FreeBSD update Tuning

freebsd-update - the pains and some help

Intro

FreeBSD comes with the freebsd-update script as "the way to go" from FreeBSD 10 and up.

Where the script shines is the automatic patching of the system. Where it does not shine at all is handling your /etc.

Where there's been solutions like etcmerge for a decade, freebsd-update still uses mergemaster with incomplete flags and on any larger update you're forced to you through dozens of /etc files, often to do no more than replacing "9.1" with "10.1" in the merge. It's stupid, unbearable and should have been killed 10 years ago. Since most people don't run larger numbers of servers, there's still a lot of reluctance to proper solving this.

what you can do!

- Start a fundraiser to integrate **etcmerge** within **freebsd-update**.

  (yeah, that's a tricky one)

- See the following threads:

  https://lists.freebsd.org/pipermail/freebsd-questions/2013-June/251823.html

  This describes how to create an /etc/mergemaster.rc with more sensible settings for mergemaster. I think there's still a setting missing (-u or something?) but it's supposed to make mergemaster smarter which will solve many problems.

  http://markmail.org/message/srjc7hdj3coxqs5d

  This describes an alternate workflow, mix of manual and not, but generally more reliable than what freebsd-update does. I don't recommend this one since it's very hands-on and not really moving forward, but the general mood of the thread is a realistic view of the subject.

  http://freebsd.1045724.n5.nabble.com/I-donot-like-using-mergemaster-td3805954.html

  This has an even extended mergemaster.rc.

I'm testing the adjusted mergemaster.rc from thread #1 and will update here once I see if it made any positive difference.

For now, read on about all the other options you have.

Further help

I've made a finely tuned freebsd-update.conf you can grab here in my FreeBSD Repo:

https://bitbucket.org/darkfader/freebsd/src

mergemaster

/etc/mergemaster.rc

```
AUTO_INSTALL='yes'
AUTO_UPGRADE='yes'
# keep our custom motd
IGNORE_FILES='/etc/motd'
# Do not display changes that only affect whitespace
DIFF_FLAG='-Bub'
FREEBSD_ID='yes'
DELETE_STALE_RC_FILES='yes'
COMP_CONFS=yes
PRESERVE_FILES=yes
PRESERVE_FILES_DIR=/var/mergemaster/preserved-files-`date +%y%m%d-%H%M%S`
IGNORE_FILES="/etc/crontab /etc/fstab /etc/group /etc/hosts /etc/inetd.conf
/etc/make.conf /etc/master.passwd /etc/motd /etc/newsyslog.conf
/etc/freebsd-update.conf
/etc/ntp.conf /etc/ntp.drift /etc/profile /etc/rc.conf /etc/resolv.conf
/etc/shells /etc/syslog.conf /etc/ssh/sshd_config
/etc/ssh/ssh_host_key /etc/ssh/ssh_host_key.pub /etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub /etc/passwd /etc/rc.conf.local
/etc/zfs/exports /etc//namedb/named.conf /etc/periodic.conf
/etc/hosts.allow
/etc/hosts /etc/pf.conf /etc/sysctl.conf /etc/make.conf /etc/src.conf
/etc/mail/aliases /etc/mail/mailer.conf /etc/remote
/etc/mail/freebsd.mc/etc/mail/
freebsd.cf /etc/mail/freebsd.submit.mc /etc/mail/freebsd.submit.cf"
```

Faster Updates

Users *outside the US* will also know the largest problem: freebsd-update is incredibly slow. Updating a server from, say, 8.x to 9.3 could take over 10 hours of freebsd-update time.

A lot of research has brought up the following points:

- There was a strong anti-mirroring policy when the original developer still was also maintaining that infrastructure. This hat has been passed on
- The script & docs to build your own build server exist, but you'd better plan a few man-days to do that, and if you *read* the documentation you'll probably also end up wondering if you want that. Besides you'd be running your own FreeBSD build forever. A splendid way to waste your bosses money on nothing...
- Right now the number of available mirrors is still incredibly low, at best 1% of the normal ftp/www mirrors.
- The lack of local (your country or rather your subnet) mirrors increases the latency, which slows down every single patch download
- freebsd-update is said to use some kind of streamed http transfer which plays badly with squid proxies!
- freebsd-update knows when to use a full file, and when to fetch patches (This is *great*, more in a moment...)

About the Proxy slowdown

For large infras overall it is still a lot better with a proxy. I usually patch one client in advance to pre-fetch everything.

Nonetheless: The per-client time looks bad. If you also have slow storage, it is NOT possible to quickly update a server due to freebsd-update's performance issues. A binary update via the dists as described in the second mailing list post above will be dozens of times faster.

Preseeding freebsd-update

What you can do to speed up your *major* updates is to use a script that uses the freebsd ISO to pre-seed the update directory. If freebsd-update finds all the right files in it's cache dir, it will happily use them, which can save you many hours. Extracting the files from the loopback mounted iso takes like a minute with this script.

fix-freebsd-update.sh

Found at: https://gist.github.com/thefloweringash/8729473

After the update

Rollback

rollback is possible but only for *one* update step.

If you need to restore some single item or one from an older state, you can also fetch & buildworld for the right version. Then just look i.e. in /usr/src/etc, /usr/src/etc.amd64 or /usr/obj (find from /usr/src and /usr/obj is perfect)

Clean up the cache

(if you found you don't need to roll back)

(command will be added)

Remove stale files

```
root@freebsd1:/usr/src # make check-old
>>> Checking for old files
/usr/share/man/man9/sleepq_calc_signal_retval.9.gz
/usr/share/man/man9/sleepq_catch_signals.9.gz
>>> Checking for old libraries
>>> Checking for old directories
/var/named/etc
/var/named/etc/namedb
To remove old files and directories run 'make delete-old'.
To remove old libraries run 'make delete-old-libs'.
```

This is not just related to freebsd-update, more a general housekeeping task.

If you don't do it regularly, you run risk of re-using an old library way past it's time and not finding out until it's damn too late.

check freebsd-update IDS

Ideally, with the ignorelist from above freebsd-update.conf, freebsd-update IDS should only show very few or NO modified files.

You could run it as a daily or weekly check to make sure you detect unwanted modifications within short notice.

As you can see on this system you get only a few alerts (which should also be investigated and added to the config or fixed)

```
otherhost # freebsd-update IDS
Looking up update.FreeBSD.org mirrors... 5 mirrors found.
Fetching metadata signature for 9.1-RELEASE from update2.freebsd.org...
done.
Fetching metadata index... done.
Fetching 2 metadata patches.. done.
Applying metadata patches... done.
Inspecting system... done.
/.cshrc has SHA256 hash ...
/etc/hosts.allow has ...
/etc/login.conf.db has ...
/etc/printcap is a symlink, but should be a regular file.
/root/.cshrc has ...
/usr/share/openssl/man/whatis has 0664 permissions, but should have 0644
permissions.
/var/tmp/vi.recover is owned by user id 1041, but should be owned by user
id 0.
```

I feel this is nicely scriptable and far less erratic than the freebsd periodic mails on their own.

Keep in mind that you can also run this from a live ISO using the -b option, this should allow some extra security against tampering.

find missing libs

The port sysutils/bsdadminscripts contains the one perfect little helper...

/usr/local/sbin/pkg_libchk - I really love this!

```
root@freebsd1:/root # /usr/local/sbin/pkg_libchk
compat8x-amd64-8.4.804000.201310_2: /usr/local/lib/compat/libopie.so.6
misses libmd.so.5
compat8x-amd64-8.4.804000.201310_2: /usr/local/lib/compat/libtacplus.so.4
misses libmd.so.5
```

In my case, this would mean I should probably reinstall the compat package, and also wonder if I care about this old lib.

find unused libs

I don't really like or care for this...

portmaster sysutils/libchk or pkg install libchk

> **ruby warning**
> libchk depends on ruby20, and even depended on ruby19 a moment back. lotsa mess.

It correctly detects this havoc:

```
Will look into:
 /bin
 /lib
 /sbin
[...]
 /usr/local/sbin
 /usr/sbin
Unresolvable link(s) found in: /usr/local/lib/compat/pkg/libaprutil-1.so.4
 libapr-1.so.4
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/fast-stemmer-1.0.2/ext/stemmer.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/fast-stemmer-1.0.2/lib/stemmer.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/ffi-1.9.3/ext/ffi_c/ffi_c.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/ffi-1.9.3/lib/ffi_c.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/posix-spawn-0.3.9/ext/posix_spawn_ext.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/posix-spawn-0.3.9/lib/posix_spawn_ext.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/redcarpet-3.1.2/ext/redcarpet/redcarpet.
so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/redcarpet-3.1.2/lib/redcarpet.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/yajl-ruby-1.1.0/ext/yajl/yajl.so
 libruby19.so.19
Unresolvable link(s) found in:
/usr/local/lib/ruby/gems/1.9/gems/yajl-ruby-1.1.0/lib/yajl/yajl.so
 libruby19.so.19
Unreadable file or directory: /usr/sbin/pkg_list
Unreferenced library: /lib/libssp.so.0
Unreferenced library: /usr/lib/libBlocksRuntime.so.0
Unreferenced library: /usr/lib/libform.so.5
[...]
```

And now one more thing...

Portmaster

In this case we're looking at a desktop system.

All packages are set to be built locally, I dont want the official packages i.e. for X.org or other things where Mr. Poetterings broken HAL system might be still active.

Also pkg lock isn't reliable (always tries to deinstall my updated linux-base even though it's locked and such stuff). For that reason I'm working with portmaster most of the time on my desktop.

I've found these to be the ideal flags for my usage.

There is still some room for improvement, described later.

Calling portmaster for updates

```
portmaster -abwgHy
            |||||\_ answer yes for common questions, including
install/removal
            ||||\_ hide the build info in log
            |||\_ create a package for the newly built port
            ||\_ save_shared WOPT
            |\_ build a backup package for installed port
            \_ work on all ports
```

SAVE_SHARED=wopt explanation missing

Possible improvements

Set it to a local package dir using --local-packagedir

Specify to --packages-local use packages first if they exist *locally*, so it'll reuse the ones it already has.

Use --update-if-newer instead of -f, but whenever I specified --update-if-newer it throws an error.

I also test using --check-depends, it works well but seems to be more useful for monitoring.

More about tuning using a portmaster.rc is found here:

http://aboredcoder.com/post/how_to_setup_and_configure_portmaster_on_freebsd

That sounds like the way to go to configure automatic deletion of old package / dist files.

Running pkg safely

Secure the pkg update

Running the pkg update is most safely done separately, independent of your normal updates / installs. By this you avoid pkg suddenly breaking while it is in the middle of doing changes. We first store the database, then update the package lists and then use the *static* version of pkg to update pkg.

```
cp -p /usr/local/sbin/pkg-static /var/tmp/pkg-static.$$
pkg backup -d /var/tmp/pkg-last-backup.db
gzip -1 /var/tmp/pkg-last-backup.db
pkg update
pkg-static upgrade pkg
```

Using the static version matters if one of the libraries pkg is using has changed during an OS update (ABI issues) or because pkg decided to update it's own helpers. Yes, all of this can happen. It does not reaaaally lock / protect it's own package or dependencies.

You can use pkg update -f if it should not correctly fetch very fresh package lists i.e. through a proxy. I don't know the specific reasons behind this, but it has helped.

No partial upgrades

They're not supported. This is not a joke.

- package you want to install has no dependencies and doesn't pull any dependent package upgrades: *Ok, go ahead.*
- package you want to install has *new* (not yet installed) dependencies and doesn't pull any dependent package upgrades: *Ok, go ahead.*
- package you want to install has dependencies and would pull any dependent package upgrades: **Don't go ahead.**
- package you want to upgrade has no dependencies and doesn't pull any dependent package upgrades: *Ok, go ahead.*

The following **should** work if your action fell into in the 3rd case

1. upgrade everything, based on your current packages.
2. add the new package.

> **Summary**
> If pkg suddenly tells you about packages you didn't mention on the command line and they're not new dependencies:
>
> Abort, you'll need to go with a full upgrade.

Option changes

See separate article FreeBSD pkg options changed / reinstall