Building an Integrated Server Product using FreeBSD

Mark Taylor
Cybernet Systems Corporation

Available online at:
http://www.freebsd.org/~mtaylor/freebsdcon.html

Abstract

Presented will be

- Goals of the NetMAX
- Included Softwares
- Introduction to Feature Creep
- Development of an Integrated Server
- Data Representation
- The Commit Process
- Porting to Linux
- Protecting Our Investment
- Future Products
- Summary

Free NetMAXes, based on FreeBSD 2.2.7, will be available after the presentation.

Goals of the NetMAX System

"Features" of Software on Typical Server Systems

- multiple vendors
- disintegrated
- varying configuration methods
- server process signaling needed (via "kill")

NetMAX Server

- comprehensive server suite
- cross-subsystem dependency knowledge
- single, integrated, web-based, configuration interface
- automatic server process signaling upon reconfiguration

Included Softwares (Short List)

- DNS Server
  - bind
- Email Services
  - sendmail, procmail, uw-imap, uw-pop
- Web Services
  - Apache, MS-FrontPage
- File Sharing
  - NFS, SAMBA, NetAtalk+asun
- FTP
  - wu-ftp
- Tape Backup
  - tar, smbclient, ftp
- Routing, Firewalling, NAT
  - OS-provided
- DHCP Server
  - ISC-dhcp

Introduction to Feature Creep

Driven by Crazed Marketing Department

- Easy to say "Add it!"

Driven by Crazed Programmers with All The Time In The World (tm)

- "I'm going to prove that I can do it!"

Sensibility and time constraints must limit effort if product is to be completed.

Development Process: Why?

- Problems with Windows/NT Server
  - Losing files
  - Slow performance
- Much in-house Unix knowledge
  - FreeBSD, IRIX, A/UX
- Publicly available file sharing packages
  - SAMBA, NetAtalk

Prototype NetMAX System

- FreeBSD 2.2-SNAP
- 13 drives
- Three SCSI adapters
- Ran for years, no OS-upgrade, until converted to a Linux/NetMAX

Development Process: In the Beginning (1)

Single Development Machine

- Installed Software
  - FreeBSD 2.2, Apache, SAMBA, NetAtalk
- Development Tools
  - vi, Netscape Navigator 3

Drawbacks

- Multiple CGI failures per hour, due to having two people editing library files
- Lost code due to file overwrites
- No code history

Development Process: In the Beginning (2)

Solution

- Separate development machines
- Single NFS-based server for installation of code base to development machines
- Edit files on development machines, copy completed files back to server

Drawbacks

- Lost code due to file overwrites
- No code history
- Need more machines

Eventually, RCS was added to the server's code tree to address the lack of code history.

Development Process: In the Beginning (3)

Solution

- Added web servers to "personal" IP ports on code machine

Drawbacks

- Code machine gets bogged down with web server processes
- No control over system programs

Development Process: Introducing CVS

Benefits

- Single location for code source, in RCS format
- Easy to merge in code changes
- Code history
- Remote access for check-out and check-in
- Branching

Having source code, as opposed to compiled binaries , makes many programming tasks easier, but adds the time to build the source code into binaries.

Development Process: Developing a System Installer

To reduce bog-down of "personal" development environments on code machine, a System Installer was developed.

Initial Installer:

- No questions asked
- Installs onto first detected hard-drive
- "finger"-based
- Single boot floppy, with extraction tools

Drawbacks

- More development machines needed
- No control over destination
- Eventual source was to be a CD-ROM
- Need a System Builder

Development Process: The Automated System Builder

Separate machine setup to do daily system building, similar to make release.

Daily Tasks of Automated System Builder

- CVS export of OS source code
- buildworld + installworld
- CVS export of subsystem source code
- build + install of subsystems
- CVS export of NetMAX specific source code
- build + install of NetMAX source code
- build system image "tar" file
- build system CD-ROM image

Development Process: Web Server and mod_perl

The entire System Configuration interface is web-based, and all of the HTML is generated from Perl CGI scripts. mod_perl speeds up the access to these scripts enormously.

Benefits

- Speed

Drawbacks

- High memory overhead for per-child caches
- Forces better Perl programming practices

Solution

- Process watcher which HUPs inactive web servers
- Eventually, Apache will be threaded under Unixes

Data Representation (1)

Initially we looked at SAMBA's "smb.conf" configuration file. Should we attempt to parse the data every time we need it, or use a private (separate) configuration database?

Benefits of Parsing

- Single data source
- Admins can "vi" the data and have it reflected in the interface

Drawbacks

- Many complex config-file parsers to write
- Cannot proceed if errors exist in configuration files
- Inter-dependency checking more complex

Data Representation (2)

Solution

- Use a private database

Reasoning

- Interface can prevent "bad states" and "bad data" from being entered
- Reducing complexity reduces time
- Speed

Summary of Database-Related Tasks

- Determine layout of database
- Implement a global, file, and record locking mechanism
- Write converters to change our format to subsystem config files
- Determine database-to-subsystem dependencies

The Commit Process

When data is entered in the interface, it is not reflected in the subsystem configurations until a Commit is performed.

Example without batched changes

- 1) Edit a web server's configuration
- 2) Wait for changes to propagate
- 3) Edit a file share's configuration
- 4) Wait for changes to propagate
- 5) Add an IP address to an interface
- 6) Wait while changes propogate to virtually every subsystem

With batched commit

- 1) Edit a web server's configuration
- 2) Edit a file share's configuration
- 3) Add an IP address to an interface
- 4) Commit the changes
- 5) Wait while changes propogate to virtually every subsystem

Porting to Linux

NetMAX "sits on top of" software subsystems, providing a common configuration interface. Why not port it to another popular Unix-variant OS?

Goals

- Do not re-invent the wheel!
- Minimize divergence between FreeBSD and Linux versions of CGI scripts
- Minimize changes in subsystems

Porting to Linux: CVS Issues

Our NetMAX and subsystems CVS Repository

- Started using "cvslines" from Network Appliances, to make parallel code-line development trivial

Our Linux CVS Repository

- Internet: no central repository for Linux + GNU code
- Imported binary RPMs and source code SRPMS
- Searching for code in SRPMs is difficult
- Making changes requires rebuilding both the RPM and SRPM
- Using binary RPMs, System Build time decreased

Porting to Linux: The Universal Translator (1)

NetMAX code base already had a tool to securely request program execution as another user. This tool was extended to perform the following.

- Request program execution as "self" (real-uid, real-gid)
- Request command-line translation, based on table lookup of OS version
- Parameter re-ordering is possible when translating

All NetMAX scripts' system() calls were converted to run through this tool.

Porting to Linux: The Universal Translator (2)

Example of UT in action, on a Linux 2.0.37 based OS:

system() requests UT to execute "grep"

- UT sees the key "grep"
  - performs a table lookup based on "Linux-2.0.37"
    - no value
  - performs a table lookup based on the more generic "Linux"
    - value = "/bin/grep"
- UT executes "/bin/grep"

Porting to Linux: The Universal Translator (3)

UT Benefits

- Flexible
- Automatic translation
- Each OS version and sub-version may use different values
- Parameter re-ordering, and deletion, is possible

Some program translations cannot be handled by the UT, because the input or output formats vary too differently. In such cases, the UT translates these calls to custom "compat" scripts.

Protecting Our Investment

NetMAX (our CGI scripts and libraries) is currently not available as Open Source software.

However

- Cybernet is actively looking at various licensing agreements which will allow us to keep our competitive advantage
- Cybernet is looking into providing portions of the NetMAX system under a BSD-style and/or GNU-style license agreement

Protecting our Investment: License Manager

Good properties for a License number

- Multi-machine licenses
- Cannot be duplicated on LAN (except for multi-machine #s)
- Specific to sub-product (Web Server, Firewall, File Server, etc.)
- Can have multiple license numbers per machine
- "demo" #s which have embedded expire date

Invalid license numbers disables major services, and will cause the Serial Number management page to come up when the Administrative Web Interface is accessed.

Protecting our Investment: Perl Source-Code Protection

Perl is compiled (tokenized) every time it is executed. Since the original text version of the source code is required, it is difficult to protect the code.

We modified a module from CPAN called "Filter" to help "hide" the source code, coupled with either a kernel modification, or user-level daemon, that checked several things about the decrypting program (perl and httpd) before allowing decryption.

Protecting our Investment: Drawbacks to Protection

- If kernel-based, limits users to one kernel, or to one provided by software company
- Cannot provide ".o", ".a", ".so", lkm, or kld, of protection client or server code
- Apache and Perl "bloat" because of static linking
- Users cannot correct bugs, review the code, or "see what’s going on inside"

Future Efforts

- Development of OEM versions for embedded systems
- Release of parts of source code
- Porting to other platforms and processors: NetBSD, OpenBSD, Windows?
- Larger-scale system support: Kerberos, CODA, Andrew File System, clustering
- More ISP-desirable features
- Client version: NetMAX as your Desktop
- Keeping up with OSs and subsystems
- More knobs and buttons in the interface, without complicating interface

Summary (1)

Code Management

- use CVS and cvslines
- the repo is for the source, not tarballs of the source

Project Management

- only one manager, with the ability to say "no"
- early decisions have huge impacts

Managing Feature Creep

- listen carefully, but list it under "future releases"

Source Code Protection and Licensing

- for protection, use a user-level daemon, as opposed to a kernel-level hack
- license numbers can encode data like product code and expiration date

Summary (2)

System Installation

- create bootable CD-ROMs and floppies, plus quick installation of daily builds for developers

Database Management

- pseudo-record locking can be acceptable

Implementing Subsystem Changes

- batch the changes for speed

Writing Portable Software

- brucify your code! (generate a code style, and strictly enforce it)
- peer review is invaluable!

The original NetMAX product, which is FreeBSD 2.2.7-based, is now available for free, here at the front, and in our vendor booth.

Presentation by
Mark Taylor
mtaylor@freebsd.org

http://www.netmax.com/