome e onde será criado o arquivo; o parâmetro b indica que o dispositivo é um arquivo especial de bloco com buffer; 147 e 0 indicam o maior e o menor número do dispositivo. Para mais informações sobre mknod e o parâmetros é só consultar o manual: "man mknod".

O dispositivo de bloco deve ser criado nos dois nodos do cluster. Para cada partição a ser espelhada deve ser criado um dispositivo (mknod /dev/drbd1 b 147 1, mknod /dev/drbd2 b 147 2, ... ), o DRBD consegue lidar com até cinco partições.

O arquivo de configuração do DRBD é o drbd.conf, nele ficam todos o parâmetros para a configuração do programa.

Vamos agora configurar o DRBD.

```
debian:~#nano /etc/drbd.conf
```

O drbd.conf fica mais ou menos assim:

```
resource drbd0 {
      protocol B;
      incon-degr-cmd "halt -f";
      disk {
            on-io-error detach;
      }
      syncer{
            rate 10M;
            group 1;
      }
      on debian1 {
            device /dev/drbd0;
            disk /dev/hda3;
            address 10.0.0.1:7789;
            meta-disk internal;
      }
      on debian2 {
            device /dev/drbd0;
            disk /dev/hda3;
            address 10.0.0.2:7789;
            meta-disk internal;
      }
}
```

## ENTENDENDO O DRBD.CONF

- resource
    - Seção referente a um dispositivo a ser espelhado;

- protocol
    - O protocolo usado pelo DRBD para controlar como os dados serão escritos no dispositivo secundário. Existem três protocolos:
        - A - assim que o dado for escrito no disco e enviado pela rede a operação de escria é considerada como realizada;
        - B - assim que o outro novo mandar um sinal dizendo que recebeu o dado enviado a operação de escrita é considerada como realizada;
        - C - assim que o outro nodo confirmou a escrita do dado no disco, a operação de escrita é considerada como feita. Cabe a cada um decidir o protocola a ser usado, o A é o mais leve, já o B e o C consumem muitos recursos no que diz respeito a rede e podem congestioná-la se a conexão entre os nodos não for dedicada;
- incon-degr-cmd
    - Executa o comando dado como parâmetro (no nosso caso halt -f) caso o nodo entre em modo degradado e a replicação dos dados fique inconsistente , opção a ser utilizada caso a consitência dos dados seja mais importante que a disponibilidade;
- disk
    - Esta seção contém os parâmetros relativos ao disco;
- on-io-error
    - Se o dispositivo de nível mais baixo reportar um erro então uma ação é executada, no nosso caso detach, que abandona o dispositivo de nível mais baixo e entra no modo disk less (sem disco);
- syncer
    - Esta seção é utilizada para discrever os parâmetros do daemon de sincronização do dispositivo;
- rate
    - Usado para limitar a banda que deve ser usada para a sincronização que acontece em background;
- group
    - A resincronização de todos os dispositivos de um grupo ocorre em paralelo, você deve evitar que a sincronização de dispositivos que tem dispositivos de baixo nível em um mesmo disco sincronizem em paralelo;
- on
    - Esta seção contém os parâmetros referentes a um determinado nó, o valor deste parâmetro deve ser o hostname da máquina, para ver esse valor

damos o comando hostname no terminal. No caso do drbd.conf acima o nodo primário tem hostname debian1 e o secundário debian2;

- device
  - É o dispositivo de alto nível criado anteriormente;
- disk
  - A partição onde serão gravados os arquivos;
- address
  - O endereço IP do nodo e a porta de comunicação utilizada;
- meta-disk
  - Local onde serão gravados os meta-dados, internal quer dizer que será na mesma partição onde serão gravados os dados, você pode fazer uma partição só para os meta-dados;

Mais informações sobre o drbd.conf podem ser encontradas em **http://www.drbd.org/fileadmin/drbd/doc/0.7.0/en/drbd.conf.html**.

## UTILIZANDO O DRBD

Depois que você configurou o arquivo drbd.conf e copiou para os dois nodos o DRBD já pode ser inicializado para isso damos o comando:

```
debian:~# /etc/init.d/drbd start
```

Isso deve ser feito nos dois nodos, para ver o status do DRBD podemos dar o comando:

```
debian:~# cat /proc/drbd
```

Aparecerá algo como:

```
version: 0.7.0 svn $Rev: 1442 $ (api:74/proto:74)
 0: cs:Connected st:Secondary/Secondary ld:Inconsistent
 ns:0 nr:0 dw:0 dr:0 al:0 bm:1 lo:0 pe:0 ua:0 ap:0
```

Como pode ser notado a conexão é entre secundário e secundário e os dados estão inconsistentes, para incializar a sincronização dos nodos e definir qual será o nodo primário devemos então dar o comando no nodo primário:

```
debian:~# drbdadm -- --do-what-I-say primary all
```

Aparecerá algo como:

```
drbd0: Resync started as SyncSource (need to sync 5000 KB [1250
 bits set]).
 version: 0.7.0 svn $Rev: 1442 $ (api:74/proto:74)
 0: cs:SyncSource st:Primary/Secondary ld:Consistent
 ns:9276 nr:0 dw:0 dr:9404 al:0 bm:2 lo:0 pe:915 ua:32 ap:0
```

```
[=========>..........] sync'ed: 50.0% (4380/5000)K
finish: 0:00:05 speed: 620 (620) K/sec
```

Essa saída significa que o DRBD está sincronizando os dados entre os nodos. O processo de sincronização pode demorar um pouco, a mudança do valor rate da seção syncer pode ajudar a aumentar a velocidade.

Vamos testar agora um pouquinho do DRBD.

Primeiramente com o DRBD inicializado e sincronizado vamos montar a partição em um local definido, devemos escolher uma pasta que será o ponto de montagem, algo como /mnt/dados.

```
debian:~# mkdir /mnt/dados
debian:~# mount /dev/drbd0 /mnt/dados
```

*O DIRETÓRIO **/MNT/DADOS** DEVE EXISTIR NOS DOIS NODOS.*

Note que o dispositivo a ser montado é o /dev/drbd0 e não a partição onde os dados serão gravados (/dev/hda3), o drbd0 será tratado como se fosse a partição original (hda3), tudo será gravado nesse dispositivo. Vamos listar agora esse novo diretório, provavelmente ele estará vazio.

```
debian:~# ls /mnt/dados
```

Vamos criar um arquivo qualquer:

```
debian:~# cd /mnt/dados
debian:/mnt/dados# touch teste.txt
```

Vamos listar o diretório novamente para verificar se o arquivo foi realmente criado:

```
debian:/mnt/dados# ls
 teste.txt
```

Agora vamos desmontar o dispositivo e parar o DRBD:

```
debian:/mnt/dados# umount /dev/drbd0
debian:/mnt/dados# /etc/init.d/drbd stop
```

Vamos tornar o nodo secundário em primário, montar o dispositivo e verificar que o arquivo que foi criado no outro nodo está devidamente replicado:

```
debian:~# drbdadm -- --do-what-I-say primary all
debian:~# mount /dev/drbd0 /mnt/dados
debian:~# ls /mnt/dados
 teste.txt
```

Uma coisa importante a ser notada é que obviamente os dados do dispositivo drbd0 não podem ser acessados se este não estiver montado, por esse motivo é impossível acessar os dados no nodo secundário pois o dispositivo não estará montado neste, montar o /dev/drbd0 no nodo secundário enquanto o mesmo estiver montado no primário pode

acarretar em corrompimento do sistema de arquivos, coisa que não é muito desejada (não monte nem no modo readonly).

## INTERPRETANDO O /PROC/DRBD

Quando você executa o comando:

```
debian:~# cat /proc/drbd
```

Aparece uma espécie de log.

```
version: 0.7.0 svn $Rev: 1442 $ (api:74/proto:74)
 0: cs:Connected st:Secondary/Secondary ld:Inconsistent
 ns:0 nr:0 dw:0 dr:0 al:0 bm:1 lo:0 pe:0 ua:0 ap:0
```

Esse log tem vários parâmetros:

- **cs** - conection state (estado da conexão) - esse parâmetro pode assumir os valores:
    1. **Unconfigured** - O dispositivo espera por configuração;
    2. **StandAlone** - Não está tentando conectar a um nó, pedidos de entrada e saída são passados apenas localmente;
    3. **Unconnected** - Estado transitório;
    4. **WFConnection** - Um dispositivo espera por configuração do outro lado;
    5. **WFReportParams** - Estado transitório, enquanto espera pelo primeiro pacote na nova conexção TCP;
    6. **Connected** (conectado) - Tudo está ocorrendo com deveria;
    7. **WFBitMap {S, T}** - Estado transitório enquanto começa a sincronização;
    8. **SyncSource** - Sincronização em progresso, o nó tem os dados certos;
    9. **SyncTarget** - Sincronização em progresso, o nó tem dados inconsistentes;
    10. **PausedSync {S,T}**- A sincronização do dispisitivo foi interrompida enquanto o dispositivo de maior prioridade estava resincronizando;
    11. **SkippedSync {S,T}** - Isso nuca deve ser visto, é apenas para desenvolvedores.
- **st** - state (estado), **Local/Remote** (Local/Remoto) - esse parâmetro pode assumir os valores:
    1. **Primary** - o nodo ativo, o meio de acessar o dispositivo;
    2. **Secundary** - o nodo passivo, não acessa o dispositivo, apenas espera os dados para serem espelhados;
    3. **Unknown** - sem função alguma;

- **Id** - local data consistency - parâmetro que mostra a consistência dos dados locais, **Consistent** ou **Inconsistent**.
- **ns** - network send (enviado pela rede);
- **nr** - network receive (recebido pela rede);
- **dw** - disk write (escrito no disco);
- **dr** - disk read(lido do disco);
- **al** - activity log updates (atualizações do registro de atividade);
- **bm** - bitmap updates (atualizações de bitmap);
- **lo** - reference count on local device (contagem de referência no dispositivo local)
- **pe** - pending (pendente);
- **ua** - unack'd, ainda esperando por ack;
- **ap** - application requests expecting io-completion (requisições da aplicação esperando por finalização de IO (entrada e saída)).

## FINALIZANDO

Antes de terminar adicione ao fstab o /dev/drbd0 para ser montado automaticamente, isso funcionará principalmente com o heartbeat.

```
debian:~# nano /etc/fstab
```

Adicione a linha:

```
/dev/drbd0 /mnt/dados ext3 defaults,noauto 0 0
```

Obs.: O espaço entre os parâmetros é uma tabulação, por isso essa linha deve ser digitada assim: /dev/drbd0-> pressione TAB ->/mnt/dados-> pressione TAB ->...; isso pode parecer óbvio para algumas pessoas, mas para outras não.

Finalmente o DRBD está todo configurado e pronto para ser utilizado, mas este sozinho não tem tanta utilidade pois o mesmo não é capaz, por exemplo, de inicializar o serviço drbd nodo secundário em caso de queda do primário (para isso utilizaremos o heartbeat).

O site do DRBD tem bastante informação e de qualidade, com alguns conteúdos em português, para acessar o mesmo: **http://www.drbd.org**.

*AO INICIALIZAR O SERVIÇO /ETC/INIT.D/DRBD, É NECESSÁRIO DEFINIR AS PRIORIDADES DOS NODOS PARA O ARMAZENAMENTO, PARA SOMENTE ENTÃO ESCREVER DADOS NO PONTO DE MONTAGEM E ESTES SEREM AUTOMATICAMENTE COPIADOS NO NODO SECUNDÁRIO.*

# LIÇÃO 8 - INSTALAÇÃO E CONFIGURAÇÃO DO HEARTBEAT

## HEARTBEAT

Agora que já temos o DRBD instalado nas duas máquinas e totalmente funcional, temos que fazer com que o cluster seja capaz de se auto-ajustar em caso de falha, ou seja, se o nodo primário vier a falhar o secundário tem de assumir todas as suas funções, inclusive a replicação dos dados que, como já foi dito anteriormente, o DRBD não é capaz de realizar esse failover sozinho.

Heartbeat significa batimento cardíaco, a função do programa Heartbeat é trocar pulsos entre os nodos para dizer que estes ainda estão vivos. Seguindo o nosso exemplo, o Heartbeat troca pulsos (heartbeats) entre os nodos através da interface ethernet dedicada entre esses dois nós, geralmente é recomendado que haja dois canais de comunicação entre os nodos, um serial e um ethernet, para caso um destes venha a falhar o outro permita a monitoração pelo Heartbeat. O grande problema de existir apenas uma interface de comunicação é que caso uma placa de rede, por exemplo, queime, ocorreria o que chamamos de Split Brain Syndrome (Síndrome de Cérebro Dividido), essa situação é totalmente indesejada. Se essa infeliz falha ocorrer enquanto de um lado o nodo secundário acha que o primário caiu e assume todos os serviços, do outro, o nodo primário percebe que o secundário caiu mas mesmo assim continua a disponibilizar os serviços, isso causará uma confusão enorme principalmente se houver uma base de dados e essa for atualizada nesse intervalo de tempo. Observando então esse problema notamos que não é nada recomendável testar o sistema retirando o cabo de rede ou disabilitando qualquer que seja a interface de comunicação.

## INSTALANDO OS PRÉ-REQUISITOS

O Heartbeat também poderia ser instalado através da ferramenta APT, mas pelo mesmo motivo do DRBD instalaremos a partir de pacotes. Baixe o pacote do endereço **http://linux-ha.org/download/index.html** , aparecerão várias opções mas a abordada aqui será a 1.2.4 por isso escolha o pacote heartbeat-1.2.4.tar.gz .

Para a instalação do Heartbeat existem alguns programas que são pré-requisitos, esses serão instalados pelo APT pois a instalação de cada pacote na mão tornaria o curso muito extenso, como eu disse anteriormente, nada impede que perguntas sejam feitas no fórum. Os pacotes são:

- iproute
- adduser
- libc6.1
- libc6
- libglib1.2
- libnet1
- libpils0

- libstonith0
- libatm1
- libglib2.0-dev
- libnet1-dev
- libglib-dev
- libltdl3-dev
- zlib1g-dev
- zlib1g

Algumas dessa ferramentas já devem estar instaladas como o iproute e adduser. Para instalar essas dependências damos então o comando:

```
debian:~# apt-get install iproute adduser libc6.1 libc6
 libglib1.2 libnet1 ibpils0 libstonith0 libatm1 libglib2.0-dev
 libnet1-dev libglib-dev libltdl3-dev zlib1g-dev
```

Os que não forem usuários debian podem ter alguns problemas, mas nada que um pouco de esforço e algumas perguntas não resolvam no fórum.

## INSTALANDO O HEARTBEAT

Vamos instalar então o Heartbeat. No diretório onde foi baixado o programa dê o comando para descompactar:

```
debian:/usr/src# tar zxvf heartbeat-1.2.4.tar.gz
```

Será criada então uma pasta chamada heartbeat-1.2.4, entre nela para fazer a compilação do programa.

```
debian:/usr/src# cd heartbeat-1.2.4
debian:/usr/src/heartbeat-1.2.4# ./configure
debian:/usr/src/heartbeat-1.2.4# make clean ; make CC=gcc-3.x
```

Antes de tudo devemos criar um grupo chamado haclient e um usuário hacluster.

```
debian:~# addgroup haclient
debian:~# adduser hacluster
debian:/usr/src/heartbeat-1.2.4# make install
```

Para evitar problemas, vamos utilizar o mesmo compilador do kernel e do DRBD. Em geral, a instalação do Heartbeat não cria muitos problemas.

Depois de ter instalado num dos nós é só instalar no outro, o usuários debian podem fazer uma coisa mais prática ainda, criar um pacote para instalar na outra máquina sem ter que recompilar. Para criar o pacote damos o comando:

```
debian:/usr/src/heartbeat-1.2.4# dpkg-buildpackage -us -uc –b
```

Serão criados vários pacotes .deb, basta copiá-los para o outro nodo e instalar com o comando:

```
debian:/usr/src/heartbeat-1.2.4# dpkg -i pacote_a_instalar.deb
```

## CONFIGURANDO O HEARTBEAT

Agora que o Heartbeat foi compilado e instalado falta apenas configurar os três scripts de configuração: ha.cf, haresources e authkeys.

Agora vá no diretório com os fontes e entre no diretório doc onde existem exemplos dos arquivos de configuração, você deve copiá-los para /etc/ha.d, crie essa pasta se ela não existir.

```
debian:~# cd /usr/src/heartbeat-1.2.4
debian:/usr/src/heartbeat-1.2.4# cd doc
debian:/usr/src/heartbeat-1.2.4/doc# cp ha.cf haresources
 authkeys /etc/ha.d/
```

## CONFIGURANDO O HA.CF

O primeiro arquivo a ser configurado é o ha.cf, nele ficam as configurações gerais que dizem respeito ao Heartbeat.

```
#Define o arquivo de debug, útil em algumas situações:
debugfile /var/log/ha-debug

#Define o arquivo com os logs, muito útil para verificar o
 funcionamento do programa:
logfile /var/log/ha-log

#Define o tempo entre os heartbeats em 2 segundos:
keepalive 2

#Um nodo é declarado morto após 30 segundos:
deadtime 30
#Tempo antes de enviar um warning de late heartbeat (heartbeat
 atrasado) para o arquivo de log:
warntime 10

#Como algumas redes demoram a funcionar após a reinicialização,
 esse parâmetro deve ser definido para tratar essa situação
 (deve ser pelo menos o dobro do deadtime):
initdead 60

#Configuração da rede:
mcast eth0:0 255.0.0.0 694 1 0

#Se essa opção estiver ativada, supondo que o nodo primário
 fique fora do ar, na hora que ele voltar todos os serviços
 deixarão o nodo secundário e voltarão para o primário:
```

```
auto_failback on

#Os nodos que fazem parte do cluster, a opção "uname -n" no
 terminal exibe o nome do nodo:
node debian1
node debian 2
```

Copie este arquivo para o outro nodo, o mesmo deve ser idêntico nos dois nodos.

Mais informações sobre a configuração do ha.cf: **http://www.linux-ha.org/ha.cf**.

## CONFIGURANDO O HARESOURCES

O haresources é o arquivo responsável por levantar os serviços as serem monitorados, no nosso caso o DRBD e o apache, este foi definido como o nosso servidor web.

Antes de mais nada vamos instalar o apache:

```
debian:~# apt-get install apache
```

Não será abordado o uso do apache neste curso pois o objetivo não é ensinar a montar um servidor e sim um sistema de alta disponibilidade. Para testar se o apache foi instalado corretamente simplesmente abra um browser qualquer (Firefox, Epiphany, Mozilla ...) e na barra de endereços digite "http://localhost", se tudo estiver certo aparecerá uma página de apresentação do apache. Depois que o apache foi instalado ele geralmente é inicializado, faça com que ele pare pois o Heartbeat que deve inicializar todos os programas a serem monitorados.

```
debian:~# /etc/init.d/apache stop
```

Agora finalmente vamos configurar o haresources.

```
debian:~# nano /etc/ha.d/haresources
```

Basta adicionar a linha:

```
debian1 IPaddr::10.0.0.100 drbddisk
 Filesystem::/dev/drbd0::/mnt/dados::ext3 apache
```

Essa linha diz ao Heartbeat qual o nodo (debian1), o ip do cluster (10.0.0.100) e os programas a serem inicializados (drbddisk e apache).

- **Obs1**.: O ip do cluster é o ip pelo qual o servidor será acessado, não deve ser definido em nenhum outro local, nem no /etc/network/interfaces.
- **Obs2**.: O drbddisk fica do diretório /etc/ha.d/resource.d/drbddisk, ele é responsável por incializar o DRBD e todos os procedimentos correlatos.
- **Obs3**.: Alguns parâmetros foram passados no que diz respeito ao sistema de arquivos, esses parâmetros foram passados para o drbddisk, atitude muito recomendada.

Copie este arquivo para o outro nodo, o mesmo deve ser idêntico nos dois nodos.

Mais informações sobre a configuração do haresources: **http://www.linux-ha.org/haresources**.

## CONFIGURANDO O AUTHKEYS

O último arquivo a ser configurado é o authkeys, nele ficam as informações que dizem respeito à autenticação. Podem ser escolhidos três métodos de autenticação:

- **sha1** - se você quer o método mais seguro sem se importar com o consumo da CPU, utilize esse método;
- **md5** - se a rede não é segura mas você quer economizar recursos da CPU, utilize esse método;
- **crc** - se o heartbeat está sendo executado em uma rede segura use este método, é o que exige menos recursos.

Para configurar este arquivo:

```
debian:~# nano /etc/ha.d/authkeys
```

Para crc adicione as linhas:

```
auth 2
2 crc
```

Para md5:

```
auth 1
1 md5 uma_chave_que_você_desejar
```

Para sha1:

```
auth 1
1 sha1 uma_chave_que_você_desejar
```

Devemos agora mudar a permissão do arquivo para 600:

```
debian:~# chmod 600 /etc/ha.d/authkeys
```

Copie este arquivo para o outro nodo do cluster.

## INICIANDO O HEARTBEAT

Depois de todo esse trabalho, todos os scripts configurados, vamos iniciar o Heartbeat (nessa etapa o drbd tem que estar iniciado):

```
debian1:~# /etc/init.d/heartbeat start
```

Para acompanhar os logs da inicialização:

```
debian1:~# cat /var/log/ha-log
```

É muito importante acompanhar os logs de inicialização pois eles dizem os erros que podem estar acontecendo.

Executando o comando ifconfig podemos ver que a interface que foi definida para ser o ip do cluster está levantada:

```
debian1:~# ifconfig
 eth0:0 Encapsulamento do Link: Ethernet Endereço de HW
 00:11:C8:86:AF:F4
 inet end.: 10.0.0.100 Bcast:10.0.0.255 Masc:255.255.255.0
 UP BROADCASTRUNNING MULTICAST MTU:1500 Métrica:1
 IRQ:193 Endereço de E/S:0x8800
```

Pronto, agora você tem um servidor web de alta disponibilidade quase completo, para testar o servidor basta digitar o ip do cluster na barra de endereços de qualquer navegador que se tudo estiver correto aparecerá a página inicial do apache.

Lembre-se que para testar a disponibilidade do sistema não é nada recomendado retirar o cabo de rede, tente algo como finalizar o heartbeat, desligar uma das máquinas ou qualquer outra coisa do tipo.

Uma configuração típica do arquivo ha.cf seria igual a:

```
debian1 IPaddr::10.0.0.100 drbddisk
 Filesystem::/dev/drbd0::/mnt/dados::ext3 apache
```

# LIÇÃO 9 - INSTALAÇÃO E CONFIGURAÇÃO DO MON

## MON

O último software para a configuração do cluster é o monitor MON, ele tem a função de verificar os serviços e o estado da rede e tomar uma atitude no caso de problemas, no nosso caso a atitude será o desligamento do Heartbeat. Uma situação que o MON agiria seria se, por exemplo, a placa de rede que comunica o nodo primário à rede externa queimasse ou por algum motivo parasse de responder à rede externa, nessa situação o MON derrubaria o Heartbeat, com isso na outra máquina o Heartbeat seria levantado e o serviço continuaria disponível.

O mon trabalha com alguns scripts, os monitores e os alertas, os monitores tem como função monitorar um dado serviço, os alertas executam alguma ação quando são devidamente invocados pelos monitores.

## INSTALANDO OS PRÉ-REQUISITOS

O MON pode ser baixado em ftp://ftp.kernel.org/pub/software/admin/mon/, a versão utilizada será a 0.99.2, escolha então o pacote mon-0.99.2.tar.gz. Antes de mais nada vamos intalar os pré-requisitos:

- libgcc1;
- libmon-perl;
- libtime-hires-perl;
- libtime-period-perl;
- perl;
- fping.

Agora é só intalar:

```
debian:~# apt-get install libgcc1 libmon-perl libtime-hires-perl
 libtime-period-perl perl fping
```

## INSTALANDO O MON

O primeiro passo para a instalação do programa é descompactar o arquivo baixado:

```
debian:/usr/src# tar zxvf mon-0.99.2.tar.gz
```

Vamos criar um diretório para o MON:

```
debian:/usr/src# mkdir /etc/ha.d/mon
```

Agora devemos copiar os arquivos do mon para esse diretório:

```
debian:/usr/src# cp -r /usr/src/mon-0.99.2 /etc/ha.d/mon/
```

Vamos agora editar o arquivo /etc/services:

```
debian:/usr/src# cd /etc
debian:/etc# nano services
```

Devemos adicionar as seguintes linhas no arquivo:

```
mon 2583/tcp
mon 2583/udp
```

## CONFIGURANDO O MON

O último passo para terminar a instalação do MON é a configuração. Para tal devemos editar o arquivo mon.cf. Um exemplo deste arquivo pode ser encontrado no diretório etc do diretório do mon (/etc/ha.d/mon/etc/exemple.cf), ele deve ficar parecido com esse:

```
cfbasedir = /etc/ha.d/mon/etc
alertdir = /etc/ha.d/mon/alert.d
mondir = /etc/ha.d/mon/mon.d
maxprocs = 20
histlength = 100
historicfile = /var/log/mon.log
randstart = 60s

hostgroup cluster 1.0.0.1 1.0.0.2 192.168.1.198

watch cluster
service fping
interval 1m
monitor fping.monitor –a
period wd {Mon-Fri}
alert mail.alert root@localhost
alert heartbeat.alert
alertevery 1h
period wd {Sat-Sun}
alert heartbeat.alert
alert mail.alert host@localhost
```

*FIQUE MUITO ATENTO QUANTO À LINHA QUE FOI PULADA ENTRE O HOSTGROUP E O WATCH, E RESPEITE TODA A IDENTAÇÃO, SEM ISSO PROVAVELMENTE O MON VAI ACUSAR ERROS COM O ARQUIVO DE CONFIGURAÇÃO.*

Depois de configurado este arquivo basta iniciar o MON:

```
debian:/etc/ha.d/mon# ./mon -f -c /etc/ha.d/mon/mon.cf -b
 /etc/ha.d/mon/
```

## HEARTBEAT.ALERT

Como pode ser visto no arquivo de configuração exemplo do MON, em caso de erro alguns alertas são chamados, entre eles o heartbeat.alert. A única função do script heartbeat.alert é desligar o Heartbeat em caso de falha, o conteúdo desse script é:

```
#!/usr/bin/perl
#
# Shutdown heartbeat
# derived from Jim Trocki's alert.template
#
# Jim Trocki, trockij@transmeta.com
# Sandro Poppi, spoppi@gmx.de
#
# Copyright (C) 1998, Jim Trocki
# 2001, Sandro Poppi
#
# This program is free software; you can redistribute it and/or
 modify
# it under the terms of the GNU General Public License as
 published by
# the Free Software Foundation; either version 2 of the License,
 or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be
 useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
 License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
 02111-1307 USA
#
use Getopt::Std;
getopts ("s:g:h:t:l:u");

#
# the first line is summary information, adequate to send to a
 pager
# or email subject line
#
#
# the following lines normally contain more detailed
 information,
```

```
# but this is monitor-dependent
#
# see the "Alert Programs" section in mon(1) for an explanation
# of the options that are passed to the monitor script.
#
$summary=<STDIN>;
chomp $summary;

$t = localtime($opt_t);
($wday,$mon,$day,$tm) = split (/\s+/, $t);

print <<EOF;

Alert for group $opt_g, service $opt_s
EOF

print "This alert was sent because service was restored\n"
if ($opt_u);

print <<EOF;
This happened on $wday $mon $day $tm
Summary information: $summary
Arguments passed to this script: @ARGV
Detailed information follows:

EOF

# shutdown heartbeat
system ("/etc/init.d/heartbeat stop");
```

Basta criar um arquivo contendo as linhas acima, salvar com o nome heartbeat.alert e adicionar ao diretório de scripts definido pelo parâmetro alertdir do arquivo de configuração mon.cf.

## FONTE

Centro de Difusão de Tecnologia e Conhecimento – CDTC

CDTC.org.br