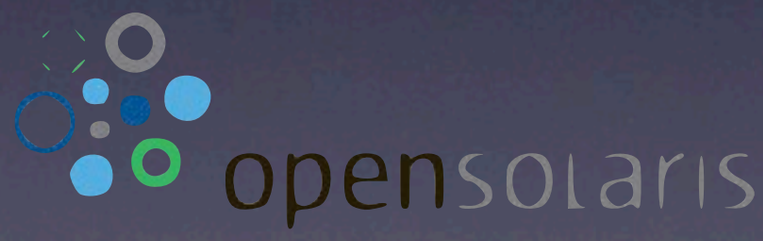
FreeBSD Jails
vs.
Solaris Zones

(and OpenSolaris)



James O'Gorman
james@netinertia.co.uk

Introduction

- FreeBSD user since 4.4-RELEASE
- Started using Solaris ~3.5 years ago
- Using jails for website hosting and package building
- Port maintainer

- I've been using FreeBSD since around 2001 - 4.4-RELEASE. Have also used Linux in that time, but found FreeBSD suited my needs better. Solaris x86 didn't run on any of my hardware at the time!
- I moved into full-time UNIX systems administration 2 years ago, which meant picking up Solaris, AIX and HP-UX. Prior to this I had been using Solaris on an ad-hoc basis.
- One of the things I do outside of work is to provide hosting for people or small projects. When data segregation or SSH access is needed, the site is segregated into its own jail.
- I also use jails for testing and packaging software.
- I'm also a port maintainer - Ports is the system for compiling third-party software. The package building cluster then makes binary packages out of these for the CD images. Users can install software from either ports or packages. Ports allows flexibility in that you can specify compile-time options if needed.

Why FreeBSD?

- Stable (look at Netcraft!)
- Clear divide between base and 3rd-party packages
- Active development, regular releases
- Early IPv6 adopter (KAME)
- Linux binary emulation
- Features being imported from OpenSolaris (ZFS, DTrace)

- So, why do I use FreeBSD?
- It's very stable. It was used for running sites such as Yahoo! and Hotmail. The Netcraft survey also shows FreeBSD as still being amongst the top webservers in the world.
- The FreeBSD community is very active, and releases are churned out regularly. The security team are also quick to respond to any issues.
- The KAME stack was imported into FreeBSD at 4.0-RELEASE
- There is a Linux binary compatibility layer, which I'll go into later.
- The project is always looking at what others doing and importing or adapting features. So far, ZFS has been imported and is now stable at ZFSv13. Version 15 was imported to HEAD last week, matching Solaris 10. DTrace is also in the kernel, although the port isn't yet complete as there are many Solaris-specific features in DTrace.

Why FreeBSD?

- “Ports” infrastructure for third-party software installation
- Jails

- The ports infrastructure is probably one of the key components to FreeBSD, and something users will use daily. The ports tree is a set of scripts and Makefiles for describing how to build software and package it. While users can use the binary packages provided by the project, many prefer to use ports to get the most up-to-date software versions and to get software patches quicker. Ports are not supported by the Security team, so port maintainers have to be vigilant and provide patches quickly. The package building cluster only has limited resources so a full set of binary packages takes a long time to produce.
- And lastly, of course, there are jails. Similar to Zones, jails are a very lightweight method for virtualising services.

Why Solaris?

- Stable
- Dynamic Reconfiguration (E10k/15k/25k/Mx000)
- ZFS
- Zones
- It's what I'm paid to do :-)

- Why do I use Solaris, then?
- Again, it's a very stable operating system. It's been used by companies large and small for a long time. It's also starting to get some really interesting features, such as:
- Dynamic reconfiguration, as seen on the bigger hardware
- ZFS. No explanation necessary.
- Of course, zones.
- And the most important point of all - I'm paid to do it!

Jails

- Came first! (4.0-RELEASE)
- Based on the idea of chroot(2), original development funded by external company
- Very lightweight
- Shared kernel
- All resources managed in-kernel

On to jails, then. What are they?
Well, they've existed since 4.0-RELEASE (circa 2000).
They were based on the idea of a chroot. The idea is, rather than providing separate chroots for, say, Apache, BIND, mail services, provide a separate environment to lock all of these services into, whilst keeping them away from the rest of the machine.
One of the FreeBSD developers was commissioned by a company to work on this for one of their products. Once complete, the work was handed back to the community and imported into FreeBSD.
Jails are very lightweight, requiring (as a base) only a few hundred megabytes of disk and around 50MB of memory. Of course this varies, depending on the services running in it.
Jails are just another instance of the operating system, so the kernel is shared, however the kernel manages all resources and knows the difference between a jailed process and a process in the host system.

Example jail setup

```
# cd /usr/src
# export DESTDIR=/data/jails/newjail
# make buildworld && make installworld
/etc/rc.conf:
      jail_enable=”YES”
      jail_list=”newjail”
      jail_newjail_rootdir=”/data/jails/newjail”
      jail_newjail_hostname=”newjail.testnet”
      jail_newjail_ip=”172.16.0.1”
# /etc/rc.d/jail start newjail
# jls
   JID  IP Address    Hostname           Path
     1  172.16.0.1    newjail.testnet    /data/jails/newjail
```

So a quick example of how to create a jail. Jail creation can be as quick as two minutes if a straight binary install is performed. Here, we’re building from source. All that needs to be done is the base system should be installed in a protected directory (i.e. no world access).
Next, the jail is configured in rc.conf. /etc/rc.conf is the FreeBSD way of enabling/disabling services, configuring networking, etc.
Once this is done, just call the jail RC script, and your jail(s) will start.
You can see which jails are running with the jls command.

Limitations of Jails

- Setup is a manual process
- Raw sockets disabled by default (but can be enabled)
- SysV IPC disabled by default (but can be enabled)
- No “console” device
- No resource control

So what are the downsides of using jails?
- Well, unlike zones, the setup is a bit more manual, however there are third-party utilities for managing jails effectively.
- Raw sockets are disabled by default as there is the potential for interacting with services outside of the jail. Some people like to enable this to allow ping and traceroute to work.
- SysV IPC is also disabled, as currently the namespace is shared in the kernel through all environments, so the jail might be able to interfere with processes outside the jail.
- There isn’t any sort of “console” device, however you can symlink /dev/console to a regular file to log what would otherwise go to the console. If direct login to the jail is required, this can be done by executing a shell inside the jail using the jexec utility (similar to zlogin).
- At the moment there isn’t any resource control, such as limiting CPU and memory usage for each jail.

Zones

- Resource management
- Shared /lib, /usr etc. OR whole root
- Managed by zoneadmd
- Automatic ZFS filesystem creation
- Branded zones (allows running Linux, Solaris 10 under OpenSolaris)
- Exclusive-IP allows zone administrator control of IP address
- Cloning

- Solaris allows the administrator of the global zone to apply resource limits (CPU, memory) to each zone. This prevents one customer or user from hogging the whole machine.
- By default, zones are created as sparse-root - /lib, /platform, /sbin and /usr are all inherited from the global zone. This means that they are read-only within the non-global zone.
- Each zone spawns a process called zoneadmd. This sets up the zone console, mounts any filesystems and configures network interfaces.
- When installing a zone, zoneadm will automatically create a ZFS filesystem if the parent directory of the zone is ZFS. This means that snapshotting zones is easy.
- Under Solaris and OpenSolaris, zones can be given an “lx” brand for creating Linux zones. This is very limited, however. Solaris 10 branded zones can also be run under OpenSolaris.
- If you have spare NICs, you can give the zone a whole NIC and its own network stack by using exclusive-IP. This then allows the zone administrator to make IP address alterations.
- ZFS-based zones can also be cloned using ZFS snapshots, making deployment much quicker.

Example zone setup

```
# zonecfg -z testzone1
testzone1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:testzone1> create
zonecfg:testzone1> set zonepath=/export/zones/testzone1
zonecfg:testzone1> add net
zonecfg:testzone1:net> set physical=e1000g0
zonecfg:testzone1:net> set address=172.26.0.2
zonecfg:testzone1:net> end
zonecfg:testzone1> verify
zonecfg:testzone1> commit
zonecfg:testzone1> exit
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - testzone1        configured /export/zones/testzone1        native   shared
# zoneadm -z testzone1 install
A ZFS file system has been created for this zone.
Preparing to install zone <testzone1>.
...
```

I'm sure you've all seen this before, but just for completeness, here is a quick run-down of creating a basic zone:
- Create the zone with zonecfg. By default 'create' will create a sparse-root zone. 'create -b' will create a whole-root zone.
- Setting the zonepath on a ZFS filesystem will, as previously mentioned, automatically create a ZFS filesystem for the zone.
- Networking can be added as required. The example here uses shared-IP networking.
- Verify that the configuration is valid - this will throw up errors if you missed something critical.
- Commit the changes
- zoneadm will show the zone as 'configured' - next we install it.
- A sysidcfg file can be placed inside the zone before booting to configure hostname, DNS or NIS, timezone and NFSv4 domain.

Comparison overview

	Jails	Zones
Resource Control	✗	✓
IPv6	✓	✓*
Managed by	kernel	zoneadmd
Runs on ZFS	✓	✓
Can be snapshotted/ cloned	✓	✓
NFS	Client-only	Client-only

* Requires manual setup of link-local address

- FreeBSD has no per-jail resource controls, however users can still be limited by the login capabilities database. Proper resource control is currently being developed.
- Zones can do IPv6, but it requires creating a false link-local address per-zone
- FreeBSD sets up the environment (mount points, IP addresses) using RC scripts (or third-party utilities). Once the jail has started all resources are handled by the kernel.
- Both will happily run on ZFS.
- Both can be snapshotted and cloned using ZFS for rapid deployment.
- NFS cannot run as a server in jails or zones as rpcbind will only listen on ALL interfaces.

Linux

- Why am I talking about Linux to Solaris people in a BSD presentation?
- Some vendors only provide Linux binaries
- Both FreeBSD and Solaris can run Linux binaries - in limited ways.

- Please don't lynch me!
- The FreeBSD documentation provides examples for installing software such as Wolfram Mathematica, Oracle and SAP. While this generally isn't supported by the software vendor, being able to run these applications on non-native systems can be useful.

Linux

- FreeBSD has Linux binary compatibility. Emulation layer pretends to be Fedora 10
- Debian/kFreeBSD can also be installed in a jail!
- BrandZ allows installation of RHEL3 into a zone

- Most Linux binaries can run on FreeBSD with no problem. This works by looking at the ELF header. If a binary isn't explicitly marked as being a Linux binary, the brandelf utility can do so. FreeBSD then knows to use the compat_linux layer.
- The Debian project don't just use the Linux kernel. They also have ports to the HURD kernel and the FreeBSD kernel. The latter project is designed to run a GNU userland with the BSD kernel (similar to Nexenta and OpenSolaris).
- BrandZ is still quite limited. RHEL3 is very old (2006) and no longer supported.

Bibliography

- http://docs.freebsd.org/44doc/papers/jail/jail.html
- http://arc.opensolaris.org/caselog/PSARC/2002/174/zones-design.spec.opensolaris.pdf
- http://uptime.netcraft.com/perf/reports/Hosters