VOL DAY III Apresenta:

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

Palestrante: Marcelo Gondim gondim@bsdinfo.com.br

Versão 1.3

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Objetivo: demonstrar como montar uma Alta Disponibilidade usando o FreeBSD 9.0 e o seu mais novo recurso que é o HAST (**Highly Available Storage**) que junto ao ZFS (**Zettabyte File System**) e o CARP (**Common Address Redundancy Protocol**).
- Vamos ver agora alguns dos recursos interessantes:
  - HAST: esse recurso seria o mesmo comparado ao DRBD no mundo GNU/Linux. Com o HAST é possível ter um cluster com 2 nós e os dados serem espelhados de um nó master para o nó slave de forma transparente independente do file system e aplicações porque ele usa o esquema de Block Level. Ele cria devices em /dev/hast/ em ambos os nós como se fossem discos e estes são espelhados para os devices no nó slave via TCP/IP.

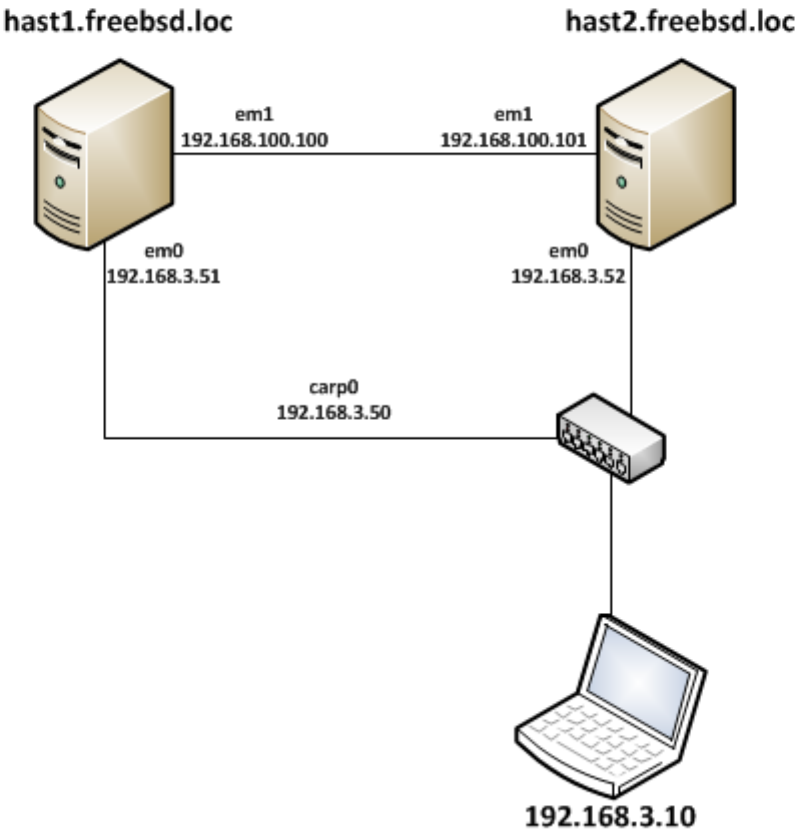
FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

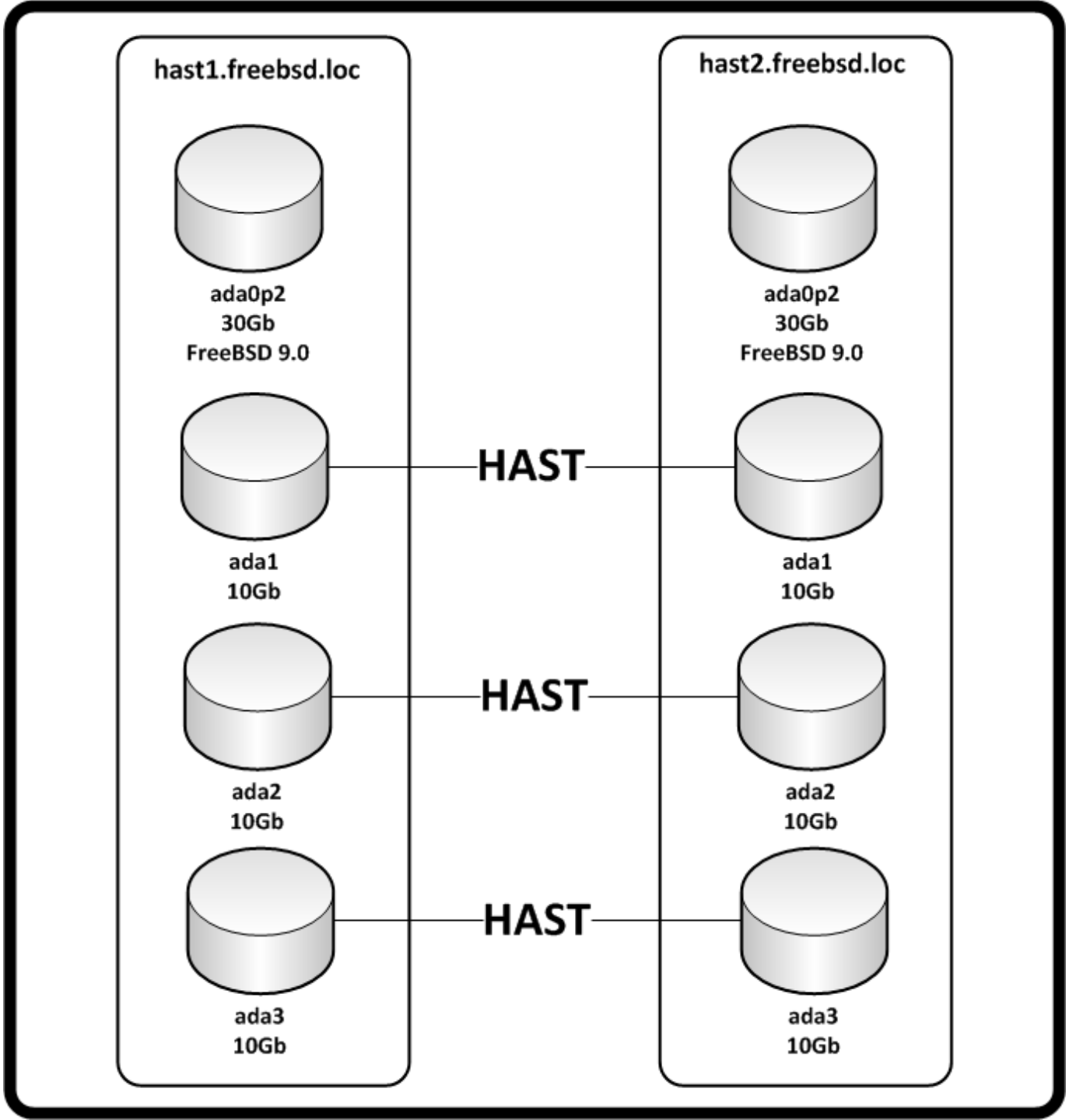
- Vamos ver agora alguns dos recursos interessantes:
  - ZFS: esse foi o file system utilizado por ser um excelente sistema e com diversos recursos que nenhum outro file system possui. Ele foi criado pela Sun Microsystems para os Sistemas Solaris e o OpenSolaris.
    - File System de 128 bits.
    - Pode armazenar até 256 quatrilhões de zettabytes (ZB). 1 zettabyte = 1 bilhão de TB.
    - Suporta multiplos volumes.
    - Integridade dos dados é a prioridade do ZFS.
    - RAID-Z.
    - ZFS Pool version 28.
    - Compressão de dados com as opções: on, off, lzjb(default), gzip(gzip-6) e gzip-N.
    - Quota por file system, user e group.
    - Deduplication – 2Gb de ram para cada 1Tb de storage.
    - Snapshot de volumes e file systems e Rollback.

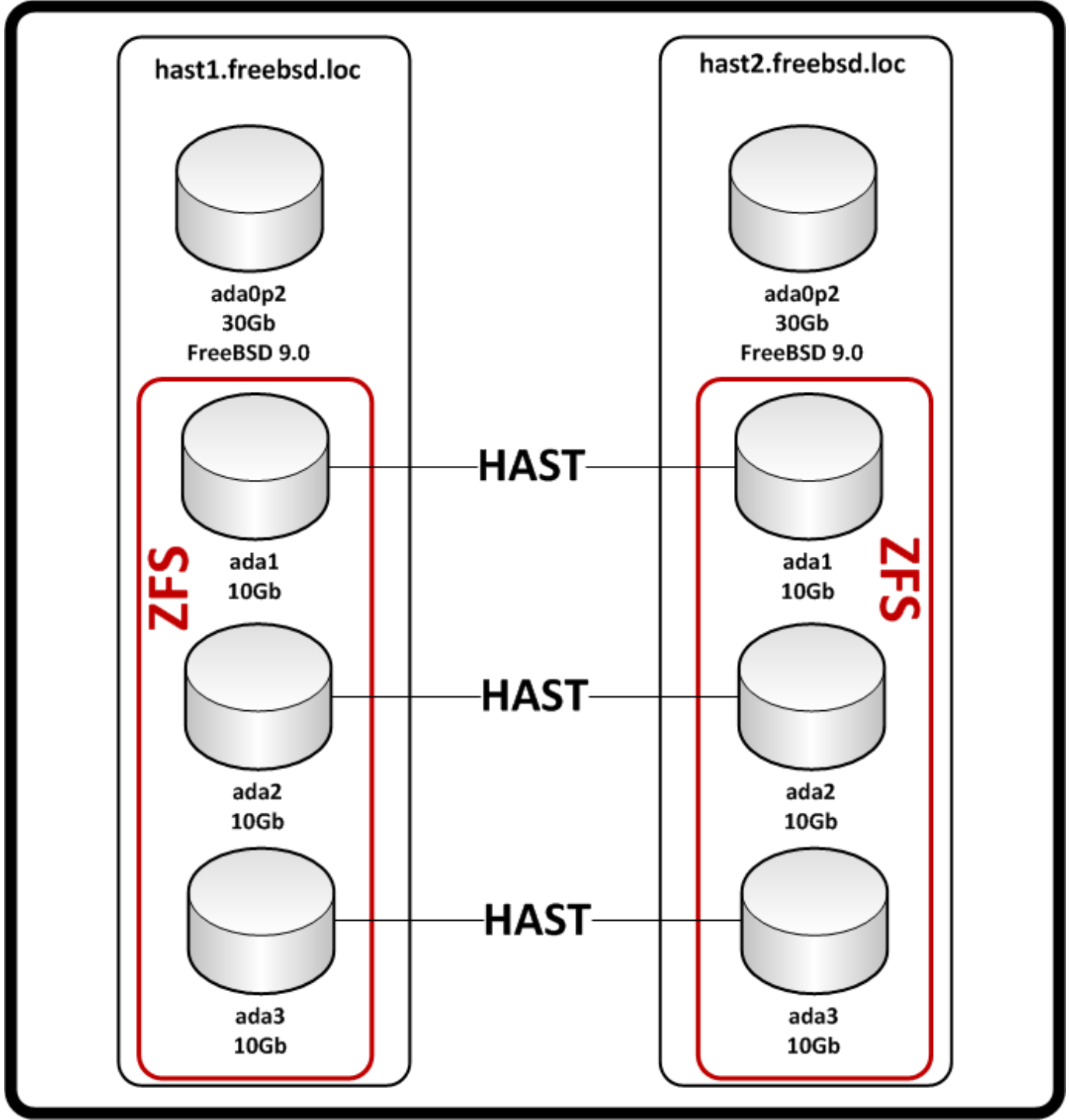
FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Vamos ver agora alguns dos recursos interessantes:
  - CARP: é um protocolo que permite compartilhar um mesmo IP entre diversos servidores. Podendo ser usado para Load Balancing ou High Availability. É uma alternativa livre ao Cisco HSRP (**Hot Standby Router Protocol**) e VRRP (**Virtual Router Redundancy Protocol**). O CARP fazendo uma analogia com a solução apresentada aqui seria o Heartbeat do GNU/Linux da parte de compartilhamento de IP.
- Para a nossa palestra tenho instalado um VirtualBox 4.1.8 com 2 VMs rodando FreeBSD 9.0 RELEASE. Cada VM tem 1Gb de ram, 2 network interfaces e 4 discos sendo 1 HD com 30Gb e mais 3 discos de 10Gb.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS




hast1.freebsd.loc
hast2.freebsd.loc
ada0p2
30Gb
FreeBSD 9.0
ada0p2
30Gb
FreeBSD 9.0
HAST
ada1
10Gb
ada1
10Gb
HAST
ada2
10Gb
ada2
10Gb
HAST
ada3
10Gb
ada3
10Gb


hast1.freebsd.loc
hast2.freebsd.loc
ada0p2
30Gb
FreeBSD 9.0
ada0p2
30Gb
FreeBSD 9.0
ZFS
ada1
10Gb
HAST
ada1
10Gb
ZFS
ada2
10Gb
HAST
ada2
10Gb
ada3
10Gb
HAST
ada3
10Gb

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Depois de conhecermos os ingredientes dessa salada, precisamos fazer as configurações necessárias para que tudo funcione. Aqui não falaremos sobre a instalação do FreeBSD e nem sobre HAST, CARP e ZFS à fundo. Para isso existem outras documentações e no final dessa palestra encontram-se links para os assuntos falados aqui bem como o link para o artigo de George Kontostanos, no qual essa palestra se baseou.

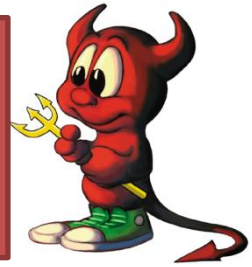
FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Iremos começar carregando o carp como módulo no boot dos 2 servidores. O carp também pode ser compilado dentro do kernel. Nesse caso não haveria a necessidade da configuração abaixo:
  - Adicione em **/boot/loader.conf** a seguinte instrução:

```
if_carp_load="YES"
```

- Agora configuraremos o **/etc/rc.conf** do nó master. O **rc.conf** é onde ficam as instruções para configuração geral do nosso FreeBSD. Nele configuramos desde interfaces de rede à serviços que irão rodar durante o boot do nosso sistema. Como referência existe o arquivo **/etc/defaults/rc.conf** que contém diversos exemplos de configurações usáveis e inclusive os valores default que são sobrescritos pelo **/etc/rc.conf**. Vejamos ele no próximo slide.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS



```
zfs_enable="YES"
hostname="hast1.freebsd.loc"
keymap="br275.iso.acc.kbd"
ifconfig_em0=" inet 192.168.3.51 netmask 255.255.255.0"
ifconfig_em1=" inet 192.168.100.100 netmask 255.255.255.0"
defaultrouter="192.168.3.254"
sshd_enable="YES"
cloned_interfaces="carp0"
ifconfig_carp0="inet 192.168.3.50 netmask 255.255.255.0 vhid 1 pass @#4tr051 advskew 0"
hastd_enable="YES"
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

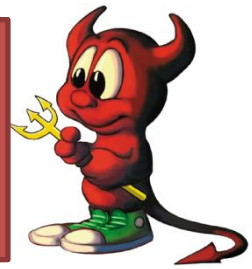
- Vamos à um breve descritivo deste arquivo tão importante:
  - **zfs_enable=“YES”** habilita carregar o ZFS durante o boot do sistema.
  - **hostname="hast1.freebsd.loc“** define o hostname do servidor.
  - **keymap="br275.iso.acc.kbd“** define o mapeamento do teclado.
  - **ifconfig_em0=" inet 192.168.3.51 netmask 255.255.255.0“** aqui configuramos nossa interface Intel em0 com IP e netmask. Essa será nossa interface pública utilizada no carp.
  - **ifconfig_em1=" inet 192.168.100.100 netmask 255.255.255.0“** a interface Intel em1 será a nossa interface dedicada ao HAST.
  - **defaultrouter="192.168.3.254“** aqui é configurado o gateway default do servidor.
  - **sshd_enable="YES“** essa linha habilita o serviço sshd durante o boot.
  - **cloned_interfaces="carp0”** cria a interface carp0 que nós usaremos para compartilhar um IP entre os nós.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Vamos ver um breve descritivo deste arquivo tão importante:
  - **ifconfig_carp0="inet 192.168.3.50 netmask 255.255.255.0 vhid 1 pass @#4tr051 adv skew”** aqui configuramos a interface carp0 com o IP à ser compartilhado e com a senha @#4tr051 usada para autenticação entre os nós.
  - **hastd_enable="YES“** por último e não tão menos importante habilitamos o serviço HAST no boot.

- No segundo nó faremos praticamente a mesma configuração e só mudaremos os IPs envolvidos e o hostname. No próximo slide a configuração já ajustada.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS



```
zfs_enable="YES"
hostname="hast2.freebsd.loc"
keymap="br275.iso.acc.kbd"
ifconfig_em0=" inet 192.168.3.52 netmask 255.255.255.0"
ifconfig_em1=" inet 192.168.100.101 netmask 255.255.255.0"
defaultrouter="192.168.3.254"
sshd_enable="YES"
cloned_interfaces="carp0"
ifconfig_carp0="inet 192.168.3.50 netmask 255.255.255.0 vhid 1 pass @#4tr051 advskew 0"
hastd_enable="YES"
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Vamos agora dar nomes aos bois do nosso projeto. Usando o arquivo **/etc/hosts**, daremos nomes aos IPs que iremos usar no **HAST**. Ambos os nós devem ter essas mesmas entradas.

```
192.168.100.100          hast1.freebsd.loc hast1
192.168.100.101          hast2.freebsd.loc hast2
192.168.3.51             storage1.hast.test storage1
192.168.3.52             storage2.hast.test storage2
192.168.3.50             storage.hast.test storage
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Agora faremos a configuração do **HAST**. A configuração apresentada aqui será a mesma também nos 2 nós. O arquivo de configuração é o **/etc/hast.conf**.
- Só relembrando: nós separamos 3 discos em cada nó, cada disco com 10Gb de tamanho. Cada disco do nó master será sincronizado com o seu disco de referência no nó slave, ou seja, cada bit gravado em um nó será transferido via TCP/IP para o outro nó e gravado lá. O **HAST** não depende de nada que esteja gravado no disco pois todo seu conteúdo será transferido do disco master para o disco slave.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

```
timeout 5

resource disk1 {
        on hast1 {
                local /dev/ada1
                remote hast2
        }
        on hast2 {
                local /dev/ada1
                remote hast1
        }
}

resource disk2 {
        on hast1 {
                local /dev/ada2
                remote hast2
        }
        on hast2 {
                local /dev/ada2
                remote hast1
        }
}

resource disk3 {
        on hast1 {
                local /dev/ada3
                remote hast2
        }
        on hast2 {
                local /dev/ada3
                remote hast1
        }
}
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Cada disco é um recurso e eu preciso referenciá-lo no arquivo tanto para o nó master quanto para o nó slave.
- No exemplo abaixo pegamos apenas um recurso que é o disk1 e definimos que no servidor hast1 o primeiro disco que usaremos é o: **local /dev/ada1** e como remoto o hast2.
- No servidor hast2 o primeiro disco será o **/dev/ada1** local e o remoto sendo o servidor hast1.

```
resource disk1 {
        on hast1 {
                local /dev/ada1
                remote hast2
        }
        on hast2 {
                local /dev/ada1
                remote hast1
        }
}
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Após configurarmos o HAST nos 2 nós, vamos iniciá-los:

```
hast1#/etc/rc.d/hastd start
```

```
hast2#/etc/rc.d/hastd start
```

- Nesse momento nós iniciamos o serviço mas nada ainda está sendo feito. Vamos precisar definir quem será inicialmente o master e quem será o slave.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Nesse momento vamos inicializar os recursos, criá-los e definir o master:

```
hast1#hastctl role init disk1
hast1#hastctl role init disk2
hast1#hastctl role init disk3
hast1#hastctl create disk1
hast1#hastctl create disk2
hast1#hastctl create disk3
hast1#hastctl role primary disk1
hast1#hastctl role primary disk2
hast1#hastctl role primary disk3
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Nesse momento vamos inicializar os recursos, criá-los e definir o slave:

```
hast2#hastctl role init disk1
hast2#hastctl role init disk2
hast2#hastctl role init disk3
hast2#hastctl create disk1
hast2#hastctl create disk2
hast2#hastctl create disk3
hast2#hastctl role secondary disk1
hast2#hastctl role secondary disk2
hast2#hastctl role secondary disk3
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Para vermos o status de como está o **HAST** nos 2 nós usamos também o **hastctl**. Com ele podemos checar se o sincronismo foi completado, quem é o primary (master), quem é o secondary (slave), qual o tipo de replicação, estatísticas, etc
- Tudo muito simples e muito bem documentado bastando fazer um: **man hast.conf** e **man hastctl**.
- Nos próximos slides veremos o resultado do comando: **hastctl status** nos 2 servidores.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

```
hast1# hastctl status
disk1:
  role: primary
  provname: disk1
  localpath: /dev/ada1
  ...
  remoteaddr: hast2
  replication: fullsync
  status: complete
  dirty: 0 (0B)
  ...
disk2:
  role: primary
  provname: disk2
  localpath: /dev/ada2
  ...
  remoteaddr: hast2
  replication: fullsync
  status: complete
  dirty: 0 (0B)
  ...
disk3:
  role: primary
  provname: disk3
  localpath: /dev/ada3
  ...
  remoteaddr: hast2
  replication: fullsync
  status: complete
  dirty: 0 (0B)
  ...
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

```
hast2# hastctl status
disk1:
  role: secondary
  provname: disk1
  localpath: /dev/ada1
  ...
  remoteaddr: hast1
  replication: fullsync
  status: complete
  dirty: 0 (0B)
  ...
disk2:
  role: secondary
  provname: disk2
  localpath: /dev/ada2
  ...
  remoteaddr: hast1
  replication: fullsync
  status: complete
  dirty: 0 (0B)
  ...
disk3:
  role: secondary
  provname: disk3
  localpath: /dev/ada3
  ...
  remoteaddr: hast1
  replication: fullsync
  status: complete
  dirty: 0 (0B)
  ...
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Agora que já temos o **HAST** rodando e fazendo o seu trabalho de Alta Disponibilidade podemos partir para o ZFS. Sim porque até o momento não temos qualquer file system rodando sobre ele e por isso ainda não possui qualquer utilidade para nós.
- Podemos criar um pool **ZFS** com um disco apenas ou mirror ou o nosso caso para fazer um RAID onde teremos mais espaço, performance e segurança. No **ZFS** temos o **RAIDZ** que seria o RAID5 melhorado e este pode usar 1, 2, 3 para referenciar as paridades como por exemplo: **RAIDZ1**, **RAIDZ2** e **RAIDZ3**. O número mínimo de discos para se usar o **RAIDZ** são 2 e o máximo recomendado são até 9 discos. O recomendado para um ganho de performance são de 3 até 9 discos. Faremos isso apenas no nó master porque ele será automaticamente replicado para o slave. **RAIDZ2** seria o **RAID6**.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

```
hast1# zpool create zhast raidz1 /dev/hast/disk1 /dev/hast/disk2 /dev/hast/disk3
hast1# zpool status zhast
 pool: zhast
 state: ONLINE
 scan: none requested
 config:

        NAME            STATE     READ WRITE CKSUM
        zhast           ONLINE       0     0     0
          raidz1-0      ONLINE       0     0     0
            hast/disk1  ONLINE       0     0     0
            hast/disk2  ONLINE       0     0     0
            hast/disk3  ONLINE       0     0     0
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- No primeiro comando nós criamos o pool chamado **zhast**, escolhemos usar o raidz1 e usamos os 3 discos gerados pelo **HAST** de 10Gb cada **(/dev/hast/disk1**, **/dev/hast/disk2** e **/dev/hast/disk3**). Como temos uma paridade ficaremos com apenas uns 19Gb para brincarmos.
- Um pool não tem muita vantagem para nós já que não podemos usar os recursos excelentes do **ZFS** e por isso vamos criar os file systems dentro do **ZFS**. Vamos criar o **"data"** dentro do pool **zhast**.
- O **/zhast/data** será nosso file system com as seguintes características: **noexec**, **nosuid**, **usando compressão gzip -9** e **com quota de 5Gb de espaço**.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Dentro do **/zhast/data** criaremos um diretório chamado **www** e nele colocarei um script php para exibir uma página **vermelha** com o **hostname** se caso o **hostname** retornado for **hast1.freebsd.loc**. Se o **hostname** for outro então a página mudará de cor para **verde** e mostrará o **hostname** atual. Será o nosso exemplo quando mudarmos do nó master para o nó slave.
- Tudo que for gravado no nosso storage que é o **/zhast** será replicado para o nosso servidor slave.
- Não será falado sobre a configuração do Apache pois sairia do escopo da palestra.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

```
hast1# zfs create -o compression=gzip-9 -o exec=off -o setuid=off -o quota=5G zhast/data
hast1# df -h
Filesystem     Size    Used   Avail Capacity  Mounted on
/dev/ada0p2     27G    2.7G     22G    11%    /
devfs          1.0k    1.0k      0B   100%    /dev
zhast           19G     41k     19G     0%    /zhast
zhast/data     5.0G     40k      5G     0%    /zhast/data
hast1# mkdir /zhast/data/www
hast1# cp /root/index.php /zhast/data/www/
hast1# cat /zhast/data/www/index.php
<?php

if ( gethostname() == "hast1.freebsd.loc" )
   $cor="#FF4848";
else
   $cor="#01F33E";

?>

<html><body bgcolor=<?php echo $cor; ?> text="#ffffff"><h1><center><?php echo gethostname(); ?></center></h1></body></html>

hast1#
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Agora que temos HAST, CARP e ZFS rodando, precisaremos configurar o sistema de maneira que quando o master caia, o slave assuma o papel de master e levante nosso serviço Apache do outro lado.
- Para esse trabalho usaremos o serviço **devd** e um script que será executado pelo **devd** e ambos serão configurados nos 2 servidores da mesma forma. O **devd** é um serviço que checa a mudança de estado em devices, no nosso caso checaremos as mudanças no device do **CARP** e através dele executaremos o nosso script. No **/etc/devd.conf** de ambos os servidores acrescente as linhas do próximo slide;

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

```
notify 30 {
        match "system" "IFNET";
        match "subsystem" "carp0";
        match "type" "LINK_UP";
        action "/usr/local/bin/failover master";
};

notify 30 {
        match "system" "IFNET";
        match "subsystem" "carp0";
        match "type" "LINK_DOWN";
        action "/usr/local/bin/failover slave";
};
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Chegamos na última parte que faltava para que a Alta Disponibilidade funcione por completa. Nessa parte criamos o script que será rodado pelo serviço devd e que fará com que quando o nó master cair, o nó slave assuma como sendo master. Esse script se chamará **failover** e será o mesmo nos 2 servidores em **/usr/local/bin/**.

```
#!/bin/sh

# Original script by Freddie Cash <fjwcash@gmail.com>
# Modified by Michael W. Lucas <mwlucas@BlackHelicopters.org>
# and Viktor Petersson <vpetersson@wireload.net>
# Modified by George Kontostanos <gkontos.mail@gmail.com>

# The names of the HAST resources, as listed in /etc/hast.conf
resources="disk1 disk2 disk3"

# delay in mounting HAST resource after becoming master
# make your best guess
delay=3

# logging
log="local0.debug"
name="failover"
pool="zhast"

# end of user configurable stuff

case "$1" in
        master)
                logger -p $log -t $name "Switching to primary provider for ${resources}."
                sleep ${delay}

                # Wait for any "hastd secondary" processes to stop
                for disk in ${resources}; do
                        while $( pgrep -lf "hastd: ${disk} \(secondary\)" > /dev/null 2>&1 ); do
                                sleep 1
                        done

                        # Switch role for each disk
                        hastctl role primary ${disk}
                        if [ $? -ne 0 ]; then
                                logger -p $log -t $name "Unable to change role to primary for resource ${disk}."
                                exit 1
                        fi
                done

                # Wait for the /dev/hast/* devices to appear
                for disk in ${resources}; do
                        for I in $( jot 60 ); do
```

```
                        [ -c "/dev/hast/${disk}" ] && break
                        sleep 0.5
                done

                if [ ! -c "/dev/hast/${disk}" ]; then
                        logger -p $log -t $name "GEOM provider /dev/hast/${disk} did not appear."
                        exit 1
                fi
        done

        logger -p $log -t $name "Role for HAST resources ${resources} switched to primary."


        logger -p $log -t $name "Importing Pool"
        # Import ZFS pool. Do it forcibly as it remembers hostid of
        # the other cluster node.
        out=`zpool import -f "${pool}" 2>&1`
        if [ $? -ne 0 ]; then
            logger -p local0.error -t hast "ZFS pool import for resource ${resource} failed: ${out}."
            exit 1
        fi
        logger -p local0.debug -t hast "ZFS pool for resource ${resource} imported."
        /usr/local/etc/rc.d/apache22 onestart

;;

slave)
        logger -p $log -t $name "Switching to secondary provider for ${resources}."

        # Switch roles for the HAST resources
        zpool list | egrep -q "^${pool} "
        if [ $? -eq 0 ]; then
                # Forcibly export file pool.
                out=`zpool export -f "${pool}" 2>&1`
                 if [ $? -ne 0 ]; then
                        logger -p local0.error -t hast "Unable to export pool for resource ${resource}: ${out}."
                        exit 1
                 fi
                logger -p local0.debug -t hast "ZFS pool for resource ${resource} exported."
        fi
        for disk in ${resources}; do
                sleep $delay
                hastctl role secondary ${disk} 2>&1
```

```
                        if [ $? -ne 0 ]; then
                                logger -p $log -t $name "Unable to switch role to secondary for resource ${disk}."
                                exit 1
                        fi
                        logger -p $log -t $name "Role switched to secondary for resource ${disk}."
                done
                /usr/local/etc/rc.d/apache22 onestop
        ;;
esac
```

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- O que esse script faz na prática é isso abaixo:

```
hast1# zpool export zhast
hast1# hastctl role secondary disk1
hast1# hastctl role secondary disk2
hast1# hastctl role secondary disk3
```

```
hast2# hastctl role primary disk1
hast2# hastctl role primary disk2
hast2# hastctl role primary disk3
hast2# zpool import zhast
```

- O que adicionei à mais no script foram o start e o stop do Apache, ou seja, quando for master inicia o Apache e quando for slave pára o Apache. Fique à vontade com sua criatividade.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Vamos melhorar nosso HA? Com o que fizemos até agora se o master cair, o slave assumirá. Mas e quando a máquina que era master voltar? Do jeito que configuramos os servidores ficarão invertidos pois quem era master será slave e quem era slave será o master. Se essa for a intenção então paramos por aqui mas e se quisermos que quando a máquina que estava parada retorne à ser master?
- Como muitas coisas no FreeBSD isso também não deixaria de ser simples bastando alterar apenas 2 coisas. No próximo slide faremos essas 2 pequenas alterações.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

- Para que isso funcione precisamos adicionar um parâmetro em **/etc/sysctl.conf** nos 2 servidores:

  net.inet.carp.preempt=1

- O parâmetro acima diz ao CARP que se a máquina master voltar ela deve assumir o seu papel e a outra que estava como master temporariamente deverá voltar à ser slave.
- Também precisamos alterar o parâmetro **advskew** na máquina slave para um valor mais alto tipo 100:

  ifconfig_carp0="inet 192.168.3.50 netmask 255.255.255.0 vhid 1 pass @#4tr051 advskew 100“

- O menor advskew sempre será o master.

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

Parte prática da palestra

FreeBSD e Alta Disponibilidade com HAST + CARP + ZFS

Perguntas?

- Links relacionados com essa palestra:
  - http://www.aisecure.net/2012/02/07/hast-freebsd-zfs-with-carp-failover/
  - http://www.freebsd.org/doc/handbook/carp.html
  - http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/filesystems-zfs.html
  - http://wiki.freebsd.org/HAST
  - http://www.tech-recipes.com/rx/1446/zfs_ten_reasons_to_reformat_your_hard_drives/
  - http://hub.opensolaris.org/bin/view/Community+Group+zfs/faq