FreeBSD 8 to 10

One ISP's journey forward and backward in time

Nick Wolff
Oarnet

BSDCan 2016 - June 11/12 2016 - Ottawa, Ontario, Canada

Who am I?

- Freebsd user since 2012
- 2nd BSDcan
- Linux user 2005 - 2012
- Lead OARnet Network Management Infrastructure
    - bastion hosts, authentication, logging, stats, visualization, network monitoring, development platforms, vpn, ntp, dns
- Python and shell experience
- Compile, troubleshoot, and patch more C then I am usually willing to admit
- Backfill Routing Engineer workload as needed

Who is OARnet?

- Part of The Ohio State University and state government
- Supply internet/network services to Higher-ed, K-12, state and local gov, health care, research and public broadcasting throughout state of ohio
- 4,000 km of fiber
- 100gbps backbone
- Service provider since 1987
- Juniper based routing and switching
  - ≈ 350 devices end of 2015, ≈ 1000 devices today , ≈ 2000 planned by end of 2017
- Freebsd based network management infrastructure since freebsd 4
  - 100+ devices in various states of production

Exterior design influences

- Service uptime/stability is top
- Services support and bootstrap a state’s infrastructure
- ≈ 3 hour drive max to any point from columbus
- Servers maybe anywhere in state.
- 9600 baud Serial OOB connection over OOB L2 network
- Remotely Switchable Power(hopefully)
- High bandwidth, very configurable, low latency network

In the Beginning

- Infrastructure gradually built since FreeBSD 2.2
- Majority of systems 8 Stable, some 6, one freebsd 4 system left.
- All 32 bit
- UFS
- Custom installer called cf-install
- Custom backup solution called nb-dump
- Manual source builds for userland software and rsync of /usr/local
- Lots of little tweaks to sysctl and endless configurations files
- Kernel configurations modifications needed

CF-install

- Sysinstall replacement combined with deployment system
- Designed to do full reinstall with zero physical intervention
- Deployment system based on mfsroot and compact flash cards
- Lots of incremental features and changes to fix problems

nb-dump

- UFS dump based
- Locally encrypts and compresses dump
- Sends over ssh to a server who manages backups
- Splits to multiple raid array every other day
- Does cycles of full and incremental updates

Goals

A. Integration of modern changes and updates with legacy modifications
B. Minimize custom work
C. Maximize manageability and maintainability
D. Raising the already high standards for uptime and reliability of services

Decisions, Decisions, Decisions

- inspired designs
- obsoleted technical issues
- personal preferences
- coin flips

Your problems are not just your own when it comes to system administration

Modern changes integrated

- Completed
  - ZFS
  - 64 bit
  - GPT partitioning
  - Modern usb Memstick install
  - Straightened out sysctl and other other options previously carried forward
    - Over cautious to avoid change
  - No source or kernel config modifications
- Future
  - EFI boot
  - Automated provisioning

Server-Install Overview

Internal toolset to personalize memstick image for remote server deploys

- Mainly a template system to ensure identical and simple system builds
- Shell based
- Single command gives us a memstick that will completely bring a machine up
  - Secure
  - Base services
  - Misc Configuration
  - Standard files
  - I don’t ever have to physically touch servers

Server-Install - localize-image.sh

```
usage:  -n hostname [-o output ] [-c configuration] [-f filename] [-i ip-address] [-m netmask] [-d default-route] -p
where:
        filename: The file name of the input image to be used
        configuration: The configuration directory to use
        output: The file name of output image
        hostname: Fully qualified hostname for new system
        ip-address:ip address for new system
        netmask: netmask for new system in dotted-decimal
        default-route: default route for new system
        -p makes it a S1 public info box with a more minimal config stripped of certain data like standard users and paswords
note:
        If supplying networking information ip address, netmask, and default route must all be specified
```

Server-Install - localize-image.sh

- Injects files and scripts into standard* memstick
- Install flag that get’s wiped after first install
- Uses mtree to deal with permissions
- Inject /usr/local software tarball installed as part of dist
- Use boot0 as bootloader on memstick
  - After install boot0 points to first harddrive instead (gpt/zfs disks)
  - Usb is boot disk in bios

- Future expansions
  - Wrap Nanobsd builds
  - Pull info from central database

Random Things we do

- Set home directory to /home/${hostname}/a
- Set Timezone,dns,ntp servers, sysctl settings, more
- Setup serial access to memstick and system(only need if different baud)

BSD-Install vs pc-sysinstall

PC-Sysinstall

- From PC-BSD
- Well Structured
- Good config format
- Easily injectible into builds
- Easy to modify without rewriting
- Good hooks at multiple places in the config

PC-Sysinstall - Config General

hostname=clmbs-proto10.eng.oar.net
installInteractive=no
installMode=fresh
installType=FreeBSD
installQuiet=yes
packageType=dist
installMedium=local
localPath=/usr/freebsd-dist
distFiles=base kernel lib32 src games usr-local
zpoolName=zroot

PC-Sysinstall - Config Disk

```
# Disk Setup for da0
disk0=da0
partition=ALL
bootManager=bsd
partscheme=GPT
commitDiskPart

# All sizes are expressed in MB
# Avail FS Types, UFS, UFS+S, UFS+J, ZFS, SWAP
# UFS.eli, UFS+S.eli, UFS+J.eli, ZFS.eli, SWAP.eli
disk0-part=ZFS 0 /,/usr,/var,/home (mirror: da1 da2)
disk0-part=SWAP 32000 none
commitDiskLabel
```

PC-Sysinstall - Config Network and hooks

```
#Network config for server
netSaveDev=igb0
netSaveIP_igb0=192.168.141.73
netSaveMask_igb0=255.255.255.0
netSaveDefaultRouter=192.168.141.1
netSaveNameServer=10.244.194.2
netSaveNameServer=10.244.195.2
netSaveNameServer=10.244.193.2
netSaveNameServer=10.244.192.2

#move home directory to correct mountpoint
runExtCommand="/server-build/bin/prep-chroot.sh"
runCommand="/server-build/bin/chroot-finalize.sh"
runExtCommand="/server-build/bin/clean-chroot.sh"
#finish install
runExtCommand="/server-build/bin/finalize.sh
```

Life-Preserver

- From PC-BSD
- Iscsi based
- Zfs Send/Recv to a geli disk served over Iscsi backed by zvols within a stunnel
  - Lots of advantages
  - Don’t need to trust backup server
  - Push instead of pull
- Easy to expand storage by just adding a new zpool

Life-Preserver patches (forthcoming)

Done Internally

- **Alternating Patch**
  - Alternating days
  - Pick a random time to start a cron job (00:00-4:00 every other day)
- **Metadata Backup**
  - An implementation currently exists upstream but currently require user intervention
  - Need to be able to decrypt backups
  - Public key encryption using tools in base

Still needed

- **Automated provisionings**
- **Misc Features**
  - Logging via syslog and generating/sending reports"
  - Servers being able to set themselves up (verifying IP addresses and reverse DNS and hostnames etc)

Lessons learned

- Many people are having the same problems
- Talk to the community
- Complain and listen to other people's complaints
- Praise the work that makes your life easier
- 10 hours of planned work now can save me 5 hours of unplanned work at 3am some random night
- The freebsd build system is wonderful

Thank You

Questions?

nwolff@oar.net
@darkfiberiru
darkfiberiru@gmail.com