How to Build OpenOffice Today

Virtual Machines and Containers

Arrigo Marchiori <ardovm@apache.org>

FOSDEM 2022

Overview

- What Can We Build?
- The Build System
- Building on Linux
- Building on Windows
- Building on macOS
- Building on FreeBSD
- Conclusion

Credits
And Where to Get Help

1) https://wiki.openoffice.org
   The Building Guide

2) dev@openoffice.apache.org
   The Development Mailing List

What Can We Build?
Choices Need To Be Made

Architectures

- Linux
- Windows
- macOS
- FreeBSD
- OS/2

Branches

- AOO41X
- AOO42X
- trunk

With several **options**: languages, internal/external libraries, extensions, debugging symbols, etc.

What Can We Build?
Architectures

- To build for an architecture, you need a development system with that architecture
  - Cross-compiling is not supported.

- Bare Metal
- Containers
- VM's

A decent (virtual) machine must have at least 50 GB free disk space and 8 GB RAM

What Can We Build?

Branches

AOO41X

- Stable and dependable
- Older dependencies
- Linux: older distro

AOO42X

- Will be stable tomorrow
- Still has some show stopper bugs

trunk

- May not compile, may not run
- New features are first merged here

What Can We Build?
Compilation Options

More than 180 parameters to the `configure` script

- Languages
  - English, German...
- Packaging
  - Portable, installer...
- Components
  - Quickstarter, dicts...

More than 60 influential environment variables

- Debugging options
  - Symbols, tracing
- Additional packages
  - Ant, dmake, EPM...
- Arch-specific libs
  - Cairo, DirectX, GTK...

What Can We Build?
“Your” Build vs “Our” Build

- Many variables can make your own build unique...
- ...or introduce subtle errors!
- Official **build scripts** ensure that everyone is following the same path as the official releases.
- Some scripts still allow limited choices
  - Installed format (portable vs. packaged)
  - Languages (how many of them do you speak?)
  - Debugging symbols

The Build System
Overview

We are standing on the shoulders of giants!

- Perl
- Dmake
- Autotools
- GNU Make
- Apache Ant
- SCons (in progress)

- Some external libraries and tools can be either downloaded (and built), or provided by the system.
- Other “special” tools are part of the sources (e.g. Autodoc)

The Build System
Flow

1. The `configure` script generate a **profile file** that defines (a lot of) environment variables that drive the build.
2. A **bootstrap** script downloads and compiles the tools required for the build.
3. A Perl script (**build.pl**) does the rest of the work.

A build scripts takes care of all these steps

The Build System
Source Code Organization

The source code is organized in modules:

- Programs
  - Writer, Calc...
  - Internal tools
- Libraries
  - External: serf, epm...
  - Internal: spell checker, file conversion...
- Special modules:
  - **solver:** contains all the compiled output
  - **instsetoo-native:** builds the final package

Building on Linux
Overview

AOO41X: CentOS 5

- Using a VM is strongly suggested
- Old OpenSSL makes installation problematic
  - We could provide installed VM images!

AOO42X: CentOS 7

- Dockerfile available!
  - `docker build` to build the container
  - `docker run` and compile!

Build scripts are also available for Ubuntu and openSUSE.

Building on Linux
CentOS 7 Dockerfile

```
FROM centos:7
RUN yum update -y
RUN echo "assumeyes=1" >> /etc/yum.conf
RUN yum install epel-release -y
RUN yum install -y\
    gcc \
 [...]

RUN ccache -M 2G



RUN wget
https://dlcdn.apache.org/ant/binaries/\
        apache-ant-1.9.16-bin.tar.bz2
RUN tar xvf apache-ant-1.9.16-bin.tar.bz2
RUN mv apache-ant-1.9.16 ant
ENV ANT_HOME=/ant

RUN wget https://url_of_{dmake,epm}
RUN tar xvf {dmake,epm}.tar.gz
WORKDIR /{dmake,epm}
RUN ./configure --prefix=/usr/local; \
    make install
```

Container initialization, installation of dependencies

Configure CCache

Download Apache Ant and set ANT_HOME

Download and compile Dmake and EPM

Building on Linux
Using CentOS 7 Dockerfile

```
$ wget https://svn.apache.org/.../linux/Dockerfile
$ docker build -t aoo_centos .
$ docker run -ti -v /sources:/sources aoo_centos
```

- The Dockerfile does not download the source code.
  - You can “bind mount” it into the container.
  - Mount it into the same path to ease debugging.

Building on Linux
Container/VM vs Bare Metal

Containers and VM's:

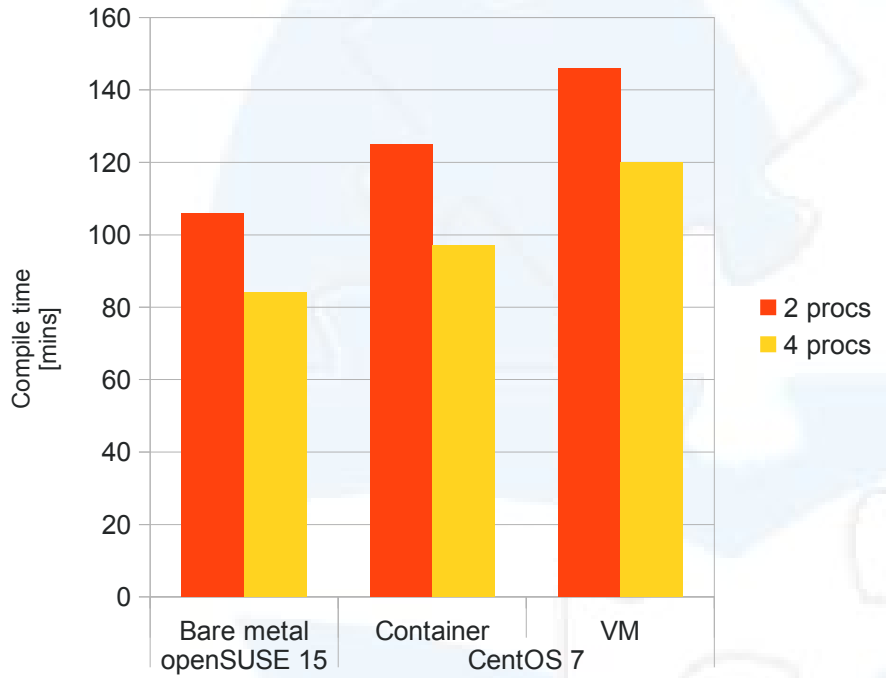
- Keep your system “clean”.
- Allow you to make “official” builds.
- Allow you to test different distros.
- Can be configured automatically

Bare Metal:

- Has full access to your CPU and RAM
- Requires less disk space

Development can be comfortable on either

Building on Linux
Container vs Bare Metal Performance



AOO42X, en-US, DEB & RPM

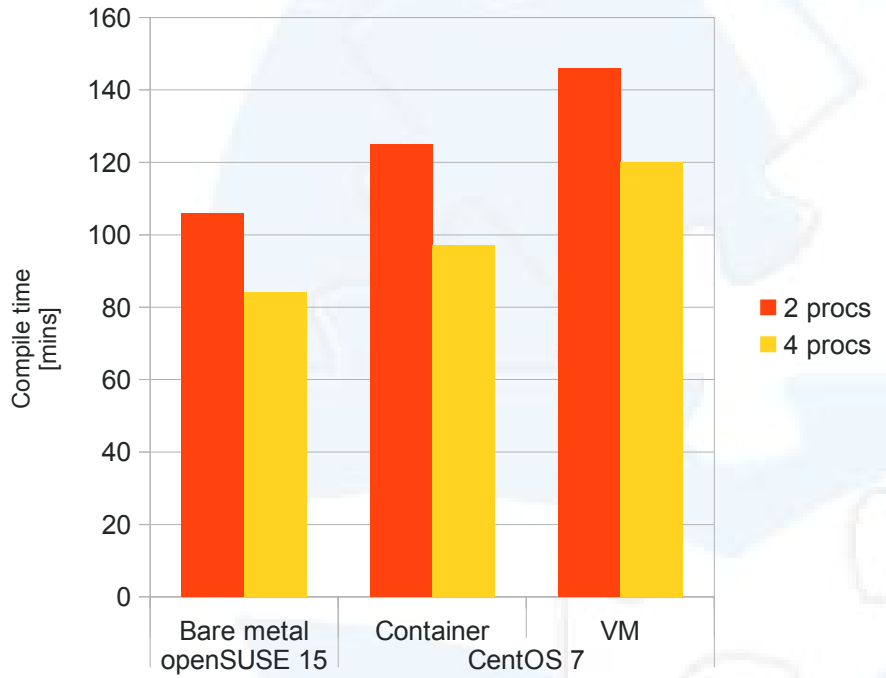
- Container took:
  - 18% longer than bare metal with 2 procs;
  - 15% longer than bare metal with 4 procs;

System: Intel(R) Core(TM) i3-9100F @ 3.60GHz
16 GB RAM, SSD, Ext4
openSUSE Leap 15.3

VM: 4 cores, 12 GB RAM

Docker: default configuration

Building on Linux
VM vs Bare Metal Performance



AOO42X, en-US, DEB & RPM

- VM took:
  - 38% longer than bare metal with 2 procs;
  - 43% longer than bare metal with 4 procs

System: Intel(R) Core(TM) i3-9100F @ 3.60GHz
16 GB RAM, SSD, Ext4
openSUSE Leap 15.3

VM: 4 cores, 12 GB RAM

Docker: default configuration

Building on Windows
Overview

AOO41X:
Cygwin 32-bit

AOO42X:
Cygwin 64-bit

- Visual Studio 2008 required
  - The Build System invokes the compiler
- Some software components (SDK's etc) may be difficult to find today.
  - Contact dev@ !
- Building machines currently run Windows 10
  - Windows 11 is WIP

Building on Windows
VM vs Bare Metal

Virtual Machines:

- Keep your system clean
  - Older MSVC
  - Lots of SDK's
  - Two versions of Cygwin

Bare Metal:

- Has full access to your CPU and RAM
- Requires less disk space
- No extra license required

- Preparation of VM's has to be done by hand.
- Contributions welcome!
  - Contact dev@ !

Building on macOS
Overview

AOO41X

- macOS 10.13 (High Sierra)
- Xcode 11
- MacOSX10.11 SDK

AOO42X

- macOS 10.15 (Catalina)
- Xcode 12.2

Some libraries and tools must be compiled manually or with MacPorts

VM's are a good idea because of the dependencies

Building on FreeBSD
Using the Ports Collection

- The Ports Collection allows you to get an official* build with two commands:

```
$ cd /usr/ports/editors/openoffice-4/
$ make install clean
```

- AOO42X is available as Port `editors/openoffice-devel`
- Debugging symbols: `WITH_DEBUG=yes`

* Most dependencies are taken from the base system or other Ports

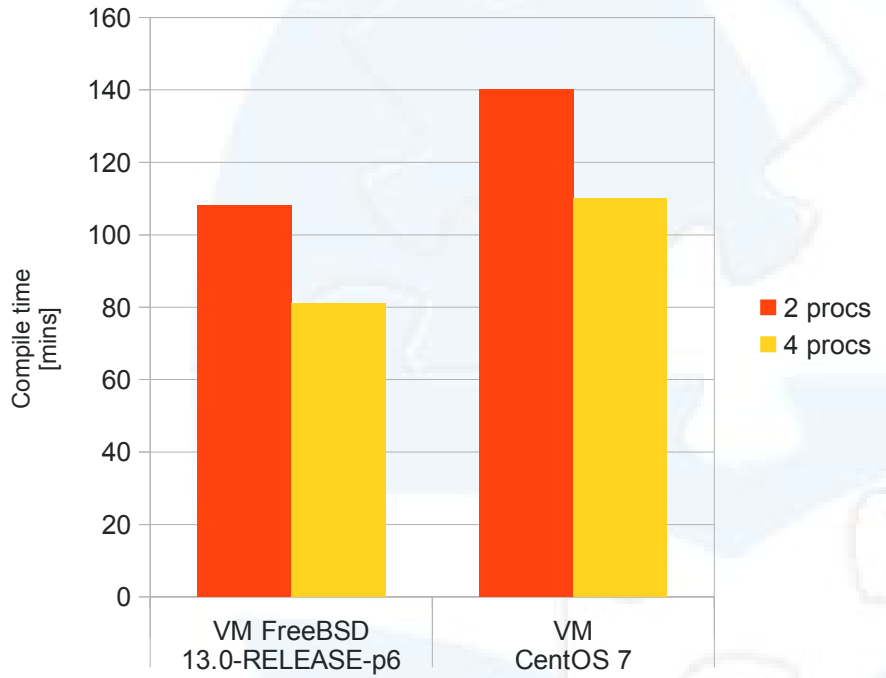
Building on FreeBSD
Your Own Code

- The Ports' maintainers are doing a very good (and very hard) job as FreeBSD keeps evolving
- Use their patches!

```
$ cd main && \
  for p in \
  /usr/ports/editors/openoffice-devel/files/patch*; do \
  patch -p0 < $p; \
  done
```

- The configure script invocation is also specific. Contact dev@ for more information.

Building on FreeBSD
Linux VM vs FreeBSD VM



AOO42X, en-US, “archive”

- FreeBSD took 25% less than CentOS 7, thanks to less dependencies being built

System: Intel(R) Core(TM) i3-9100F @ 3.60GHz
16 GB RAM, SSD, Ext4
openSUSE Leap 15.3

VM: 4 cores, 12 GB RAM

Conclusion

- VM’s and containers a good choice for most users thanks to:
  - The number of dependencies
  - The possibility to test multiple architectures
  - Fair performance
- Room for improvement:
  - Reproducible Linux VM for AOO41X
  - Automatically configured containers/VM's for other architectures
- Contributions welcome!