Cluster de serviços e alta disponibilidade com Software Livre

Autor: Patrick Melo
Contato: patrickimelo3@gmail.com
Twitter
LinkedIn

Resumo

A dependência de sistemas computacionais se tornou visível nos dias de hoje, e a possibilidade de indisponibilidade dos serviços providos pelos sistemas se tornou intolerável. Já imaginou o tamanho do prejuízo para as empresas se o sistema da bolsa de valores sair do ar? Ou o sistema da folha de pagamento da sua empresa parar de funcionar no dia do seu pagamento? O objetivo desse projeto é construir um ambiente clusterizado e altamente disponível, utilizando software livre.

Introdução

HA (High availability) Alta disponibilidade - Técnica de projetar sistemas de forma que os serviços fornecidos por eles cumpram os requisitos de estarem acessíveis por um período mínimo definido por um contrato de nível de serviço ou SLA (Service Level Agreement).

Os sistemas computacionais podem ser divididos em quatro categorias:

Sistemas Gerais: São aqueles em que falhas e interrupções curtas de serviço são toleráveis desde que o sistema volte a funcionar posteriormente.

Sistemas Altamente Disponíveis: Disponibilidade crítica, medida a maneira de se determinar a qualidade dos serviços, sendo inaceitável a falha do sistema como um todo. Ex. Sistemas bancários, telecomunicações.

Sistemas Computacionais Críticos: Podem comprometer a segurança de pessoas ou ter um alto impacto econômico. Exemplos: Sistema Aéreo,Militar, Industriais.

Sistemas de Longa Vida: A confiabilidade é prioridade máxima e normalmente é impossível realizar manutenção não-planejada. Exemplos: Satélites, Controladores de Vôo Espacial.

Todo sistema possui uma taxa de confiabilidade, que é a probabilidade do mesmo realizar suas funções em dado período de tempo, utilizada para calcular o SLA. Bem comuns são o MTBF (Medium Time Between Failures) e o MTTR (Medium Time to Repair).

Downtime – Tempo que determinado serviço ou sistema ficou indisponível, podendo ocorrer de duas formas: planejada e não-planejada. Períodos não planejados são decorrentes de falhas não controladas e devem ser evitadas através de técnicas de alta disponibilidade. Períodos planejados são normalmente períodos de manutenção agendados de maneira a minimizar o

impacto no negocio e tem o objetivo de adicionar, modificar, reparar ou atualizar componentes do sistema.
O SLA pode ser estimado conhecendo-se o MTBF dos sistemas utilizados aplicando a formula:

Disponibilidade (%)	Downtime Anual	Downtime Mensal	Downtime Semanal
90%	36,5 dias	72 horas	16,8 horas
99%	3,65 dias	7,2 horas	1,68 horas
99,9%	8,76 horas	43,2 minutos	10,1 minutos
99,9999%	31,5 segundos	2,59 segundos	0,6 segundos

Causas de Indisponibilidade

Pesquisas realizadas em 2010 apontam os maiores causadores de indisponibilidade:

1. Falta de boas praticas de gerencia de mudanças.
2. Falta de boas praticas de monitoração dos componentes dos sistemas.
3. Falta de boas praticas de definição de requisitos.
4. Falta de boas praticas de procedimentos operacionais.
5. Falta de boas praticas em evitar falhas de rede.
6. Falta de boas praticas em evitar problemas internos com aplicações.
7. Falta de boas praticas em evitar falha de serviços externos.
8. Ambiente físico inadequado.
9. Falta de boas praticas de modelo de redundância de rede.
10. Falta de solução de backup.
11. Falta de boas praticas na escolha da localização do Datacenter.
12. Falta de redundância de infra-estrutura.
13. Falta de redundância de arquitetura de Storage.

Modelos de Alta disponibilidade

Um ambiente que contem computadores redundantes ou Nós, é chamado de cluster. Seu principio de funcionamento é que caso um dos nós falhe, o outro nó assumirá o lugar do outro de forma transparente até que seja realizado o reparo. A utilização de redundância visa eliminar a necessidade de intervenção humana no ambiente para que o mesmo continue em funcionamento em caso de falhas.

- **Redundância Passiva:** Nestes sistemas a alta disponibilidade é alcançada adicionando-se capacidade extra no projeto do sistema de forma que a falha de um componente reduza a performance do ambiente, porem o mesmo continue funcionando de maneira suficiente a atender seus objetivos.

- **Redundância Ativa:** Nestes sistemas a alta disponibilidade é atingida sem perca de performance, e envolve a utilização de mecanismos mais complexos e capazes de detectar a falha de componentes e reconfigurar automaticamente o ambiente.

Classificação de Nós em um cluster

O tamanho mais comum em um cluster de alta disponibilidade é o cluster de 2 nós, pois é o mínimo necessário para fornecer redundância, porem existem outras configurações mais complexas.

- **Ativo/Passivo:** Provê uma instancia completamente redundante do serviço em cada nó, que é ativada apenas em caso de falha no nó primário.

- **Ativo/Ativo:** Os nós compartilham os dados e podem operar de forma conjunta. Em caso de falha o trafego destinado ao nó que falhou é redirecionado para algum dos nós remanescentes ou balanceado por um balanceador existente na “frente” do cluster.

- **N+1:** Semelhante a configuração Ativo/Ativo, porem utiliza um nó extra, passivo, capaz de assumir todos os serviços de um nó que eventualmente falhe.

- **N+M:** Em alguns clusters é possível que apenas um nó extra não seja capaz de fornecer redundância suficiente. Nestes casos mais de um nó são mantidos inativos, sendo ativados de acordo com necessidade.

Comunicação entre os Nós de um cluster

Para o pleno funcionamento é necessário haver alguma forma de coordenação entre os nós, para que cada um “saiba” quais serviços estão ativos e em que membro. Assim é possível que serviços sejam iniciados e parados por membros específicos.

Essa comunicação é feita através do Stack de Clusterização, que é composto de duas partes:

- **Gerenciador de Recursos de Cluster(CRM):** Responsável por iniciar e parar o serviços que o cluster mantém.

- **Mensageiro:** É responsável pela comunicação entre os Nós do cluster, mantendo atualizadas as informações de processos e estados dos servidores.

LVS (Linux Virtual Server) - É uma solução de balanceamento de carga avançada para sistemas Linux. É um projeto Open Source começado por Wensong Zhang em maio de 1998. A missão do projeto é construir um servidor de alto desempenho e altamente disponível para Linux usando a tecnologia de clustering. Servidor virtual é um servidor altamente escalável e altamente disponível construído em um cluster de servidores reais. A arquitetura de cluster de servidor é totalmente transparente para os usuários finais, e os usuários interagem com o sistema de cluster, como se fosse apenas um servidor de alta performance virtual único.

O suporte ao LVS é parte do Kernel e sua parte principal é o código do ip_vs (IP Virtual Server), executado em um servidor denominado diretor do grupo LVS. Este diretor é implementado através do daemon *ldirectord*. O diretor age como um switch de camada 4, recebendo solicitações de conexão de um cliente e escolhendo um servidor para atender a solicitação. Um servidor backend é denominado *realserver* na terminologia do LVS.

O LVS será utilizado para balancear a carga dos acessos entre os servidores reais presentes no cluster, e caso todos os servidores reais fiquem indisponíveis o serviço será automaticamente migrado para o host do próprio LVS.

Ambiente Proposto:

Abaixo poderemos visualizar como ficará nosso ambiente clusterizado com os servidores reais, para isso utilizaremos o modelo LVS (NAT). A topologia NAT permite uma grande latitude na utilização de hardware existente, mas é limitado em sua capacidade de lidar com grandes cargas devido ao fato de que todos os pacotes passarão pelo roteador LVS.

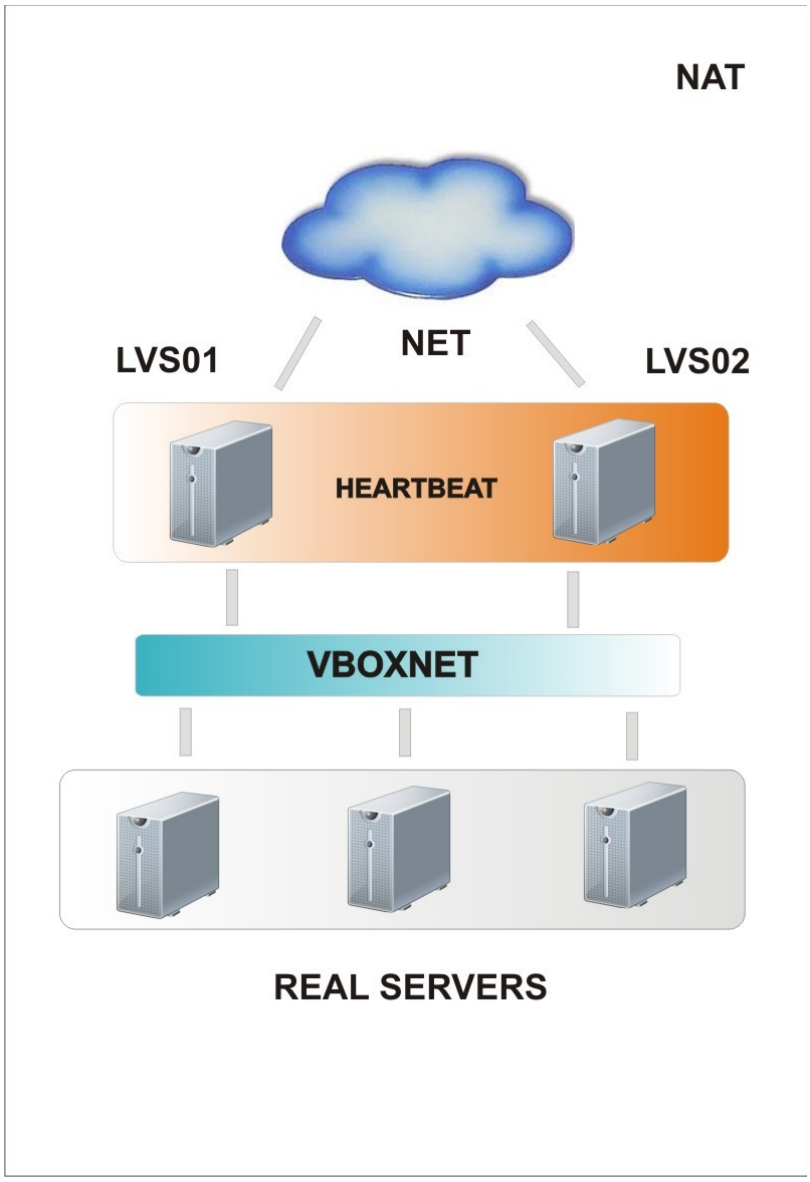

Figura 1. Ambiente proposto

Instalação do Ldirectord

1. Primeiro passo na montagem do LVS é instalar os aplicativos necessários para controlar o Diretor. Isso é feito, no debian, instalando o pacote do diretor LVS.

```
# apt-get install ldirectord
```

2. Copie o arquivo de configuração que está no diretório HOME do usuário para o diretório padrão do ldirectord.

```
# cp /home/files/ldirectord.cf /etc/ha.d/
```

3. Existe um arquivo de exemplo presente na documentação do ldirectord, para visualizá-lo execute:

```
# zcat/usr/share/doc/ldirectord/examples/ldirectord.cf.gz
```

4. Por fim vamos apontar a localização correta do arquivo de configuração no script de inicialização do ldirectord, edite o arquivo **/etc/init.d/ldirectord** na linha**:**

CONFIG_FILE="${CONFIG_FILE:=/etc/ldirectord.cf}"

Para:

CONFIG_FILE="${CONFIG_FILE:=/etc/ha.d/ldirectord.cf}"

5. No momento da elaboração do curso, a ultima versão disponível do ldirectord é a v1.186-ha, e possui um bug documentado que fazia com que o daemon terminasse inesperadamente quando havia um timeout da verificação do HTTP, aplique o patch para corrigir:

```
# patch –p0 /usr/sbin/ldirectord  /root/LVS/ldirectord.patch
```

Configuração do ldirectord

O arquivo que copiamos anteriormente é um modelo que deve ser adequado de acordo com o ambiente proposto, visualize o arquivo /etc/ha.d/ldirectord.cf.

```
# Tempo sem resposta, em segundos, até declarar um servidor fora do ar.
checktimeout=3

# Intervalo de verificação dos servidores, em segundos
checkinterval=1

# Recarregar o serviço em caso de alteração no arquivo de configuração
autoreload=yes

# Não remover da tabela de roteamento servidores que falharem, apenas definir seu peso como “0”
quiescent=yes

# Definição de um novo serviço do Cluster
virtual=ip.ip.ip.ip:80
        real=10.240.0.20:80 masq
```

```
        real=10.240.0.21:80 masq
        real=10.240.1.20:80 masq
        real=10.240.1.21:80 masq
        fallback=127.0.0.1:80 gate
        fallbackcommand="/etc/init.d/apache2"
        service=http
        request=".testpage"
        receive="test page"
        scheduler=rr
        protocol=tcp
        checktype=negotiate
```

Uma explicação mais detalhada de cada diretiva de serviço:

Virtual: Define um serviço indicado por endereço IP, porta e/ou marca de firewall.

Real: Define o realserver que irá atender requisições deste servidor virtual.
Note que a diretiva "**masq**" após o endereço, especifica que será realizado mascaramento de endereço.

Fallback: Endereço para o qual serão enviadas as solicitações caso haja falha em todos os Realservers. Normalmente utilizada como feedback para o cliente.

FallbackCommand: Comando que sera executado com parametro "start" caso todos os realservers estejam indisponiveis e com o parametro "stop" quando o primeiro realserver se tornar disponivel.

Service: Tipo de service que esta sendo verificado. Tipos validos são: dns,ftp,http,https,http_proxy,imap,imaps,ldap,mysql,nntp,Oracle,pgsql,pop,pops ,radius,simpletcp,sip,smtp, para serviços genéricos utilizamos o "simpletcp".

Request: URI do objeto que será solicitado para verificação.

Receive: Resposta (conteudo) que é esperada apos solicitor o objeto na solicitação acima.

Scheduler: Forma que a carga será distribuída entre os realservers do cluster. Alguns dos agendadores possíveis são:

- **RR:** Round Robin, distribui conexões igualmente entre os servidores a medida que são feitas.

- **WRR:** Weighted Round Robin, designa conexões para servidores de acordo com pesos (prioridades).

- **LC:** Least Connnections, envia novas requisições para servidores que estão no momento com pouca demanda.

- **WLC:** Weighted Least Connnections, envia novas requisições para servidores que estão no momento com pouca demanda, de acordo com pesos definidos.

Checktype: Tipo de verificação que será efetuada. Os tipos são:

- **Negociate:** Irá tentar uma conexão utilizando o protocolo definido em "Service" e tentar obter uma resposta válida para uma solicitação.

- **Connect:** Irá apenas tentar abrir uma conexão na porta indicada no servidor virtual.

- **Ping**: Irá utilizar o protocolo ICMP para verificar a disponibilidade dos realservers.

Ajustes necessários:

Será necessário editarmos algumas configurações referentes a rede das VM's, para que não haja conflito nos endereçamentos.

Validando LVS:

Para testarmos o funcionamento do diretor LVS devemos em primeiro lugar iniciar o diretor, execute:

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

Os Logs relatives ao funcionamento são armazenados por padrão em /var/log/ldirectord.log, visualize:

```
# tail –f /var/log/ldirectord.log
```

Redundância para o LVS

Apesar de termos os serviços balanceados entre vários servidores, devemos criar a redundancia para evitarmos o ponto de falha de um único servidor LVS, para isso utilizaremos o Heartbeat para configurar a clusterização.

Para maiores informações consulte:
http://www.linuxvirtualserver.org/

Heartbeat

Ferramenta que funcionará como gerente do cluster e pode ser chamado de o "coração" da infra-estrutura de alta disponibilidade, é o daemon responsavel pela comunicação entre os Nós do cluster.

Instalação do Heartbeat

Para a instalação iremos utilizar o repositorio:

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

Apos a instalação vamos copiar os arquivos de configuração criados em "/root/heartbeat/".

```
# cp /root/heartbeat/* /etc/ha.d/
```

Para que funcione a comunicação entre os Nós devemos criar credencial para autenticação. Adicione o conteudo no arquivo "/etc/ha.d/authkeys"

```
auth 1

1 md5 123456
```

O arquivo de configuração principal do heartbeat é o "ha.cf" e o significado das diretivas são:

Keepalive: Intervalo entre o envio de pacotes de verificação do Heartbeat.

Deadtime: Tempo para que o Heartbeat leva para informar que um nó está inativo depois da verificação.

Warntime: Tempo que o heartbeat leva para emitir um aviso de demora para a resposta do pacote keepalive.

Initdead: Semelhante ao Deadtime, porem é utilizado na inicialização do serviço e em valores maiores, permitindo o inicio em todos os nós.

Bcast: interface na qual o Heartbeat utilizará para o broadcast dos pacotes de verificação.

Node: Informa quais maquinas farão parte do cluster, é importante que a maquina seja capaz de resolver nomes e seu próprio hostname conste nela.

Crm: Especifica se será utilizado um gerenciador de recursos externo.

Auto_failback: Especifica se um recurso deverá retornar ao nó primário após uma recuperação de falha.

Debug e Logfile: Definem o arquivo que será utilizado para gravar as informações de log do serviço.

Para seu efetivo funcionamento devemos informar qual recurso será compartilhado entre os servidores, antes de tudo devemos configurar o VIP (ip virtual) utilizado pelo cluster. Edite o arquivo "/etc/ha.d/haresources"

```
lvs01 VIP ldirectord
```

Dessa forma o Heartbeat irá gerenciar o VIP(Ip Virtual), e o ldirectord alem de informar que o nó primário será o lvs01, onde deverão ser mantidos os serviços a menos que esteja indisponível.

Testando as configurações

No momento já possuímos uma estrutura básica montada, o Heartbeat irá gerenciar o LVS nos nós. Para os testes devemos adicionar o VIP no LVS e parar o serviço de LVS.

Edite o arquivo "/etc/ha.d/ldirectord.cf" e modifique para o ip do VIP:

```
virtual=ip.ip.ip.ip
```

Pare o serviço do LVS:

```
# /etc/init.d/ldirectord stopl
```

Inicie o serviço do Heartbeat:

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

Vamos verificar os arquivos de log do Heartbeat para verificar se os serviços irão subir corretamente:

```
# tail -f /var/log/ha-debug.log
```

Check se o serviço do LVS e o IP estão definidos no host:

```
# service ldirectord status & ifconfig eth1:0
```

Adicionando o segundo Nó

A configuração e instalação do Heartbeat será a mesma para o nó primario, após a instalação é necessario copiar a chave de autenticação entre os nós:

```
# scp -p /etc/ha.d/authkeys root@lvs02:/etc/ha.d/
```

Inicie o Heartbeat no segundo nó e verifique o log de erros:

```
# tail -f /var/log/ha-debug.log
```

Testes de funcionamento do cluster

Para testar a redundancia vamos parar o Heartbeat no nó primario e verificar se os serviços migrarão para o nó secundário.

Conclusão

Neste mini-curso conseguimos em um curto espaço de tempo criar um balanceamento de carga de serviços entre servidores e também configurar a alta disponibilidade dos mesmos, como sempre, não existe uma fórmula mágica para calcular o ponto ideal (é justamente por isso que existem consultores e analistas), mas é sempre prudente ter pelo menos um nível mínimo de redundância, nem que seja apenas um backup atualizado, que permita restaurar o servidor (usando outra máquina) caso alguma tragédia aconteça.

Referências

http://www.linuxvirtualserver.org/VS-NAT.html

http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.LVS-NAT.html

https://access.redhat.com/knowledge/docs/pt-BR/Red_Hat_Enterprise_Linux/5/html/Virtual_Server_Administration/ch-lvs-setup-VSA.html

http://www.vivaolinux.com.br/artigo/Alta-disponibilidade-com-Debian-Lenny-+-Heartbeat-+-DRBD8-+-OCFS2-+-MONIT-+-LVS

Curso 426 HA Alta disponibilidade (4linux)