# To cache or not to cache
# making pkg_add faster

Marc Espie <espie@openbsd.org>, <espie@lse.epita.fr>

June 3, 2022

## Teaser

This is just a "lightning talk" to give you some of the recent elements. There is more to the story, which will get expanded upon in this year's summer week, hopefully

## The story so far

`pkg_add` is a tool specific to OpenBSD. Historically, we do "just in time" updates.

- open new package and peek at meta information
- decide whether we want to update
- if so, extract the new package, then delete the old one

## The meta information

We got structured information (packing-lists) that looks like this:

```
@pkgpath x11/dbus
@newgroup _dbus:572
@newuser _dbus:572:_dbus::dbus user:/nonexistent:/sbin/nologin
@extra ${SYSCONFDIR}/machine-id
@rcscript ${RCDIR}/messagebus
@bin bin/dbus-cleanup-sockets
@bin bin/dbus-daemon
@bin bin/dbus-launch
@bin bin/dbus-monitor
@bin bin/dbus-run-session
@bin bin/dbus-send
@bin bin/dbus-test-tool
[... more files]
```

*This is the source information for the* `dbus` *package, telling us it requires some user/groups, has a service start-up script, and contains a bunch of files*

## The whole story I

After going through `pkg_create`, the full *packing-list* looks more like

```
@name dbus-1.14.0v0
@version 8
@comment pkgpath=x11/dbus,-main ftp=yes
@arch amd64
+DESC
@sha TYbBC2oO7XXOXqnQOFU6qikEuiN+fqoN2azXrJA9jJg=
@size 448
@pkgpath x11/dbus
@wantlib X11.18.0
@wantlib c.96.1
@wantlib execinfo.3.0
@wantlib expat.14.0
@wantlib pthread.26.1
@wantlib xcb.4.1
@newgroup _dbus:572
@newuser _dbus:572:_dbus::dbus user:/nonexistent:/sbin/nologin
@cwd /usr/local
```

## The whole story II

```
@extra /etc/machine-id
@rcscript /etc/rc.d/messagebus
@sha G8InGFO+lEOiPUMpXqicxPO1KEkofHOguRhxV9sMXHk=
@size 172
@ts 1653570364
@bin bin/dbus-cleanup-sockets
@sha lew9jO3YckJ1VnMPtypbKh1k1eedAXgwCvYU3hE44jU=
@size 13318
@ts 1653570364
[... more files]
```

## Structured information

- a *packing-list* is a structured object which has constructors. Most often it starts as
  `my $plist = OpenBSD::PackingList->from_file("filename");`
- objects (*packing elements*) can be added to it using the right method:
  `OpenBSD::PackingElement::Wantlib->add($plist, $w);`
- some complex objects can have a multiline representation, like files:
  - name
  - extra modes
  - checksum
  - timestamp
  - ownership

## OO properties

- there's a whole hierarchy of objects: anything file-system related is a `FileObject`, annotations are `Meta`, anything depend-related is a `Depend`
- objects are emitted in a specific order: first all the meta information, then the actual objects (in order)
- most operations happen as visitors on the packing-list
- there are specialized scanners that take advantage of the text structure of the packing-list to avoid reading it all

```
sub DependOnly
{
		my ($fh, $cont) = @_;
		while (<$fh>) {
				if (m/^\@(?:libset|depend|wantlib|define-tag)\b/o) {
						&$cont($_);
				# XXX optimization
				} elsif (m/^\@(?:newgroup|newuser|cwd)\b/o) {
						last;
				}
		}
}
```

## Speed

In order to decide whether to update a package

- we look up all packages that have the same name with a different version number
- we open every package to see whether it's a valid candidate
- we filter the ones we don't want
- pathological case: autoconf. We have a branch for each version, which means 17 packages to consider.
- ... and we decide to update

## Slower and slower

- for a long time, the network was slow, bandwidth-wise, so opening lots of files was not a big issue
- actually properly closing was an issue with ftp: premature closing requires full telnet support (with "attention" commands")
- and so I had to fix ftp-proxy back in the day
- ... but recently, latency is more of an issue, most people have lots of bandwidth, and so does our current setup

## CDN for the masses

We got a CDN

- ftp is dead, long live http(s).
- establishing connections might be a bit slow
- the cdn first gives you a redirect which means two connections
- we've parsed the redirect from the start, to make sure an update connects to exactly one mirror
- bandwidth is not an issue, latency is

## https ? not such a good idea

- http connection establishment is 1 $\frac{1}{2}$ RTT
- https is 2 $\frac{1}{2}$ RTT at best !
- we did implement session resumption (with fun results)
- ... so updates is slow, because we establish lots of connections
- also, signatures

## Signatures ?

```
untrusted comment: verify with openbsd-69-pkg.pub
RWSG2ib5ZXSfQTrcxxj+A9b6oeFI/OiJVB49nvIs+UPIull+Mk/BclTXRuG4a+XbnyoiZffDILfP58BNelKOyMjZNE
date=2021-02-26T23:06:32Z
key=/etc/signify/openbsd-69-pkg.sec
algorithm=SHA512/256
blocksize=65536

9d61ddfc76218e7c3745bd942a29725ff1bc651f64af27a450da33a73f292d69
8621c7932e29c838783177287fc5779186c854b35eaa541e787979f78288c2a6
d895cc173cb9058341bbcbe6abe3c018b915eb9218fd65c31f490f9af9c11041
9895735d7a109e497ef3f616f35938ae4d6e66f851f038ba50aa2a69808ef53a
ce2331349065 6aaeda9b21aa137a7e70fb268db9372cafeefe860e3fb98c4dfb
d34eedc74d714c7a5702b386d36ee422d614d0239cf45e3ae417dd5cd6a09f6f
55330726f9221f239c76d4809463ebc251a634360f7098cff98931f8948b7669
e84f66e180f1be0c5ef057ea2c4bc74106791b6b794e2de74dc56a9968fa8410
e2e4283c81ace8474a32dfc6e43fa3515f02e9bdc93daa86d84875cf9d4ac72d
aa1588b1ca21bf13dc132fd12e485cf0edebc787ee53a4cf6df6aa8d5e5e5611
9f6723f0419bc16b0a1230407ab3e25015dda27793c424bc50a6ace4f7de4a2e
```

## chicken and egg

We'd like to store the update info somewhere but

- we don't have any db tools in the base system
- we need to generate and grab it securely from the cdn

## no database

- But we have `locate` in the base system.
- it's been designed to store efficiently "similar" strings (by sorting and compressing according to prefix)
- already used for `pkglocatedb`
- this stores each path in packages prefixed by the pkgname/path location

## pkglocatedb

```
nausicaa$ pkglocate /usr/local/bin/vim
graphviz-2.42.3p0:math/graphviz,-main:/usr/local/bin/vimdot
vim-8.2.5036-gtk3-lua:editors/vim,-main,gtk3,lua:/usr/local/bin/vim
vim-8.2.5036-gtk3-lua:editors/vim,-main,gtk3,lua:/usr/local/bin/vimdiff
vim-8.2.5036-gtk3-lua:editors/vim,-main,gtk3,lua:/usr/local/bin/vimtutor
vim-8.2.5036-gtk3-perl-python3-ruby:editors/vim,-main,gtk3,perl,python3,ruby:/usr/local/bi
vim-8.2.5036-gtk3-perl-python3-ruby:editors/vim,-main,gtk3,perl,python3,ruby:/usr/local/bi
vim-8.2.5036-gtk3-perl-python3-ruby:editors/vim,-main,gtk3,perl,python3,ruby:/usr/local/bi
vim-8.2.5036-gtk3-python3:editors/vim,-main,gtk3,python3:/usr/local/bin/vim
[...]
```

## pkglocatedb 2

- It is very efficient: 300MB compress to 23MB
- It is fast
- It is in the base system

## locate vs updateinfo

- generate data with `pkgname:update-info-line`
- this should compress correctly
- where to put it to make this accessible
- I did a script that worked. Compression is okay (compresses 23M to 3M)

## chicken and egg too

- I gave the script to my fellow builders and asked for pkgindex.tgz to be on the mirrors
- they did it for a while, but I got distracted
- and then they no longer did it
- right when I got motivated again

## so better integration

- it had to be on, all the time. Add glue at the end of dpb to generate it ?
- delivery system. Sign it specifically ? teach `pkg_add` how to read it ?
- scrape that, let's use quirks

## quirks ?

- Quirks is the package that holds "Exceptions" to the rules (such as package renames, or packages that got dropped).
- First action of `pkg_add` ever is always to try to update quirks.
- So it's a natural location to drop update info

## first experiment

- so I got the script that builds the db into quirks
- told my friends to always regenerate quirks at the end
- and waited for the new package to show up

## Timings: whazzza

- (there was a small issue with "always-update" packages, let's avoid them)
- try to grab the updateinfo from the locate before going to the packages
- result **over twenty times speed-up**
- so worth making it work

## details

- the db is linked to a given quirks, which means a given package repository.
- this is not a big issue because we got unique objects for repositories
- furthermore, quirks is an "always-update" package, so if we find we don't need to update it, it means the quirks we got contains update info for our packages
- we can actually put that in production !

## more speed-ups I

- we run a separate locate for each updateinfo
- we can actually run a single locate upfront, because we got the list of pkgnames we want to handle

```
sub prime_update_info_cache
{
        my ($self, $state, $setlist) = @_;

        my $progress = $state->progress;
        my $found = {};

        my $pseudo_search = [$self];

        for my $set (@{$setlist}) {
                for my $h ($set->older, $set->hints) {
                        next if $h->{update_found};
                        my $name = $h->pkgname;
                        my $stem = OpenBSD::PackageName::splitstem($name);
```

## more speed-ups II

```
                                next if $stem =~ m/^\.libs\d*\-/;
                                next if $stem =~ m/^partial\-/;
                                $stem =~ s/\%.*//; # zap branch info
                                $stem =~ s/\-\-.*//; # and set flavors
                                $self->add_stem($stem);
                        }
                }
                my @list = sort keys %{$self->{stems}};
                return if @list == 0;

                my $total = scalar @list;
                $progress->set_header(
                    $state->f("Reading update info for installed packages",
                        $total));
                my $done = 0;
                my $oldname = "";

                open my $fh, "-|", $self->pipe_locate(map { "$_-[0-9]*"} @list)
```

## more speed-ups III

```perl
            or $state->fatal("Can't run locate: #1", $!);
        while (<$fh>) {
                if (m/^(.*?)\:(.*)/) {
                        my ($pkgname, $value) = ($1, $2);
                        $found->{OpenBSD::PackageName::splitstem($pkgname)} = 1;
                        $self->{raw_data}{$pkgname} //= '';
                        $self->{raw_data}{$pkgname} .= "$value\n";
                        if ($pkgname ne $oldname) {
                                $oldname = $pkgname;
                                $done++;
                        }
                        $progress->show($done, $total);
                }
        }
        close($fh);
        return unless $state->defines("CACHING_VERBOSE");
        for my $k (@list) {
                if (!defined $found->{$k}) {
```

# more speed-ups IV

```
                                $state->say("No cache entry for #1", $k);
                        }
                }
        }
```

## always-update

- at first those were not handled at all
- it means a package that needs an update each time it changes
- after a few tries, I decided that storing a crypto hash would work
- so now it is `@option always-update <hash value>`
- and `pkg_create` generates it

## the generation of quirks

Rougly ten lines in dpb:

```
if ($state->{all}) {
        my $core = DPB::Core->get;
        my $w = DPB::PkgPath->new('devel/quirks');
        if ($state->{engine}{built_packages}) {
                $state->grabber->clean_packages($core, $w->fullpkgpath);
        }
        my $subdirlist = {};
        $w->add_to_subdirlist($subdirlist);
        $state->grabber->grab_subdirs($core, $subdirlist, undef);
        $state->engine->check_buildable;
        $core->mark_ready;
        main_loop();
}
```

## that's all folks

Questions ? more fun details in the summer week