Implementing Filesystem Allocation Limits On FreeBSD Jails

FreeBSD jails are very powerful indeed. While it is rather easy to setup a jail for encapsulation of single services (let's think of it as a more luxury chroot environment then), it's also no big thing to adapt them to create 'virtual' servers similar to what Virtuozzo and OpenVZ for Linux try to achieve.

Obviously FreeBSD jails were not designed for this in the first place, probably the only reason why the implementation lacks some basic resource control functionality. This may of course change in the (not so distant) future, in the meantime we need to get along with the the limitations placed upon us.

This article as the first in a series covers a simple concept to implement file system allocation limits for FreeBSD jails in a virtual server environment.

What is this about?

As outlined above I will talk about setting up file system allocation limits. This is to restrict the maximum disk space a single jail (or virtual server if we go by that name) may allocate on the host system.

The basic idea is to use as much of the functionality already available in today's FreeBSD 6.1-release.

At the end we'll have jails that have their own encapsulated root filesystem which is dynamically (un-)mounted on jail startup/shutdown. Exceeding pre-allocated disk space will not be possible anymore using this approach, hardening the host against race conditions.

What is this not about?

This article is not about filesystem quota, though in theory they should work on top of the approach given.

Prerequisites

This has been tested on FreeBSD 6.1-RELEASE on Sparc64 but should work on other platforms, too.

Patching mdmfs

This approach will use functionality already available by today: mdmfs.

mdmfs will allow creation of so called vnode backed (or swap- or malloc-backed, there are actually three ways for this) memory devices (in other words a "RAM"-disk).

The idea is to mount a vnode-backed filesystem (which is actually stored inside a physical file on your hard drives) and place the jail's root filesystem inside.

One caveat is that mdmfs will format the vnode's loopback file so there's is some patching involved.

Luckily the changes necessary are already available in FreeBSD 6-current and they can easily be backported to FreeBSD 6.1-RELEASE and possibly also elder releases.

Create a new temporary build directory first:

#mkdir /root/mount_md

#cd /root/mount_md

Now get the source code from CVS. Maybe it's easiest to get this particular release through WebCSV at http://www.freebsd.org/cgi/cvsweb.cgi/~checkout~/src/sbin/mdmfs/mdmfs.c?rev=1.27.

Save this file to your previously created build directory.

If you have wget at hands you can also download it directly.

#wget --user-agent='Mozilla/5.0' 'http://www.freebsd.org/cgi/cvsweb.cgi/~checkout~/src/sbin/mdmfs/mdmfs.c?rev=1.27' --output-document=/root/mount_md/mount_md.c

Compile the source:

#gcc /root/mount_md/mount_md.c -o /root/mount_md/mount_md

Then copy the file to some location you like, eg. /usr/sbin.

#cp /root/mount_md/mount_md /usr/sbin/mount_md

Make sure you call the file as given in the example. It won't work otherwise.

Special care must be taken that you DO NOT replace your existing mdmfs binary file by this new version. DO NOT overwrite it. Do not rename this patch to mdmfs. Use the names provided in this example instead.

Create/Update Jail Directory Structure

You may already have an existing directory structure to store you jail virtual servers. Some changes may be necessary according to your local setup. I'm referring to my own setup as an example.

I have a directory structure below /mnt/devname/jails where the virtual servers are stored in subdirectories named by their IP address. I use a host with the IP 192.168.0.1, which gives me this directory structure:

/mnt/r5_vol1/jails/192.168.0.1

Inside of this directory I have two files:

```
#ls -l /mnt/r5_vol1/jails/192.168.0.1
total 1049106
-rw-r--r--  1 root  wheel         61 May 31 22:36 fstab
-rw-r--r--  1 root  wheel  1073741824 Jun 22 00:32 rootfs.volume
```

The 'fstab' file contains mount information for the jail while the rootfs.volume will serve as the loopback filesystem container.

The loopback filesystem container will be mounted below /var/jails/192.168.0.1.

Create the filesystem container

Now it is time to create the filesystem container, in other words the file that will host the jail's root filesystem.

You will use 'dd' to achieve this. In the example a container file of 1 GB will be created:

#dd if=/dev/zero of=/mnt/r5_vol1/192.168.0.1/rootfs.volume bs=1m count=1024

Configure a vnode using mdconfig, this will give the id of the vnode (md3 in the example).

#mdconfig -a -t vnode -f /mnt/r5_vol1/192.168.0.1/rootfs.volume
md3

Create the file system:

#newfs /dev/md3

Mount the container file:

#mount /dev/md3 /var/jails/192.168.0.1

Install the FreeBSD userland into the loopback filesystem as you would do usually. Umount the loopback device afterwards.

#umount /var/jails/192.168.0.1

Then remove the md vnode unit (md3 in the example will give '-u 3'):

#mdconfig -d -u 3

Setup Jail's fstab

To have your jail regurlary use the loopback container as it's root device you need to create a fstab for it. This is actually the fstab file I wrote about above.

Add these lines to it (you need both):

md /var/jails/192.168.0.1 md rw,-P,-F/mnt/r5_vol1/jails/192.168.0.1/rootfs.volume
md /var/jails/192.168.0.1 ufs rw,noauto

Update Jail Settings In rc.conf

Finally you must tweak your jail settings in rc.conf. This is an example how I did it.

192.168.0.1 jail
#
jail_192_168_0_1_rootdir="/var/jails/192.168.0.1"
jail_192_168_0_1_hostname="example.phunsites.net"
jail_192_168_0_1_ip="192.168.0.1"
jail_192_168_0_1_interface="hme0"
jail_192_168_0_1_devfs_enable="YES"
jail_192_168_0_1_mount_enable="YES"
jail_192_168_0_1_fstab="/mnt/r5_vol1/jails/192.168.0.1/fstab"

Conclusion

This is a way how to enforce disk space usage limits for jail-based virtual servers.

It certainly has it's drawbacks like fixed pre-allocation of disk space. It's also slower than real disk access.

On the other hand it places strict limits on the virtual server. If cannot exceed it's disk space allocation, takeing away possible race conditions from the host.

This approach will likely not be required anymore in the future as there is a project to create a storage virtualisation layer within geom. This will provide us most certainly with a better solution on a long term basis.