Arbre des ports d'OpenBSD

Charles Neyrand

charles.neyrand@epita.fr

LRE Summer Week
29 juin 2023

Table des Matières

Qu’est ce que l’arbre des ports ?

- L’infrastructure de compilation des packages d’OpenBSD.
- Un ensemble de Makefiles, de scripts et de patchs.
- Les sources sont récupérées soit depuis le disque soit par HTTPS.

Un template de Makefile

Quelques règles et variables...

`bsd.port.mk(5)` est le patron des autres Makefile de l'arbre, il définit plusieurs cibles, dont install.

FLAVOR

Certains packages peuvent avoir différentes options de compilation. Pour les spécifier, on utilise les flavors, un mécanisme propre à OpenBSD apparu depuis la version 2.7.

Un bref interlude

- Smt est desactivé par défaut sur OpenBSD.
- La moitié des coeurs sont désactivés.

Un bref interlude

- Changement de version majeure en avril.
- Un problème d'UEFI sur quelques ordinateurs.
- Débogage à coup de mails et de patchs.
- Compilation de GENERIC et non GENERIC.MP

Comment compiler tous les ports ?

Un simple script shell pourrait suffire à tout compiler...

Un ordre de grandeur

Il y a plus de 11 000 packages qui sont construits lors d'une compilation complète. Sur 4 machines, avec les outils optimisés cela prend 32h. Il y a deux snapshots pour l'architecture amd64 par semaine, plus les snapshots des autres architectures...

Le biais du survivant

Nous ne sommes pas au courant des utilisateurs qui n'ont pas de problème.

- Par contre la plupart des bugs nous sont remontés rapidement.

DPB (1)

Distributed Ports Builder

- Permet la compilation des packages sur une machine ou un cluster de machine.
- Permet de réparer au vol des problèmes de compilation
- Optimise l'ordre de la compilation des packages
- Passage d'un snapshot tous les 15 jours à 2 par semaine

DPB (1)

Distributed Ports Builder

- Quelques gros packages, quelques moyens, beaucoup de petits.
- Développement d'une heuristique particulière.
- S'assurer que les dépendances sont construites en amont, afin d'éviter des goulots d'étranglement.

Cluster de machines

Distributed Ports Builder

- Une façon simple de distribuer la compilation.
- Très simple à simuler :
  - Déclarer deux hôtes locaux (localhost et myCluster)
  - Créer un fichier de configuration et le passer à `DPB(1)`

PROOT(1)

Ports Chroot Builder

- Un excellent jeu de mots...
- Facilite grandement la mise en place de l'environnement de DPB(1).

Le modèle de sécurité de `DPB(1)`

- Un utilisateur pour récupérer les sources par https : `_pfetch`
- Un utilisateur pour compiler, sans accès à internet : `_pbuild`
- `DPB(1)` démarre en tant que root et drop ses privilèges au fur et à mesure.
- Le mieux est de se chroot dans un dossier avant de lancer `DPB(1)`.
- Comment mettre en place ce dossier facilement ?

PROOT(1)

- Remplit le dossier pour chroot
- Privilégie les liens au lieu de la copie

/usr/ports

- Les outils sont écrits en PERL.
- Ils se situe dans le dossier `infrastructure` de l'arbre des ports.
- Pour DPB, une partie est dans bin, l'autre dans lib/DPB

State.pm I

```
1  sub handle_build_files($state)
2  {
3          return if $state->{fetch_only};
4          return unless defined $state->{build_files};
5          print "Reading build stats...";
6          for my $file (@{$state->{build_files}}) {
7                  $state->parse_build_file($file);
8          }
9          $state->heuristics->calibrate(DPB::Core::In
           ↪  it->cores);
10         $state->add_build_info($state->heuristics,
           ↪  "DPB::Job::Port");
11         if (defined $state->{permanent_log}) {
12                 print "zapping old stuff...";
13                 $state->rewrite_build_info($state->
                   ↪  {permanent_log});
14                 print "Done\n";
15         }
```

State.pm II

```
16              $state->heuristics->finished_parsing;
17      }
18
19
20
```

rewrite_build_info I

```
1  my $i = 0;
2  for my $s (@{$p->{stats}}) {
3      $i++;
4      last unless $i <= $state->{stats_backlog};
5      print $f DPB::Serialize::Build->write($s), "\n"
6          or return 0;
7  }
8
9
```

Config.pm I

```
1  if ($state->define_present('STATS_BACKLOG')) {
2      $state->{stats_backlog} =
       ↪  $state->{subst}->value('STATS_BACKLOG');
3  }
4  $state->{stats_backlog} //= 25;
5
6  if ($state->define_present('STATS_USED')) {
7      $state->{stats_used} =
       ↪  $state->{subst}->value('STATS_USED');
8  }
9  $state->{stats_used} //= 10;
10
```

Nouvelles Options

```
doas dpb -B /build -P ~/localports -h ~HostDPB
  ↪  -DCOLOR=1 -DSTATS_BACKLOG=30
```

```
29 Jun 08:57:17 [6530] control-lse-6530 elapsed: 00:00:14
LISTING [22272] at editors/xemacs21/stable,mule
editors/elvis(build) [64088] 62%
editors/fte(build) [54279] on mycluster 2%
editors/hexcurse(package) [82742] on mycluster 114%
editors/le(build) [2232] 3%
~editors/se(build) [47862] 24%
~editors/vim-spell/am(package) [9859] on mycluster 122%
editors/vim-spell/hu(build) [69506] on mycluster 19%
Hosts: localhost mycluster [89081]
I=4 B=0 Q=44 T=91 F=0 ≠0
```

Figure – DPB

Bibliographie I

PORTS(7)
OpenBSD Man pages.
Available online : https://man.openbsd.org/ports.7

BULK(8)
OpenBSD Man pages.
Available online : https://man.openbsd.org/bulk.8

EuroBSDCon 2022, Sep 15-18, 2022, Vienna, Austria
Marc Espie - *Architectures vs the ports tree : a losing battle ?*
Available online : https://www.openbsd.org/papers/eurobsdcon2022-espie-arches.pdf

EuroBSDCon 2019, September 19-22, 2019, Lillehammer, Norway
Marc Espie - *Advanced ports toolkit : near-perfect packing-list generation*
Available online : https://www.openbsd.org/papers/eurobsdcon2019-plist.pdf

Questions?