Cross build in the FreeBSD ports tree

Baptiste Daroussin
bapt@FreeBSD.org

EuroBSDCon 2014
Sofia - Bulgaria
September 28, 2014

Goals

- Building packages for Tiers-2 arches
- Building packages for low power machines
- Building boostrap packages for non self hosting languages
- Building "emulation" ports (aka linuxulator)

Easier way: qemu user emulation

- binary image activator
- 18k packages sucessfully build for armv6 (thanks sbruno!)
- Requires no particular modification of the ports tree beside

Easier way: qemu user emulation

- binary image activator
- 18k packages sucessfully build for armv6 (thanks sbruno!)
- Requires no particular modification of the ports tree beside
- qemu-bsd-user is still buggy and fragile
- slow

Hybrid way: qemu user emulation + native cross tools

- use qemu-bsd + binary image activator
- native cross toolchain

Hybrid way: qemu user emulation + native cross tools

- use qemu-bsd + binary image activator
- native cross toolchain
- qemu-bsd-user is still buggy and fragile
- still slow

The one true way: cross compilation

- Faster
- Simpler
- Easier to use for regular users

The one true way: cross compilation

- Faster
- Simpler
- Easier to use for regular users
- overhead some ports are built twice

Build systems

- Good Players:

Build systems

- Good Players:
    - autotools:

Build systems

- Good Players:
  - autotools: really works out of box

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...

FreeBSD

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...
  - cmake

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...
  - cmake
  - /usr/share/mk/* (somehow)

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...
  - cmake
  - /usr/share/mk/* (somehow)
- The bad players

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...
  - cmake
  - /usr/share/mk/* (somehow)
- The bad players
  - scons

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...
  - cmake
  - /usr/share/mk/* (somehow)
- The bad players
  - scons
  - The cusom home made build systems

FreeBSD

Build systems

- Good Players:
  - autotools: really works out of box ... when used correctly...
  - cmake
  - /usr/share/mk/* (somehow)
- The bad players
  - scons
  - The cusom home made build systems
  - ./please_build_me.sh

Main complications

Main complications

- Perl
    - Cross build friendly

Main complications

- Perl
    - Cross build friendly

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...

FreeBSD

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost
  - try to run the built python instead of a native one :(

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost
  - try to run the built python instead of a native one :(
  - FreeBSD ports wtf

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost
  - try to run the built python instead of a native one :(
  - FreeBSD ports wtf (fixed now thanks python@)

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost
  - try to run the built python instead of a native one :(
  - FreeBSD ports wtf (fixed now thanks python@)
  - Working patches available for very very long still not fully in python 3.3
  - Python 3.4?
- OpenJDK

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost
  - try to run the built python instead of a native one :(
  - FreeBSD ports wtf (fixed now thanks python@)
  - Working patches available for very very long still not fully in python 3.3
  - Python 3.4?
- OpenJDK
  - Cross build friendly

Main complications

- Perl
  - Cross build friendly
  - by requiring a ssh connection to a target server ...
- Python
  - Cross build friendly ... almost
  - try to run the built python instead of a native one :(
  - FreeBSD ports wtf (fixed now thanks python@)
  - Working patches available for very very long still not fully in python 3.3
  - Python 3.4?
- OpenJDK
  - Cross build friendly ... It really is!

FreeBSD

Toolchains

Toolchains

- Clang
  - a cross build friendly compiler

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches
  - Not really a cross build friendly compiler

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches
  - Not really a cross build friendly compiler
- No consistent behaviour between gcc and clang

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches
  - Not really a cross build friendly compiler
- No consistent behaviour between gcc and clang
- binutils

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches
  - Not really a cross build friendly compiler
- No consistent behaviour between gcc and clang
- binutils
  - FreeBSD patches for arm were missing

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches
  - Not really a cross build friendly compiler
- No consistent behaviour between gcc and clang
- binutils
  - FreeBSD patches for arm were missing
  - Cross build friendly for all

Toolchains

- Clang
  - a cross build friendly compiler
  - number of targets very limited (only sane arm on FreeBSD)
- GCC
  - gcc 4.2 ... real world needs a modern compiler
  - FreeBSD people never upstream lots of patches
  - Not really a cross build friendly compiler
- No consistent behaviour between gcc and clang
- binutils
  - FreeBSD patches for arm were missing
  - Cross build friendly for all ... but gas

Making a cross building environment

- make xdev

Making a cross building environment

- make xdev
  - Create a sysroot
  - Create a cross build toolchain

Making a cross building environment

- make xdev
  - Create a sysroot
  - Create a cross build toolchain
  - Inconsistent over versions
  - gcc/clang problems
- Use clang
  - clang from base is available and recent enough
  - fall back on clang from ports otherwise
  - use binutils from ports all the time

Making a cross building environment

- make xdev
  - Create a sysroot
  - Create a cross build toolchain
  - Inconsistent over versions
  - gcc/clang problems
- Use clang
  - clang from base is available and recent enough
  - fall back on clang from ports otherwise
  - use binutils from ports all the time ...requires fixing our *.S files

Making a cross building environment

- make xdev
  - Create a sysroot
  - Create a cross build toolchain
  - Inconsistent over versions
  - gcc/clang problems
- Use clang
  - clang from base is available and recent enough
  - fall back on clang from ports otherwise
  - use binutils from ports all the time ...requires fixing our *.S files
  - create a ports cross building aware version of freebsd

Making a cross building environment

- make xdev
  - Create a sysroot
  - Create a cross build toolchain
  - Inconsistent over versions
  - gcc/clang problems
- Use clang
  - clang from base is available and recent enough
  - fall back on clang from ports otherwise
  - use binutils from ports all the time ...requires fixing our *.S files
  - create a ports cross building aware version of freebsd ... requires upstreaming our patches

Making a cross building environment

- make xdev
  - Create a sysroot
  - Create a cross build toolchain
  - Inconsistent over versions
  - gcc/clang problems
- Use clang
  - clang from base is available and recent enough
  - fall back on clang from ports otherwise
  - use binutils from ports all the time ...requires fixing our *.S files
  - create a ports cross building aware version of freebsd ... requires upstreaming our patches

Making a cross building environment (create the sysroot)

- $make sysroot:
  make: don't know how to make sysroot. Stop

Making a cross building environment (create the sysroot)

- $make sysroot:
  make: don't know how to make sysroot. Stop
- any way manually that is easy

Making a cross building environment (create the sysroot)

- $make sysroot:
  make: don't know how to make sysroot. Stop
- any way manually that is easy

```
TARGET?= arm
TARGET_ARCH?= armv6
XCFLAGS= isystem ${WRKDIR}/tmp/usr/include -L${WRKDIR}/tmp/usr/lib \
--sysroot=${WRKDIR}/tmp/ -B${LOCALBASE}/arm-gnueabi-freebsd/bin \
-B/usr/bin \
-target armv6-gnueabi-freebsd10.0
XMAKE_ENV= PATH=${LOCALBASE}/arm-gnueabi-freebsd/bin:/usr/bin:/usr/sbin:/bin \
WITHOUT_PROFILE=yes __MAKE_CONF=/dev/null SRCCONF=/dev/null \
NO_FSCHG=yes MAKEOBJDIRPREFIX=${WRKDIR}/obj \
TARGET=${TARGET} TARGET_ARCH=${TARGET_ARCH} \
MACHINE=${TARGET} MACHINE_ARCH=${TARGET_ARCH} \
_SHLIBDIRPREFIX=${WRKDIR}/tmp \
CC="${CC} ${XCFLAGS}" \
CPP="${CPP} ${XCFLAGS}" \
CXX="${CXX} ${XCFLAGS}" \
NO_WERROR=yes NO_WARNS=yes
NOFUN= -DNO_FSCHG MK_HTML=no MK_INFO=no -DNO_LINT \
MK_MAN=no MK_NLS=no -DNO_PROFILE \
MK_KERBEROS=no MK_RESCUE=no MK_TESTS=no -DNO_WARNS
cd ${WRKSRC}/lib/ncurses/ncurses ; \
MAKEOBJDIRPREFIX=${WRKDIR}/obj make build-tools
cd ${WRKSRC}; \
mtree -R uid,gid -deU -f etc/mtree/BSD.root.dist -p ${WRKDIR}/tmp >/dev/null ; \
mtree -R uid,gid -deU -f etc/mtree/BSD.usr.dist -p ${WRKDIR}/tmp/usr >/dev/null ; \
mtree -R uid,gid -deU -f etc/mtree/BSD.include.dist -p ${WRKDIR}/tmp/usr/include >/dev/null ; \
setenv -i ${XMAKE_ENV} WITHOUT_MAN=yes -f Makefile.inc1 par-includes libraries \
DESTDIR=${WRKDIR}/tmp
```

Changes to the ports infrastructure

Variable set when cross building

- HCC/HCXX (host compiler)
- CC/CXX (set to the cross compiler + special flags)
- STRIP_CMD to the cross binutils version
- ABI_FILE=${X_SYSROOT}/usr/lib/crt1.o
- PKG_CONFIG_SYSROOT_DIR="${X_SYSROOT}"

Changes to the ports infrastructure

Behaviour changed

- LIB_DEPENDS BUILD_DEPENDS are built twice: native and target
- native are installed on the host
- target are installed to the sysroot
- Automatically add dependencies to sysroot (if not provided) and toolchain

Changes to the ports infrastructure

Behaviour changed

- LIB_DEPENDS BUILD_DEPENDS are built twice: native and target
- native are installed on the host
- target are installed to the sysroot
- Automatically add dependencies to sysroot (if not provided) and toolchain

tweaks have to be done ports by ports

FreeBSD

Ports tweak

Perl

- perl-cross (unofficial)
- provide config.h per supported architecture/freebsd version

Ports tweak

Perl

- perl-cross (unofficial)
- provide config.h per supported architecture/freebsd version

Python

- patch python 2.7 to 3.3 to use native python
- check python 3.4

Ports tweak

Perl

- perl-cross (unofficial)
- provide config.h per supported architecture/freebsd version

Python

- patch python 2.7 to 3.3 to use native python
- check python 3.4

Scons

- impossible to get a global solution
- use a saner build system

Ports point of view

Without sysroot

```
# cd devel/pkgconf
# make X_BUILD_FOR=armv6-gnueabi-freebsd10.0 package
```

With sysroot

```
# cd devel/pkgconf
# make X_BUILD_FOR=armv6-gnueabi-freebsd10.0 \
X_SYSROOT=/path/to/sysroot package
```

Limitations

- ports requiring a different compiler than the default are not supported (meaning openmp and non libc++ ports ports using C nested functions)
- plateforms using gcc as a default compiler doesn't work

Thank you!

Questions ?

FreeBSD